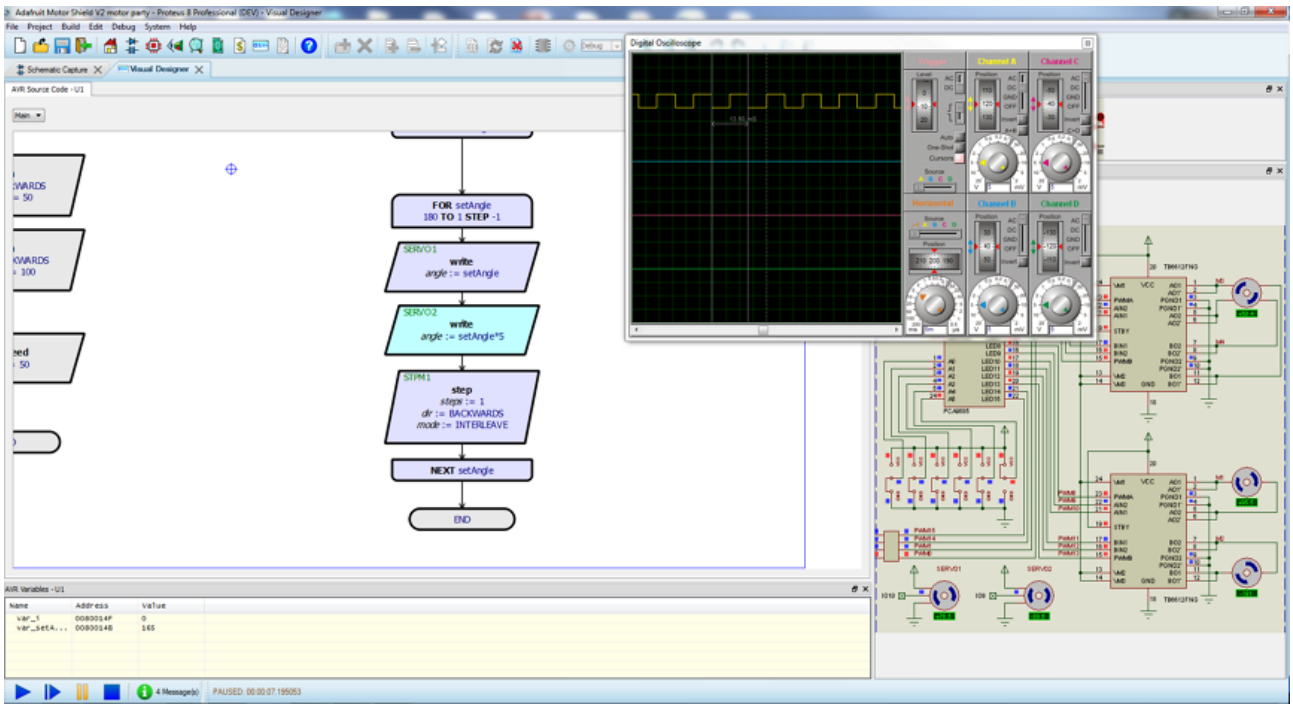
VISUAL DESIGNER

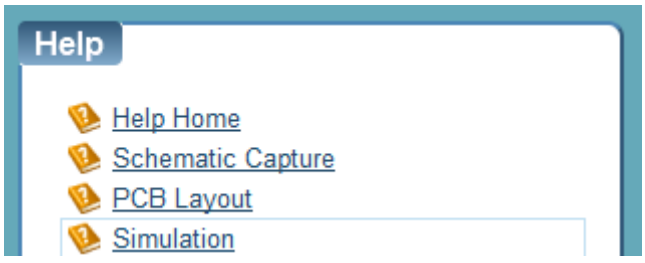
Introduction

Visual Designer is a unique development tool that uses flowchart blocks together with schematic clips of Arduino™ shields or Raspberry Pi hats to allow drag and drop creation of real-world Arduino and rPi embedded systems.



Single step debugging in Visual Designer with the AdaFruit Motor Shield

This document explains the various features of Visual Designer and includes several practical example projects and walkthroughs. Other modules in the Proteus system such as the schematic capture and general simulation have their own help files and are companion resources to this document. You'll find these help files either on the help menu in the schematic capture module or via the Help section on the Proteus home page.



Launchpad for other help files.

What is Arduino™ / Genuino™?

Arduino is an open-source computer hardware and software company, project and user community that designs and manufactures microcontroller-based kits for building digital devices and interactive objects that can sense and control objects in the physical world.

The project is based on microcontroller board designs, manufactured by several vendors, using various microcontrollers. These systems provide sets of digital and analog I/O pins that can be interfaced to various expansion boards ("shields") and other circuits.

See Also:

Arduino Website

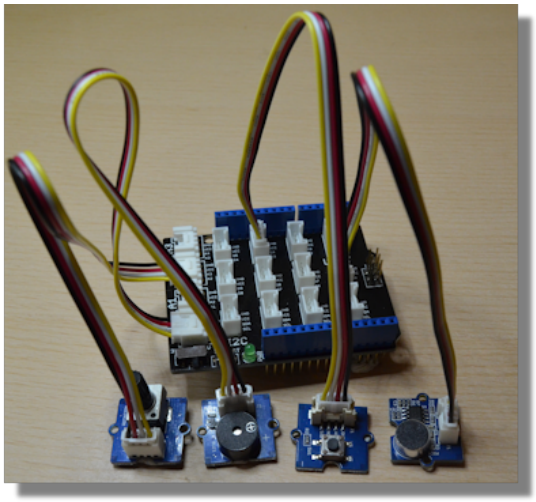
> The Arduino and Genuino names and logos are registered trademarks of Arduino, LLC. The Visual Designer for Arduino simulation product is not directly connected or endorsed by Arduino.

What is Raspberry Pi ?

Raspberry Pi is a series of small, low cost, single board computers. The Raspberry Pi is still a complete Linux computer but is still often used to control external electronics.

What is Grove ?





Grove is a modular electronic platform for quick prototyping. Every module has one function, such as touch sensing, creating audio effect and so on. Just plug the modules you need to the base shield, then you are ready to test your ideas.

This Grove Starter Kit is a great way for beginners and students to get started with Arduino. The Grove Base Shield drops onto your existing Arduino and lets you connect up to 16 Grove modules without any fuss. The Arduino headers are still broken out on the top of the shield making it very easy to connect other breakout boards and sensors to your Arduino.

See Also:

Grove Starter Kit (Website)

Raspberry Pi: Limitations of Simulation

The Raspberry Pi is a high level behavioural model in terms of simulation. Since the rPi runs a full copy of Linux and arbitrary code can be executed a full simulation model is not possible. Specifically, the following libraries are supported:

smbus
pygame
wiringpi
RPi.GPIO
spidev

There is also a set of Python libraries, which are supplied by us in the Raspberry Pi drivers directory:

Adafruit_GPIO
Adafruit_I2C
Adafruit_MCP230xx

Adafruit_MCP3008
Adafruit_MotorHAT
Adafruit_PCA9685
Adafruit_PureIO
automationhat
explorerhat
ADS1x15
cap1xxx
grove_128_64_oled
grove_rgb_lcd
grovepi
max31855
pcf8574
pcf8591
piglow
sn3218

If you are writing your firmware with Visual Designer flowchart blocks then the Raspberry Pi simulation should work with no problems. However, if you convert your project to write your program in Python then of course you can write code that is based on other Python libraries. In such cases you may not be able to simulate your project because your source code may pull in libraries unsupported in simulation.

GUIDED TOUR

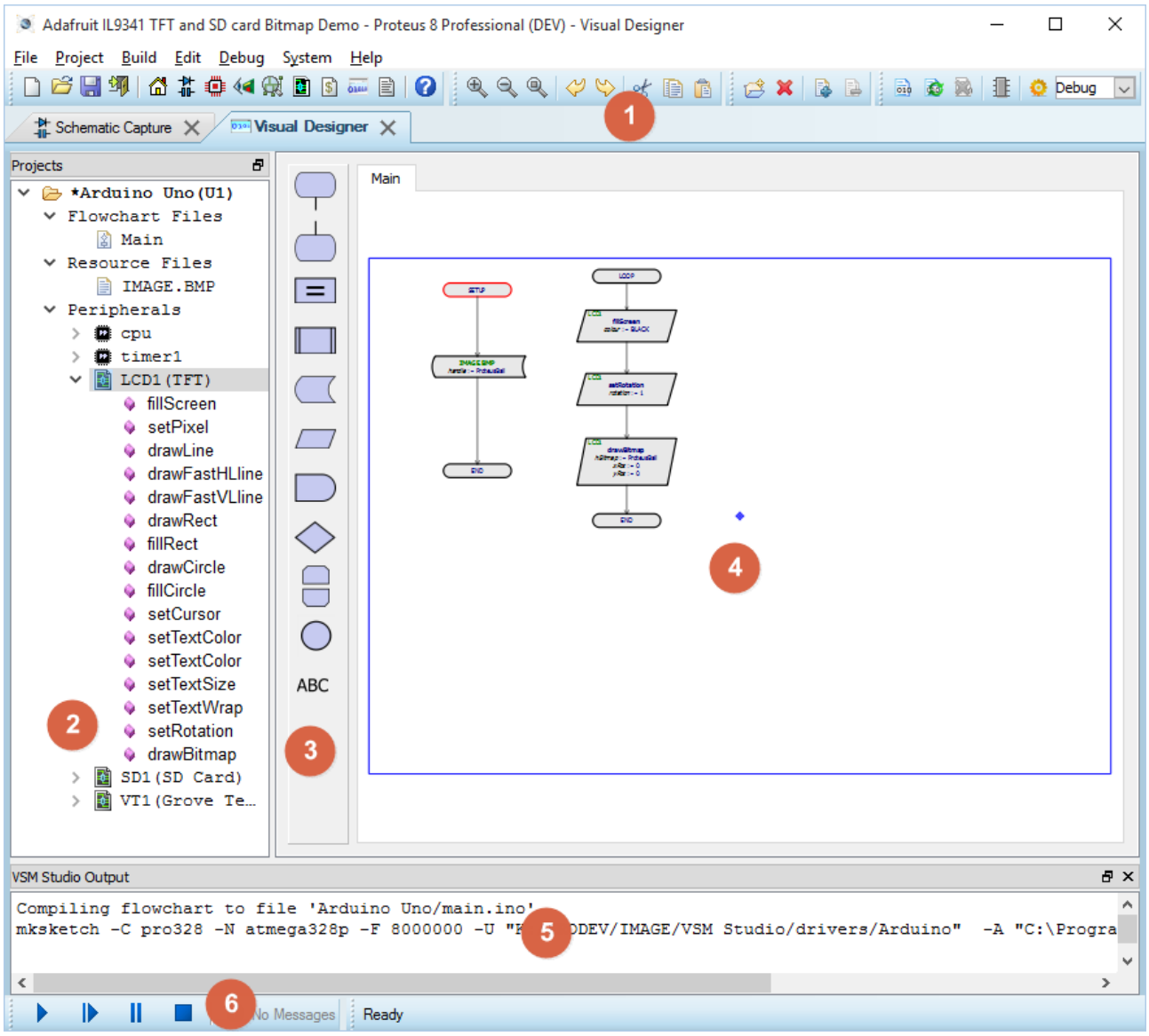
Overview

Visual Designer is a tool for designing embedded systems via a simple flowchart interface and then simulating and debugging the entire embedded system in software. It makes sense therefore that the IDE changes and provides both an Editing Layout and a Debugging Layout. This topic introduces both environments and includes links to more detailed topics where appropriate.

Visual Designer Environment

Editing Layout

At design time, we are creating our embedded system, adding both hardware shields and embedded control logic. The Visual Designer Environment breaks down into six main areas as shown below



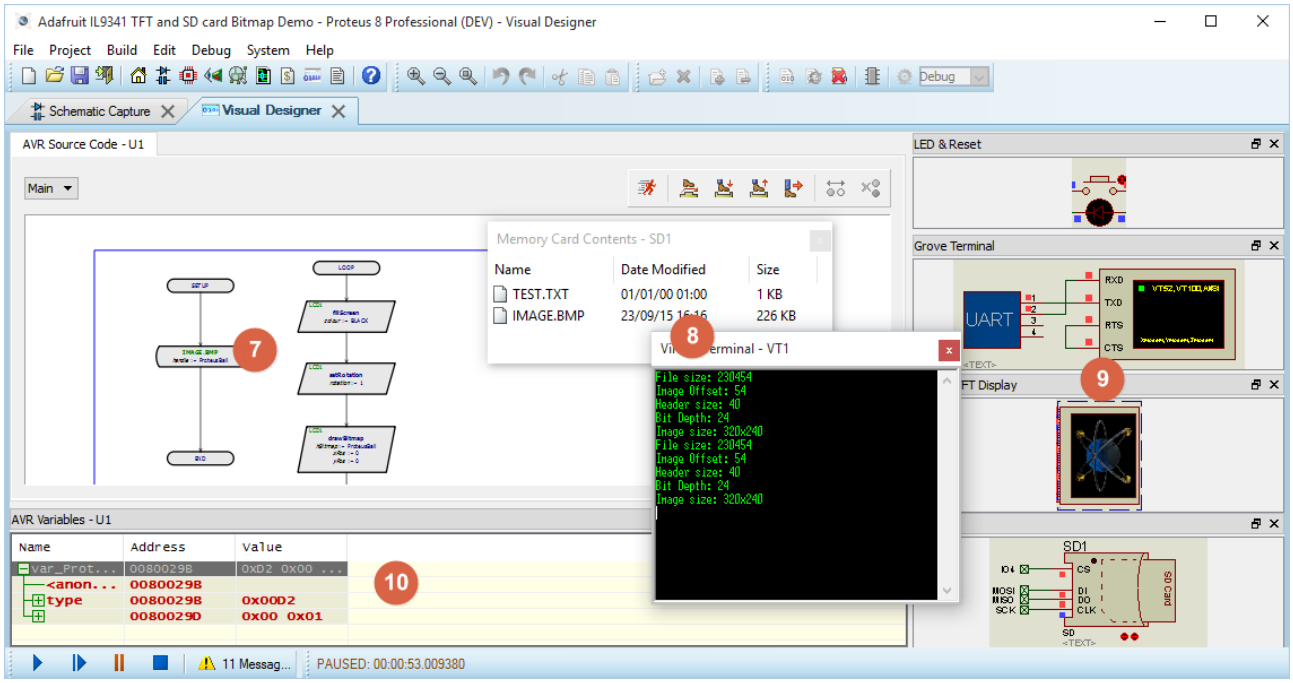
The Visual Designer IDE at Design Time.

1) Menus, Icons and Tabs	2) Project Tree
3) Flowchart Blocks	4) Editing Window

5) Output Window	6) Simulation Control Panel

Debugging Layout

During simulation and debug, the environment changes to provide relevant tools.

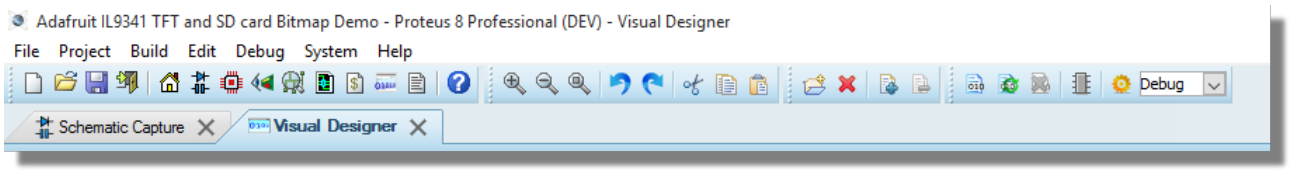


The Visual Designer IDE when single step debugging a simulation.

7) Source Code Window	8) Pop-up Peripheral Windows
9) Active Popups	10) Variables Window / Debug Windows

These items are all discussed in the following sections and are also covered in a more practical way in the tutorials.

1) Menu's, Icons and Tabs

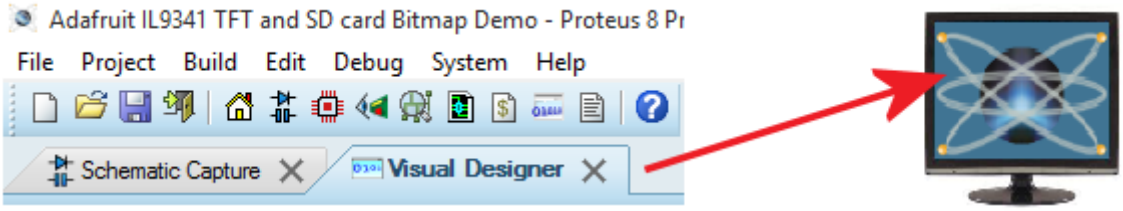


The top menu commands will be familiar to most users and specific functionality is discussed where relevant in the tutorial and other areas of the documentation. Some of the iconography may be new however and each icon is therefore listed below along with its use:

ICON	DESCRIPTION	ICON	DESCRIPTION	ICON	DESCRIPTION
	New Proteus Project		Open VSM Studio / Visual Designer		Create New Firmware Project
	Open Project		Open Project Notes		Delete Existing Firmware Project
	Save Project		Open the Help files		Add Source Files to the Existing Project

	Close Project		Zoom In		Remove Source Files to the Existing Project
	Open Proteus Home Page		Zoom Out		Build / Compile Project
	Open Schematic Capture Program		Zoom All		Rebuild / Compile Project
	Open the PCB Layout Program		Undo		Stop Build
	Open the 3D Viewer		Redo		Attached to IDE / Hardware
	Open the Gerber Editor		Cut		Project Settings
	Open the Design Explorer		Copy	Debug	Type of Build
	Open the Bill of Materials		Paste		

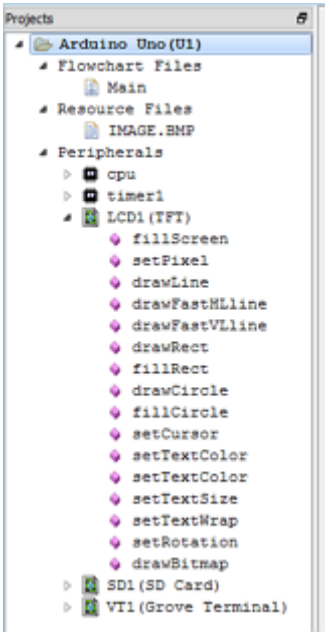
The tabs allow you to switch between different areas of the Proteus Design Suite. We will be working primarily inside Visual Designer but we are building an embedded system and that creates a schematic design as well. The schematic will be automatically drawn as we add shields and peripherals to Visual Designer but you can switch to the Schematic tab at any time to look at your 'virtual hardware'. You can even drag and drop a tab onto a different monitor if you want to look at both modules simultaneously.



Drag and Drop Tabs onto a second monitor to split screen.

Having virtual hardware in the form of a schematic really shows its power during simulation as you can take measurements anywhere you please. You simply place and wire instruments such as oscilloscopes and logic analyzers to inspect waveforms. See the tutorial for more information and examples.

2) Project Tree

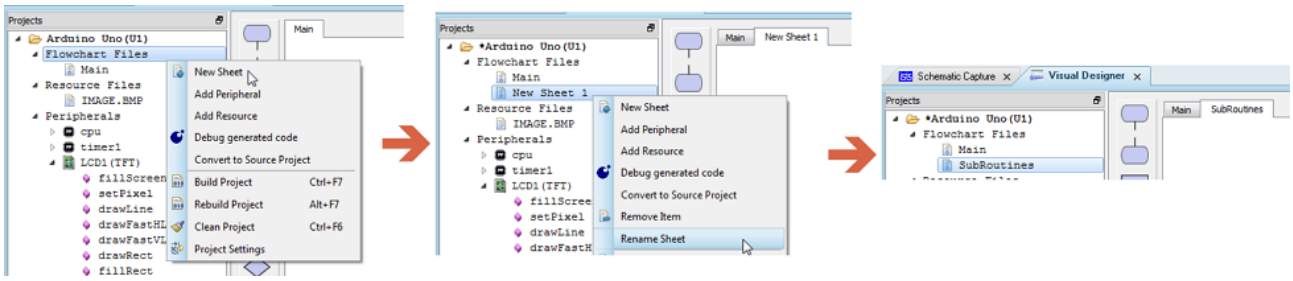


The project tree in Visual Designer performs three main roles:

- Control of Flowchart Sheets
- Control of Resources in the Embedded System
- Control of Hardware Peripherals in the Embedded System

Control of Flowchart Sheets

When you start a new project you will have by default a single sheet in the Editing Window called 'Main'. You can add more sheets (for example a subroutines sheet) if your program is likely to become quite large.



Adding and naming a second sheet in the Flowchart.

When you have more than one sheet in your project you can quickly move between sheets by double clicking on the sheet name in the Project Tree. Alternatively, you can select the sheet you wish to navigate to from the sheet tabs at the top of the Editing Window.

> You can also find the sheet commands on the Project Menu at the top of the Visual Designer IDE.

Control of Resource Files

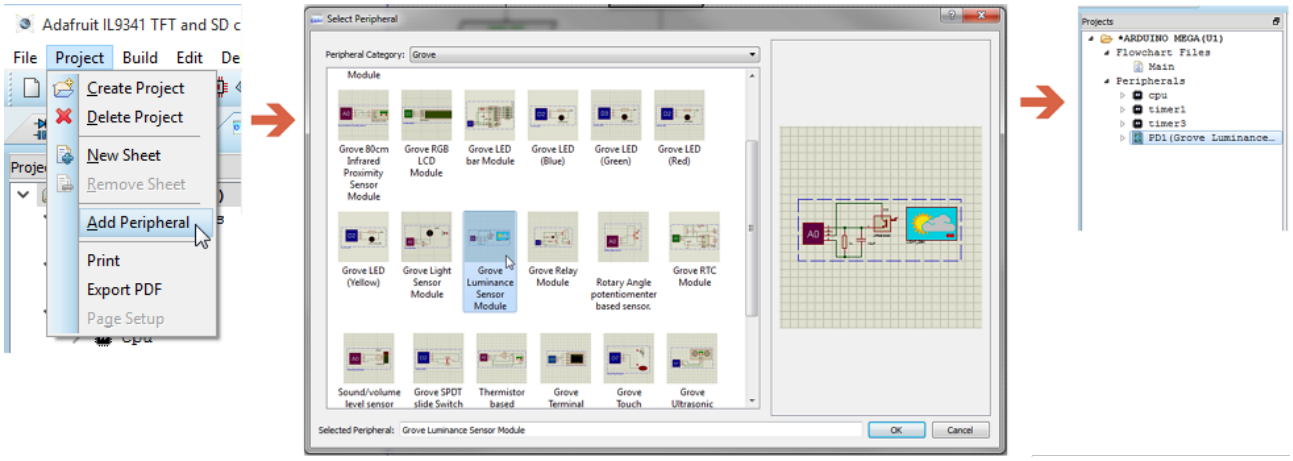
Resource files provide a great way to include pictures and audio files into your project. You can add /remove resource files from the right click context menu on the Project Tree. Once you have a resource file you can drag and drop it straight into the setup routine of your flowchart program to assign it a handle.

See Also:

- Resource Files and Data Storage.
- Worked Example: Bitmap Resource & TFT Display.

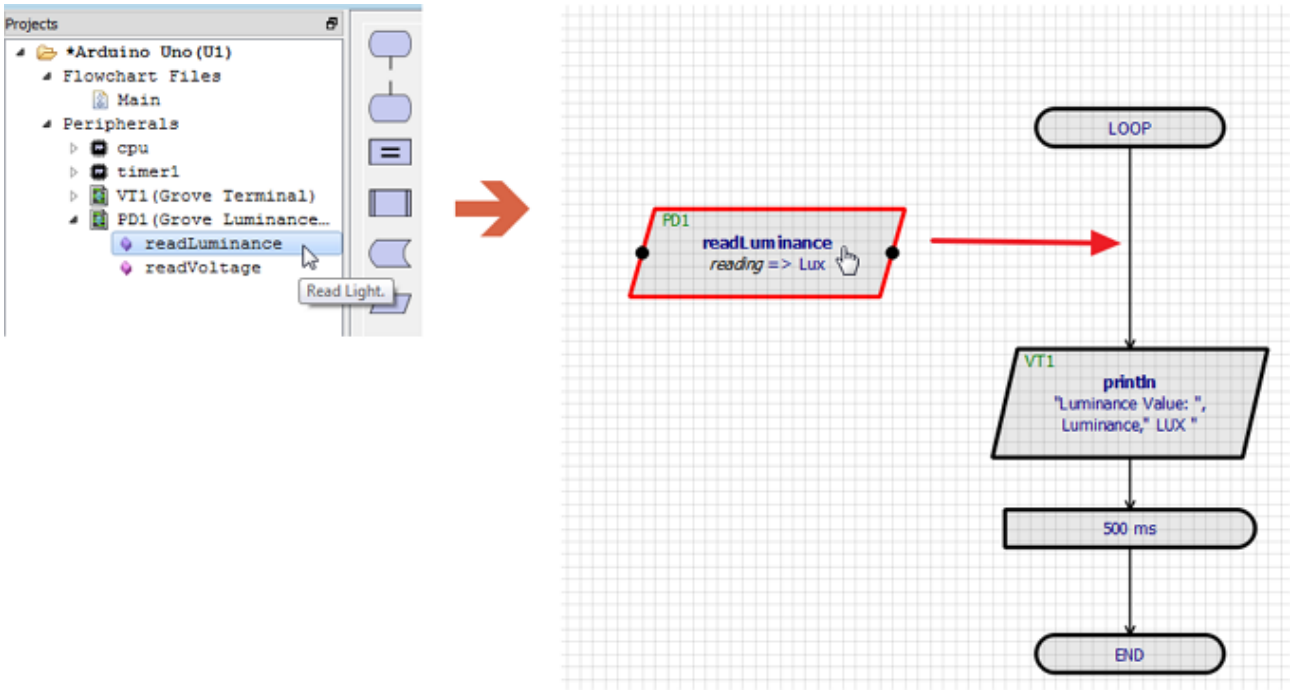
Control of Hardware Peripherals

Visual Designer is a development environment for a complete embedded system and so the peripherals block includes both on-board peripherals of the CPU and also external peripherals (supported Arduino shields or Grove Sensors) that you add to build up your hardware design. When you start a new project you will see two or three peripherals such as CPU core and Timer1 related to the processor core and you add extra peripherals via the right click context menu on the Project Tree or the command on the Project Menu.



Adding a LUX sensor to the current Flowchart Project

Once added, you can see all of the methods available to you. These are the means by which you interact with the hardware and you simply drag and drop the method into your flowchart program where it will appear as an I/O Peripheral Block.



Drag and Drop the method you want straight into place in the Flowchart Editor.

You can also rename Peripherals once added. This is particularly useful for example, if you have several buttons or LED's where it is more transparent to name them according to their job. Simply right click on the peripheral and select the rename option from the resulting context menu.

See Also:

- Visual Designer Hardware

3) Flowchart Blocks

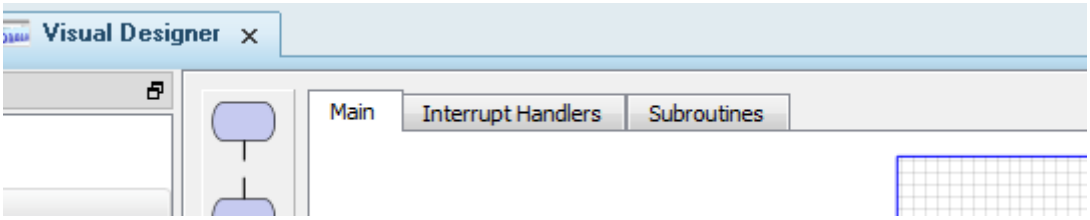
The Flowchart Blocks are the building blocks of your program. While you will often drag and drop straight from the Project Tree you can also drag and drop flowchart objects from this column onto the Editing Window. Indeed, some object types such as Delay Blocks, Loop Constructs and Event Handlers are only available from this column.

See Also:

- Flowchart Blocks

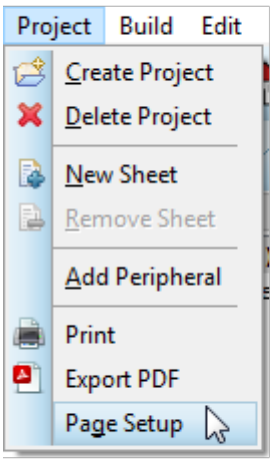
4) Editing Window

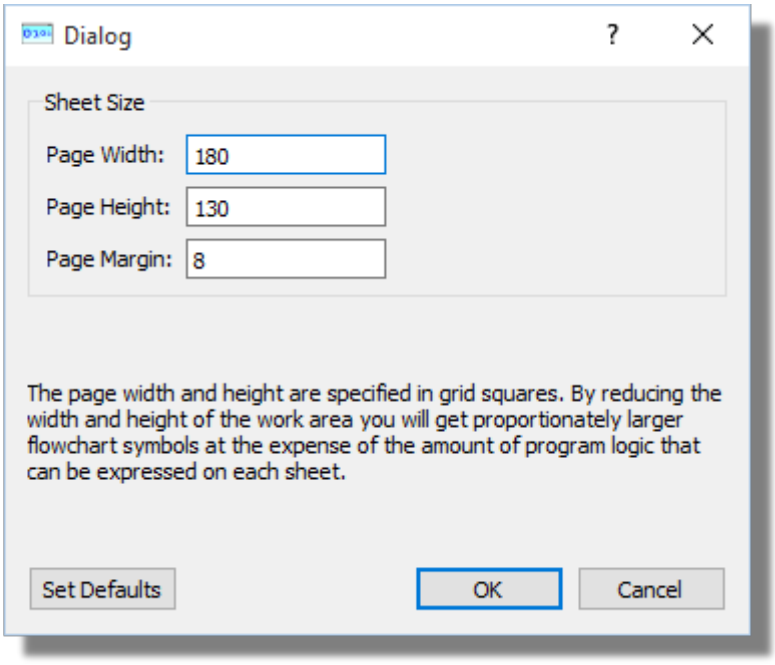
The Editing Window is the place where you actual design your flowchart and create your programs. For programs with multiple sheets the Editing Window displays the current sheet and provides a small tabstrip at the top for easy navigation to other sheets.



Three sheets in a Visual Designer Program with the Editing Window showing the Main Sheet.

The sheet size you see is simple a set number of grid squares wide and high. Flowchart blocks and other object are then scaled to also be a set number of grid squares. You can, from the Page Setup command on the Project Menu, adjust the number of visible grid squares. This allows you to change the relative size of the flowchart blocks on the screen; for example, if you want the blocks to be bigger you reduce the number of grid squares on the screen.





When you print to physical paper the chart will automatically fit to page.

You can place flowchart objects on the editing window by drag and drop from either the Project Tree or from the Flowchart bar on the left hand size.

You can zoom in and out the Editing Window via either the middle mouse wheel, the F6 and F7 keys or the icons on the toolbar.

See Also:

- Basic Skills : Navigation.

Basic Skills: Placement

5) Output Window

The output window provides a dump of status information and will list any errors when you build your flowchart or program the physical hardware.

> If you close the output window the Editing Window will be larger and you will 'zoom in' a little more on the flowchart. The output window will come back automatically when it is needed (i.e. when you build your program)

6) Simulation Control Panel

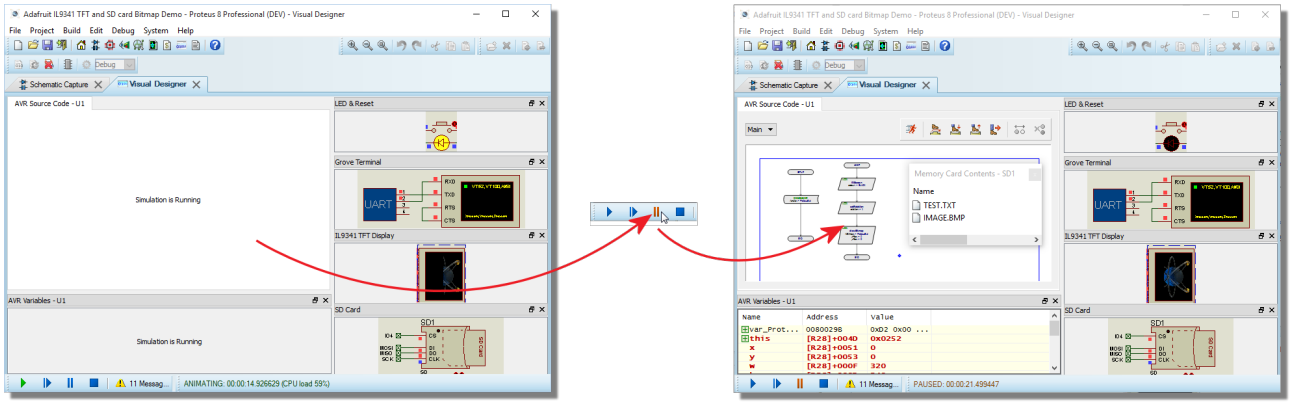
Interactive simulations are controlled from a simple panel that behaves just like a normal remote control. This control is situated at the bottom left of the screen by default. There are four buttons that you use to control your simulation.



- The PLAY button is used to start the simulator.
- The STEP button allows you to step through the animation at a defined rate. If the button is pressed and released then the simulation advances by one time step; if the button is held down then the animation advances continuously until the button is released. The single step time increment may be adjusted from the Animated Circuit Configuration dialog box on the System Menu.
- The step time capability is useful for monitoring the circuits more closely and seeing in slow motion what is affecting what.
- The PAUSE button suspends the animation which can then be resumed either by clicking the PAUSE button again, or single stepped by pressing the STEP button. The simulator will also enter the paused state if an breakpoint is encountered.
- The STOP button tells the system to stop doing a real time simulation. All animation is stopped and the simulator is unloaded from memory. All the indicators are reset to their inactive states but the actuators (switches etc.) retain their existing settings.

7) Source Code Window

The source code window is the primary tool for debugging your designs in Proteus and allows you to single step your program, advancing the entire embedded system as you debug your software. When the simulation is paused the source window will display the flowchart with the currently executing block highlighted in red.



Simulation running and then displaying the program for single step debug when paused.

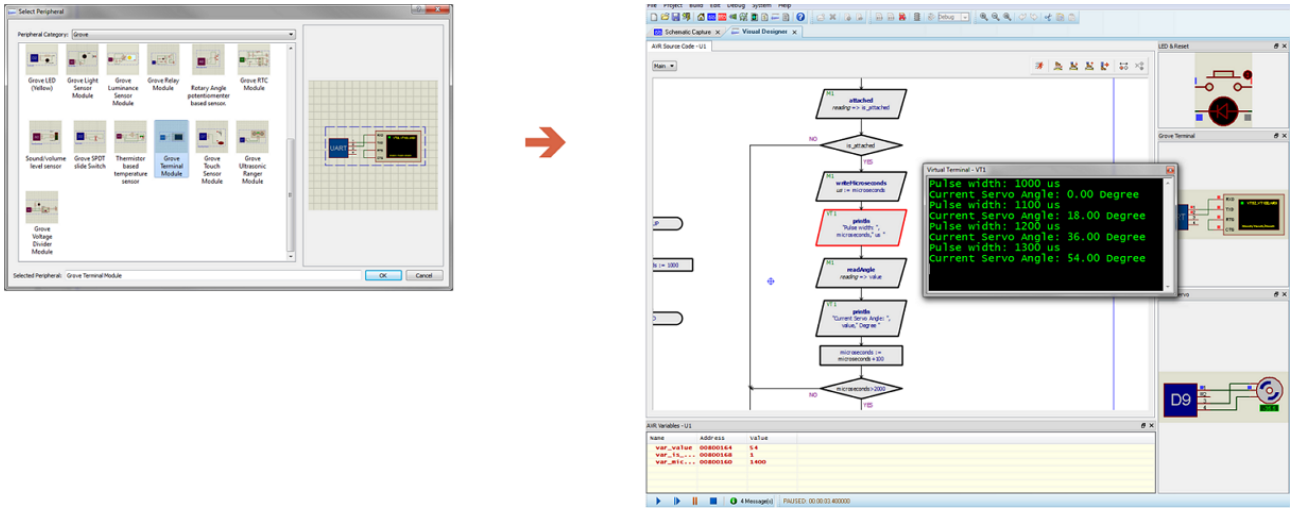
You can even choose to debug the source code rather than the flowchart by first selecting the Debug Generated Code option from the project tree context menu.

See Also:

Basic Skills: Simulating and Debugging

8) Peripheral Popup Windows

Some peripherals present a popup window during simulation. For example, if you add the Grove Terminal Module then you will see a virtual terminal popup during simulation that you can use to read and write text.



Virtual Terminal acting as a console with info dump on servo motor position.

Peripheral popups are typically visible during running simulation so that you can interact with them. They can be closed at any time from the cross at top right and then re-opened from the Debug menu when the simulation is paused.

9) Active Popups

Active popups are simply areas of interest on the schematic that appear as docked widgets on the right hand side of the of the Visual Designer debugging layout. They have two main roles :

1) You can see relevant parts of the hardware while debugging the software (E.g. text appearing on an LCD).

2) You can interact with relevant parts of the hardware while debugging the software (E.g. press a button or adjust a sensor).

The end result of using active popups is that you don't have to constantly switch tabs between the schematic and the firmware in a debug session. They allow all relevant information to be displayed on a single tab.

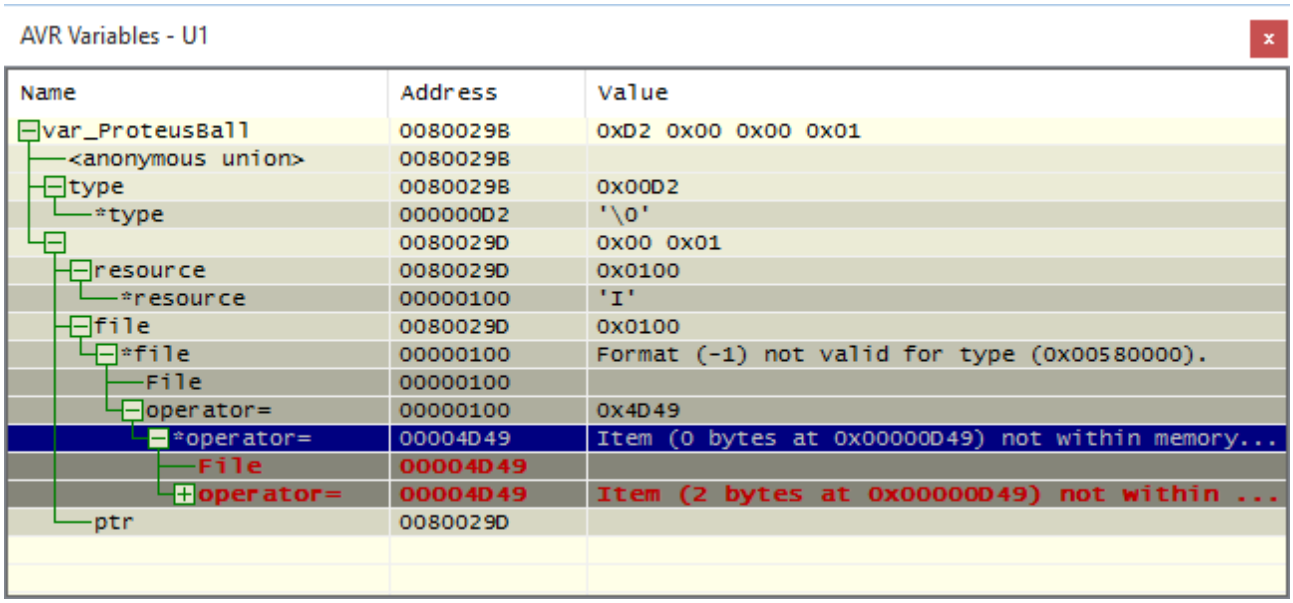
> You can, if you prefer, work with two tabs and switch between schematic and Visual Designer as required. You can even drag and drop into space if you are at a workstation with multiple monitors.

10) Variables Window

The variables window is a debugging tool that lists program variables during a debug session. It includes several powerful features to help you when debugging.

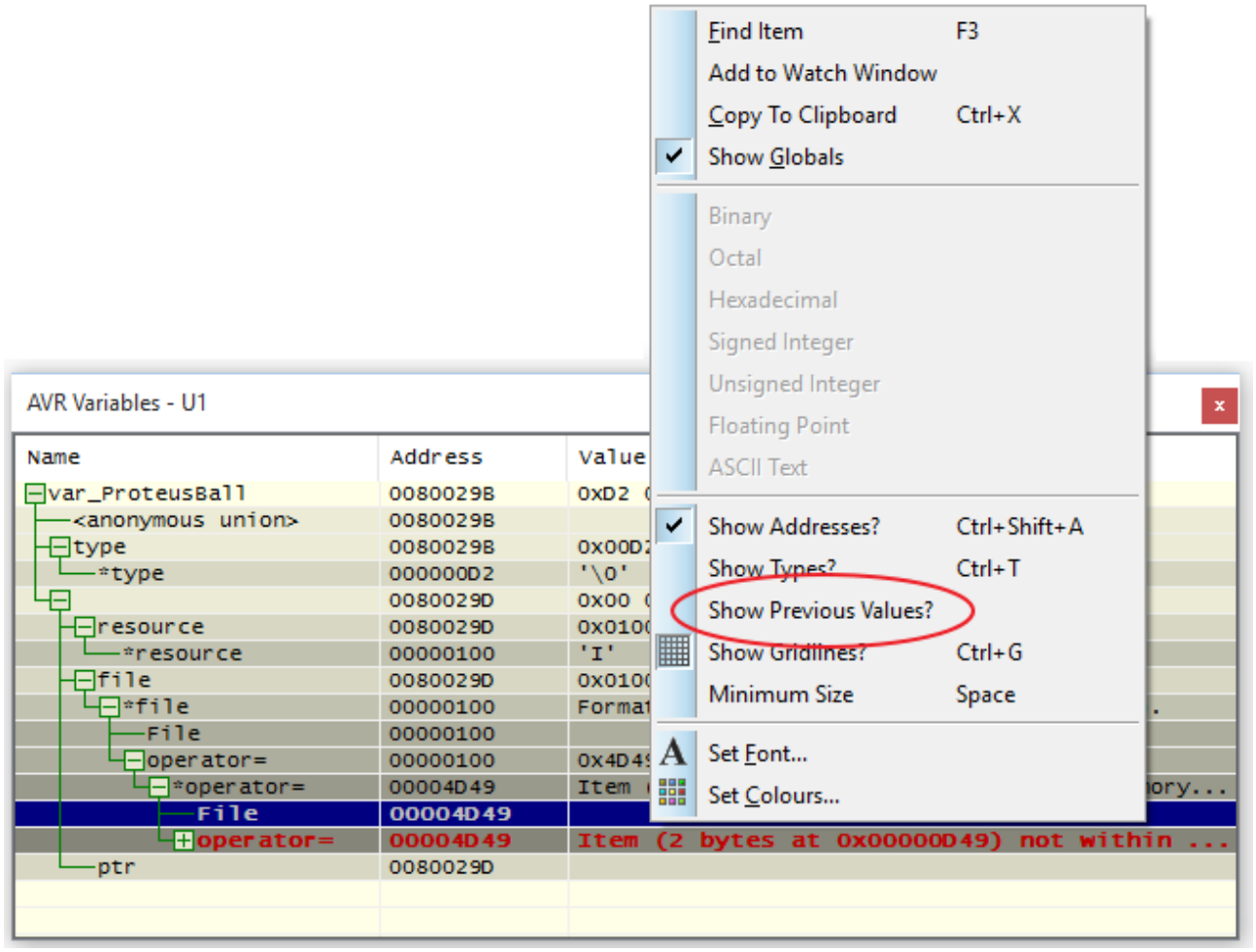
Expansion of compound types

The variables window will seamlessly handle compound types (structs, arrays, enums) and pointers, providing an expandable tree displaying the contents of the type or the de-referencing of the pointer.



Change Notification and Previous Values

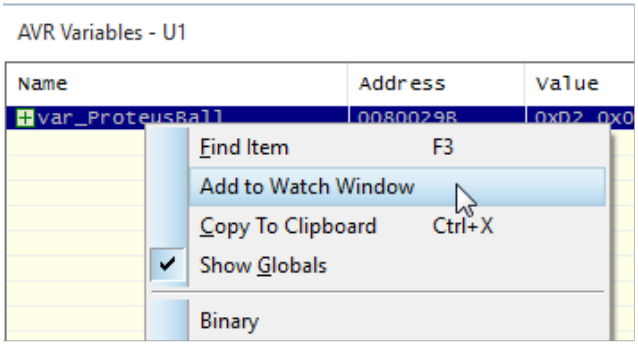
A variable in the variables window will highlight when the value of that variable changes and the simulation is paused. You can also view the previous value of the variable by selecting the Show Previous Values option from the context menu (right click on the variables window).

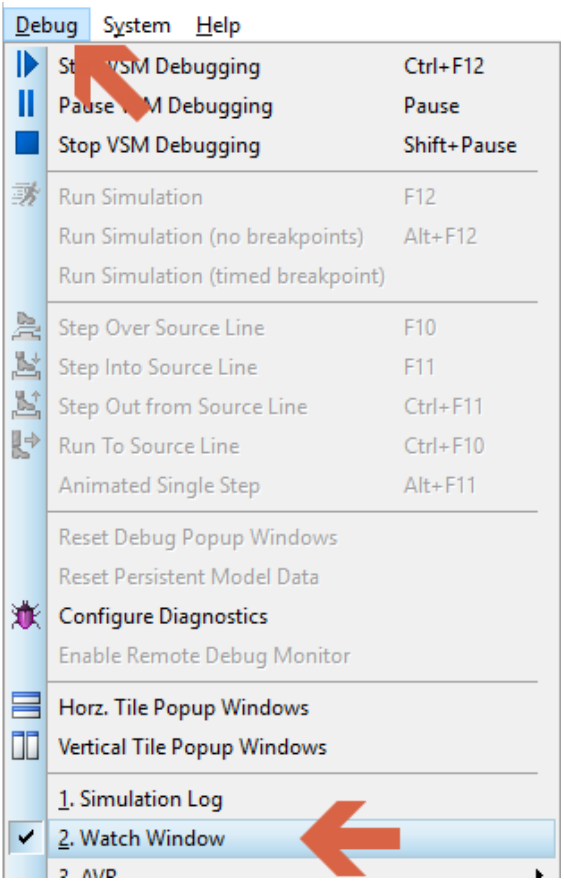


If you have a compound type (such as a struct) and an element of the type changes then only that element will highlight. This means that, unless the type is expanded, you will not see a change in the variables window at the point the element changes.

Add to Watch Window

The variables window is not visible during free running simulation but it's companion the Watch Window is. You can add a variable to the watch window by the right click context menu and you can then open the watch window from the Debug Menu.





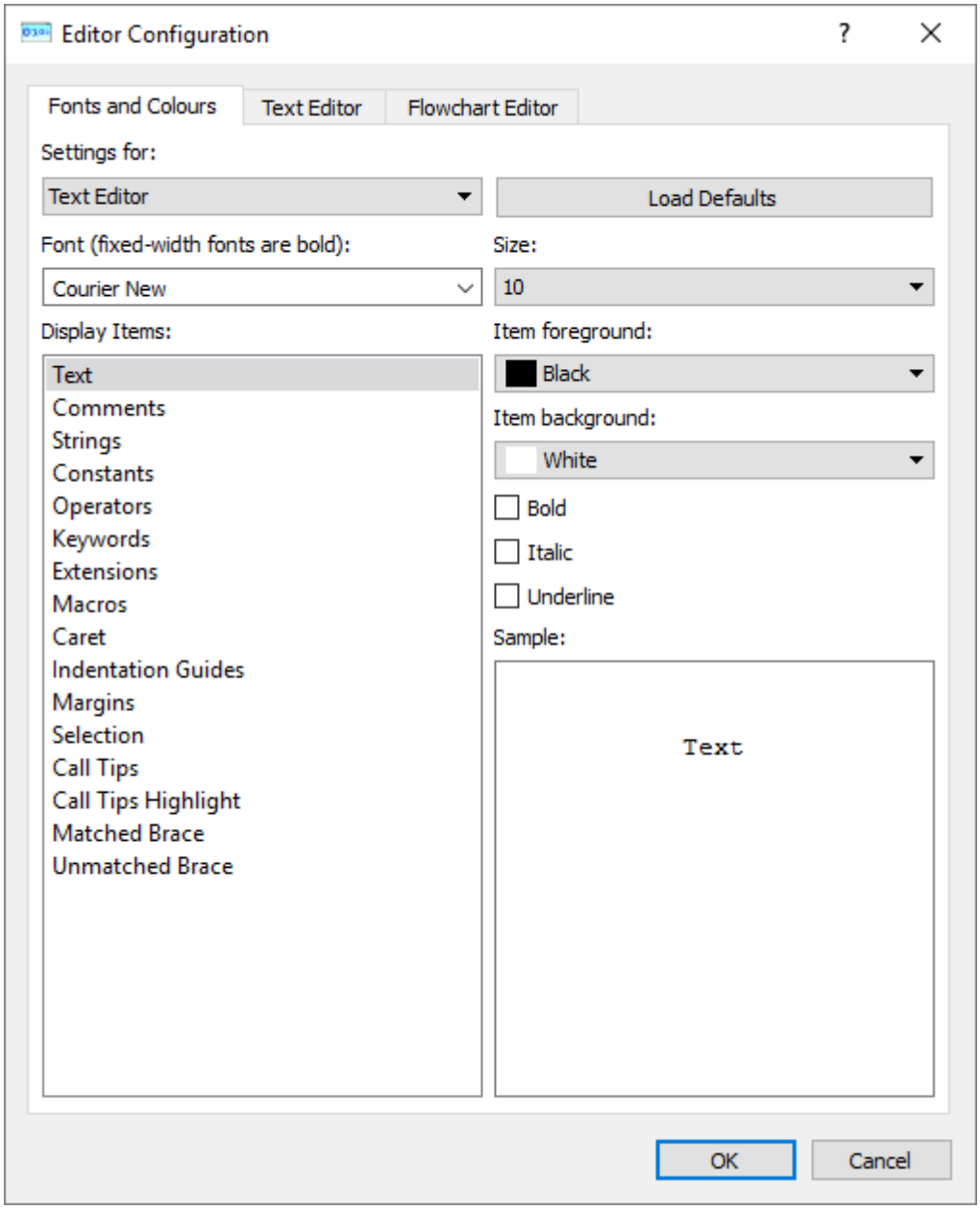
If you drag a variable to the watch window and you then recompile the program and re-simulate then there is no guarantee that you will be viewing the contents of the variable in the watch window – this depends on how the compiler re-uses memory. The watch window simply monitors an address in memory. Be careful.

In addition to the Variables and Watch Windows there are many more available debug windows which provide more detailed information about the MCU. These can all be launched from the Debug Menu when the simulation is paused and are documented in full in the Proteus VSM help files.

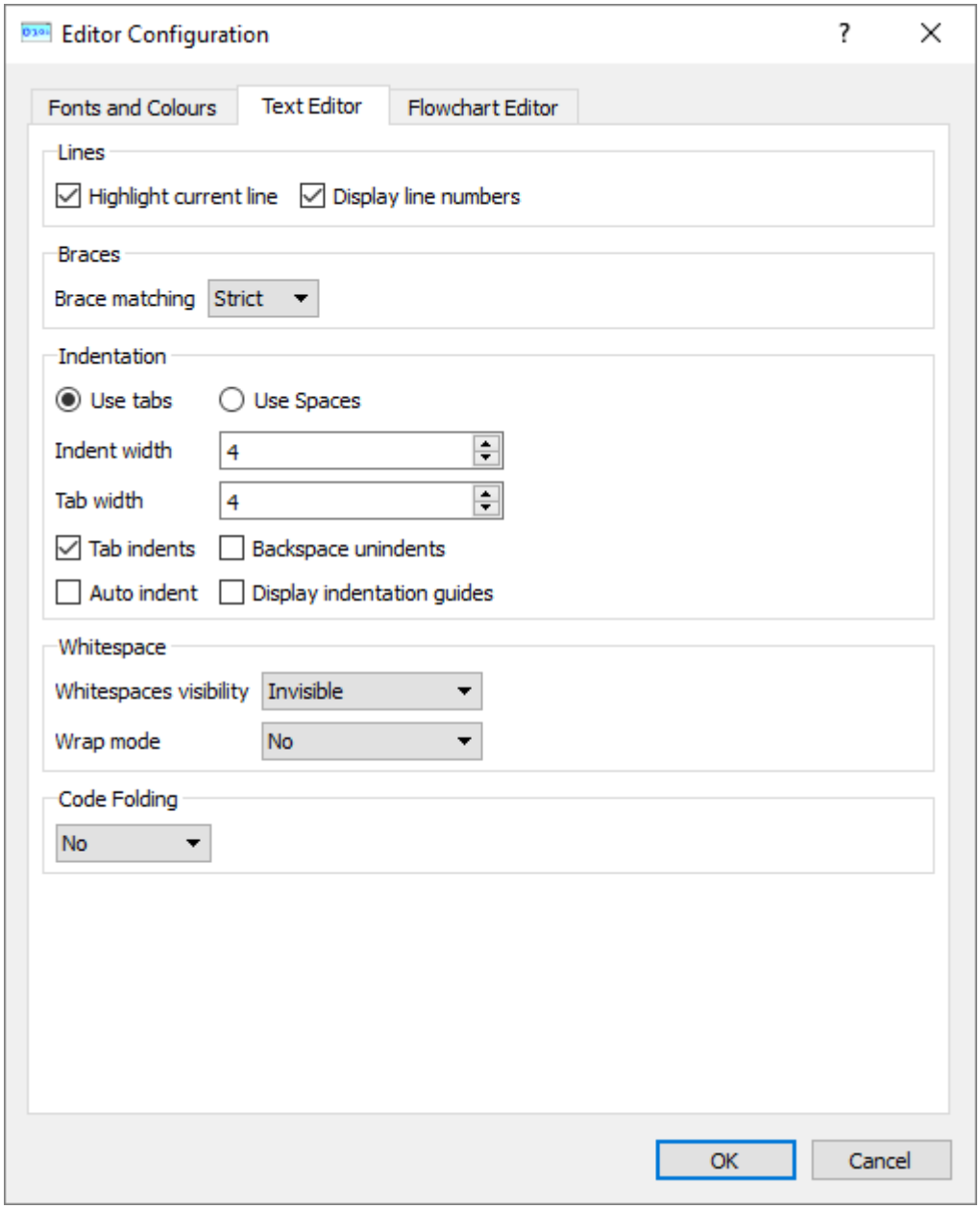
Colours, Fonts and Flowcart Editor

In Visual Designer you can edit the fonts, colours and the style of the Flowchart editor to your own preference. To open the dialog go to the System Menu > Editor Configuration

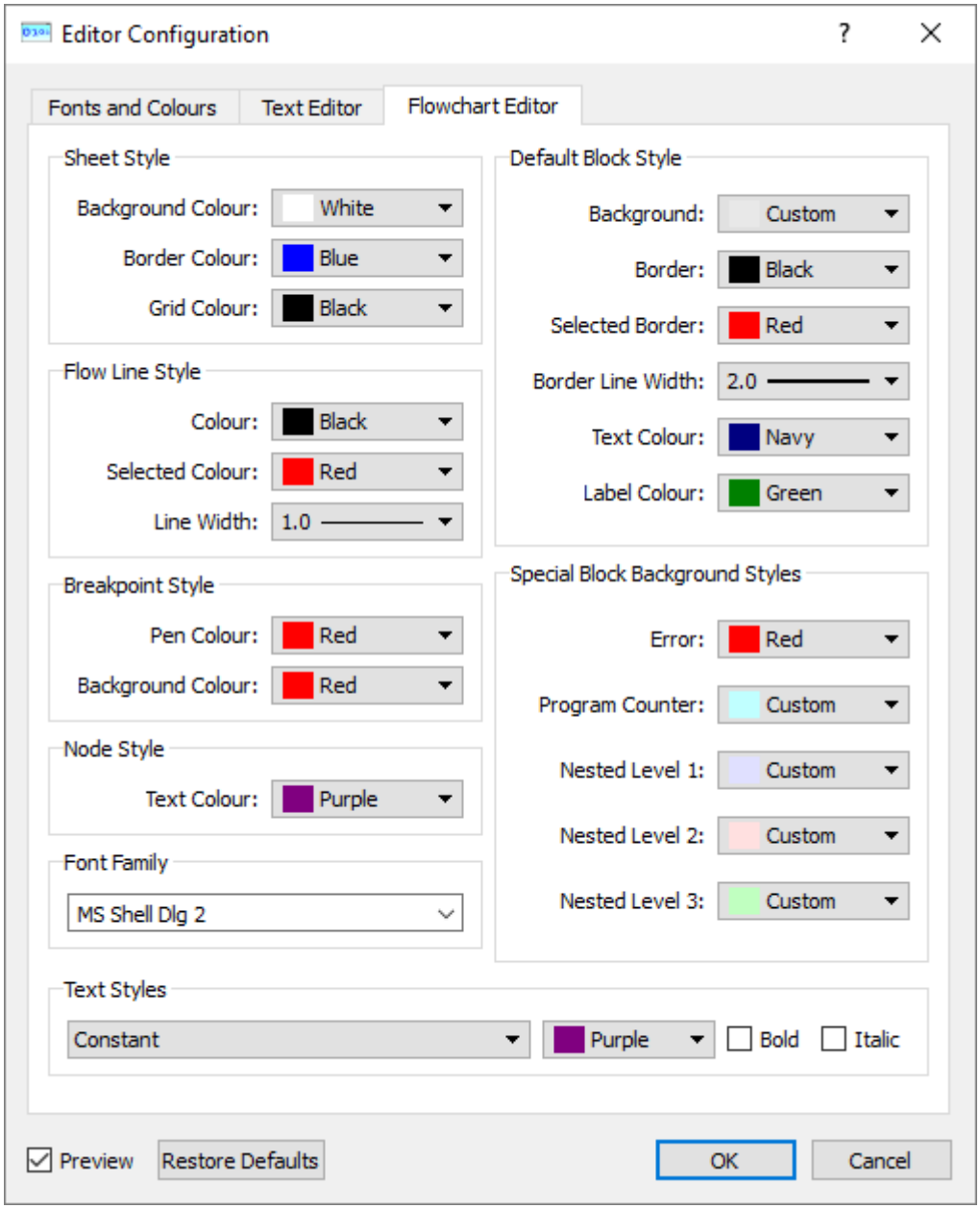
Fonts and Colours



Text Editor


Editor Configuration
Fonts and Colours
Text Editor
Flowchart Editor
Lines
Highlight current line
Display line numbers
Braces
Brace matching
Strict
Indentation
Use tabs
Use Spaces
Indent width
4
Tab width
4
Tab indents
Backspace unindents
Auto indent
Display indentation guides
Whitespace
Whitespaces visibility
Invisible
Wrap mode
No
Code Folding
No
OK
Cancel

Flowchart Editor



EDITING SKILLS

Overview

This topic explains how to interact with the flowchart editor. The focus here is on learning the skills needed to drive the software and create a program; simulation and debugging are covered separately and we will link to more detailed topics where appropriate.

Navigation

To zoom in and out on the Editing Window

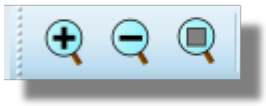
You can either:

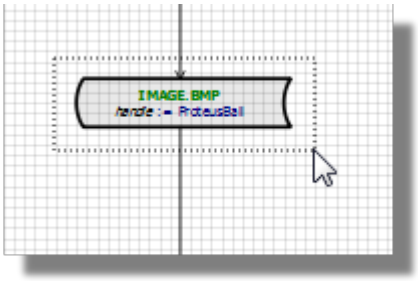
1) Roll the middle mouse wheel in and out. This will zoom the Editing Window around the mouse position.

2) Use the zoom icons. This will zoom around the center of the Editing Window.

3) Press the F6 key to zoom in and the F7 key to zoom out. This will zoom the Editing Window around the current mouse position. The F8 key will reset the zoom levels to show the entire sheet.

4) Hold SHIFT key and left drag a box with the mouse. When you release, the Editing Window will zoom to the area drawn out by the box.





To pan across the sheet in the Editing Window

You can either :

1) Roll the middle mouse wheel backwards (Zoom out), move the mouse to desired place and roll the middle mouse wheel forwards to zoom in.

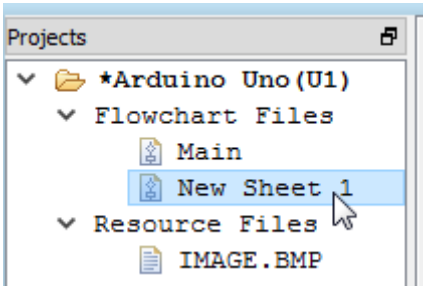
2) Hold the SHIFT key down and bump the mouse against the side of the Editing Window. This will pan the sheet across the Editing Window.

To switch between sheets in the Editing Window

You can either:

1) Click on the sheet that you want on the small tabholder directly above the main Editing Window display.

2) Double click on the sheet that you want in the project tree.

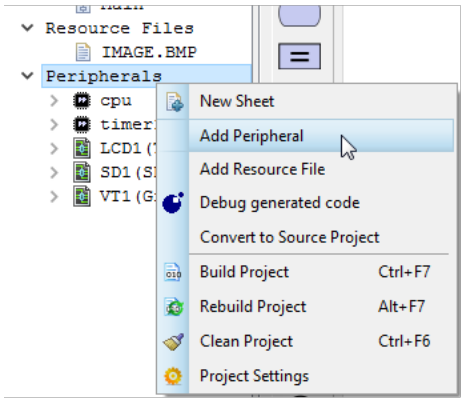


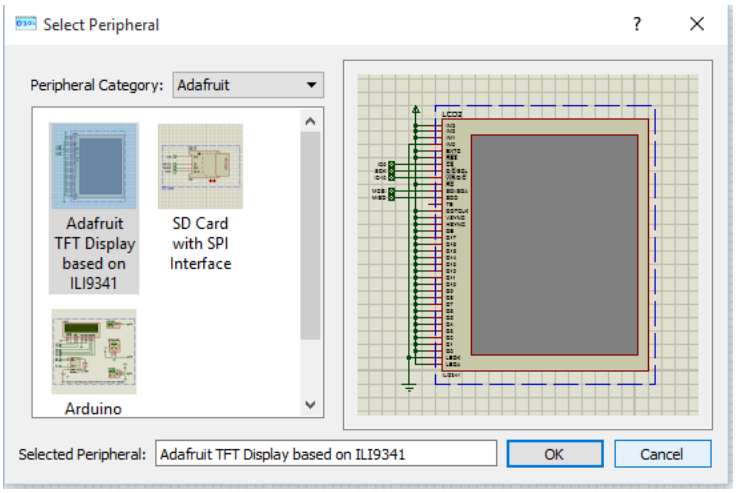
Placing, Selecting and Deleting Blocks

To place a flowchart block

Peripheral Operation Blocks:

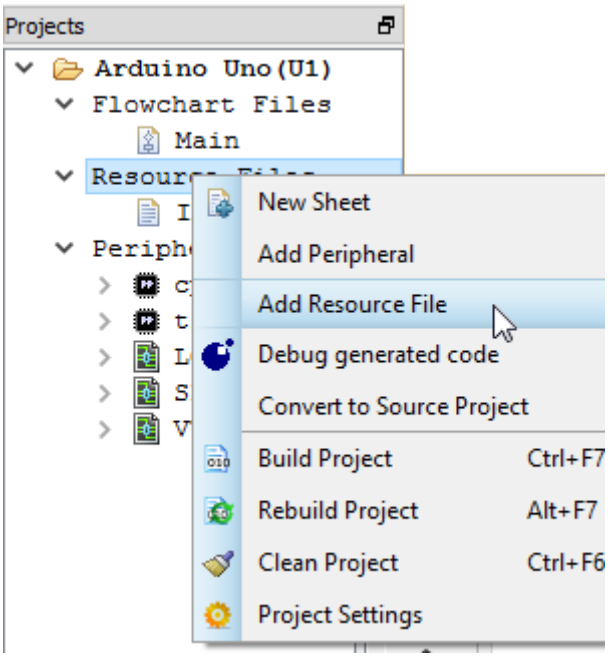
Add the peripheral via the right click context menu on the Project Tree and then drag and drop the method onto the Editing Window.





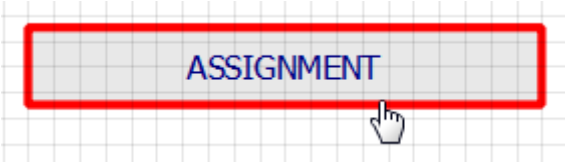
Storage Block Types

Add the resource via the right click context menu on the Project Tree and then drag and drop onto the Editing Window



Any block type:

Drag and drop from the appropriate flowchart icon across into the Editing Window.



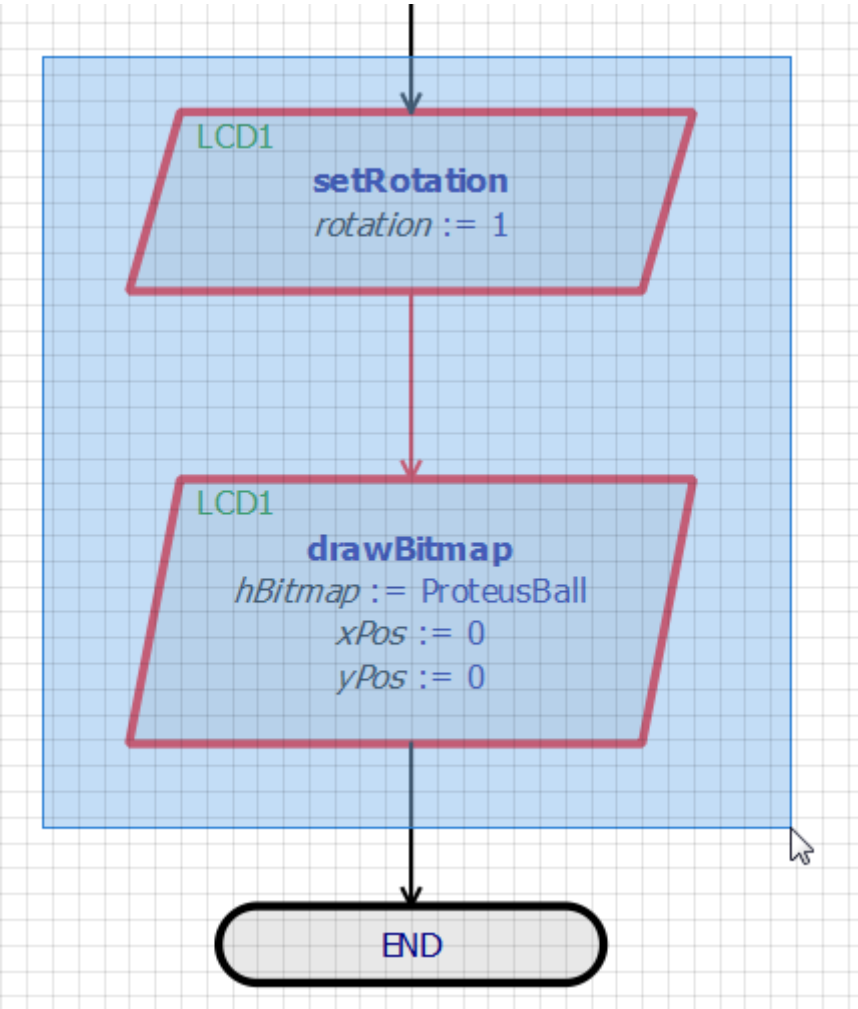
To select a flowchart block

Left click the mouse on the block.

To select multiple flowchart blocks

You can either:

1) Drag a box with the left mouse button over the blocks you want to select.



2) Hold SHIFT key down and left click on blocks individually.

> Note that many actions are specific to a single flowchart object (e.g Edit). So, while you can for example move selected objects as a group, you will be prevented from performing other block operations.

To delete a flowchart block

You can either:

1) Right click on the block and select delete from the resulting context menu.

2) Left click to select the flowchart block and press the delete key on the keyboard.

To delete multiple flowchart blocks

Select the blocks to delete and then press the delete button on the keyboard to remove them.

Attaching, Moving and Detaching Blocks

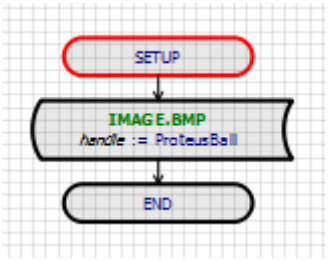
To move an unconnected block

Select the block by left clicking and then drag the block.

To move a connected block

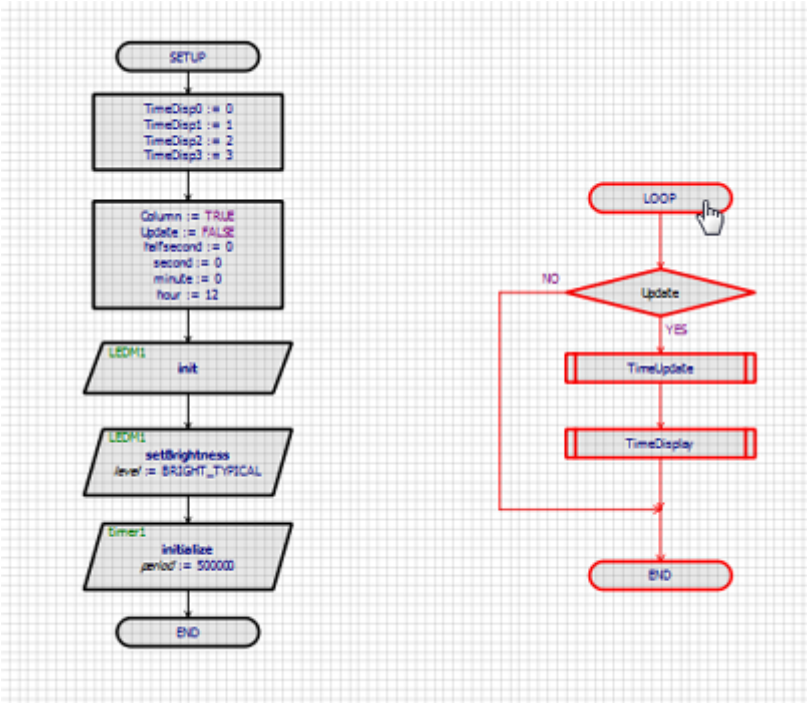
Select and then drag the block. This will bump the adjacent block in the direction of motion and result in a concertina affect. This is an important technique because it is the method you use to make space on a flowchart for inserting a block.

> You can quickly 'tidy' a flowchart vertically by moving the top or bottom blocks to make the spacing between all of the blocks the same.



To move a series of connected blocks

If the set of blocks you want to move includes both a start and end block then you can simply drag a box around it and move it to a better place on the sheet.



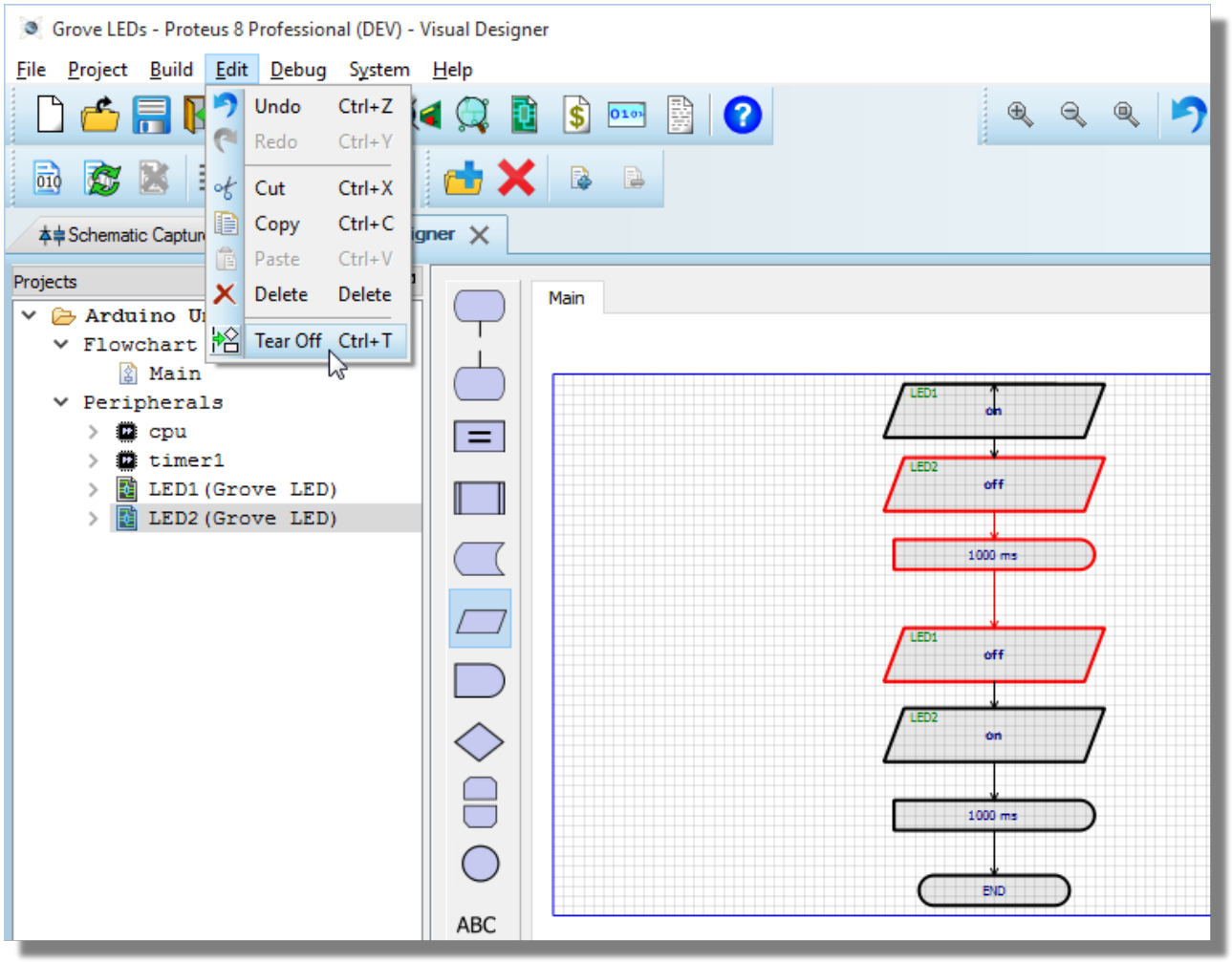
Block move of an entire routine

If you want to move the routine onto a different sheet you can use the clipboard cut and paste across the two sheets.



If you want to move a series of connected blocks that form part of a routine, but not the full routine, then you must either:

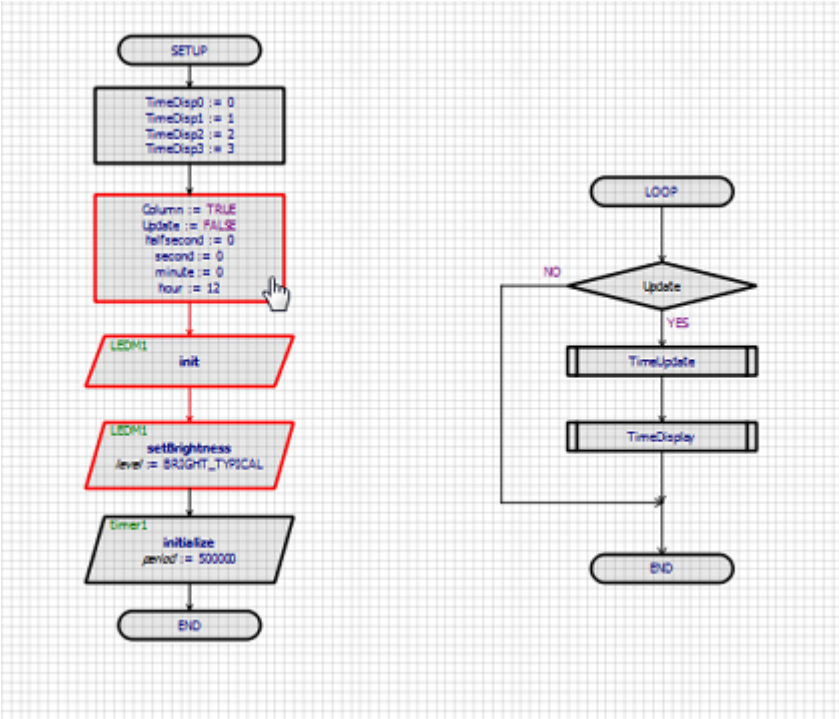
1) Tear off the series of blocks and move them elsewhere.



2) Move them in the context of the current routine.

- If you move vertically this will bump other blocks closer together but it will not allow you to change the position of the selected blocks in the flowchart.
- If you move horizontally it will move the entire routine horizontally.

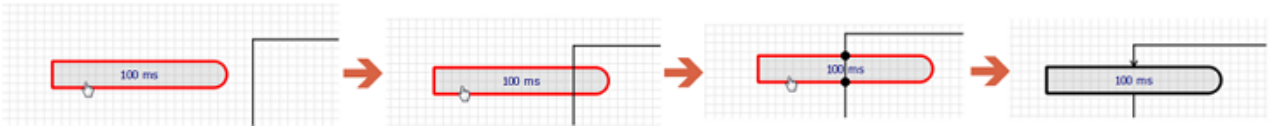
Also, it can lead to some very strange looking charts !



Block Move of part of a routine.

[To attach a block to a flowchart]

When you want to insert a block you do so by dragging and dropping. As you drag the flowchart block you will see connection nodes appear when the block is correctly positioned over the wire. Releasing at this point will connect the block to the flowchart.

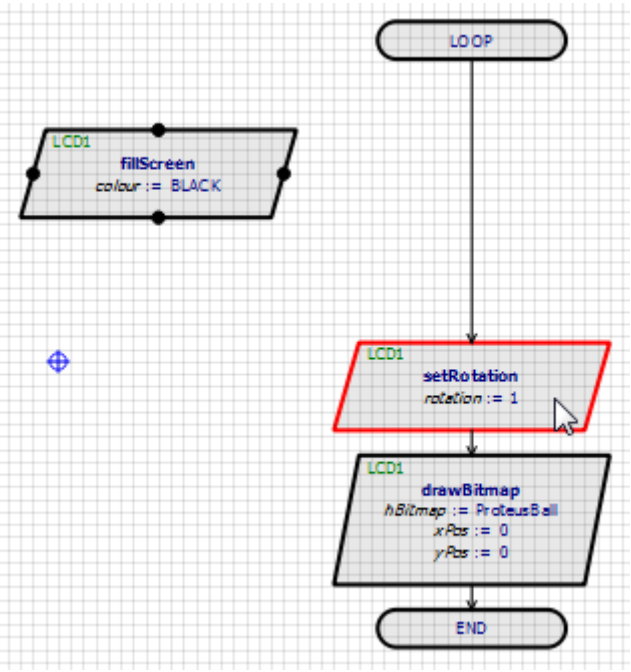


Connection Nodes tell you when to release the mouse - this will auto-connect the block to the chart

> You can do this as you initially place the block or by selecting a block that is placed but unattached.

> Note that some block can only be connected at top and bottom (e.g. delay block) while others also allow sideways entry (e.g. assignment block).

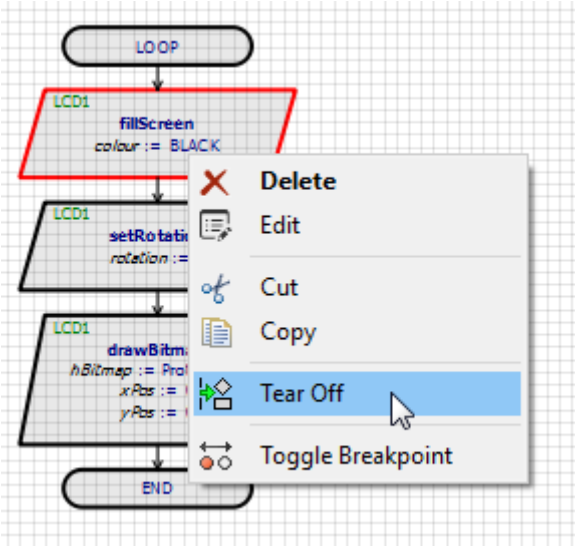
If you are inserting a block you may need to first make some space. You can do this easily by dragging one of the blocks in the flowchart up or down to clear an area for the new block to go.



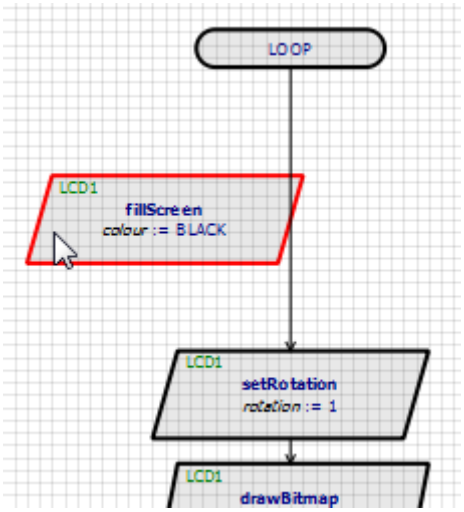
To detach a block from a flowchart

You can detach or tear off a block from a flowchart by either:

1) Right click on the block and select tear off from the resulting context menu. This will detach the block and leave it slightly to the side for re-positioning.



2) Select the block and hold the CTRL button down as you move the block away. This will tear the block from the current chart and you can then re-position as required.



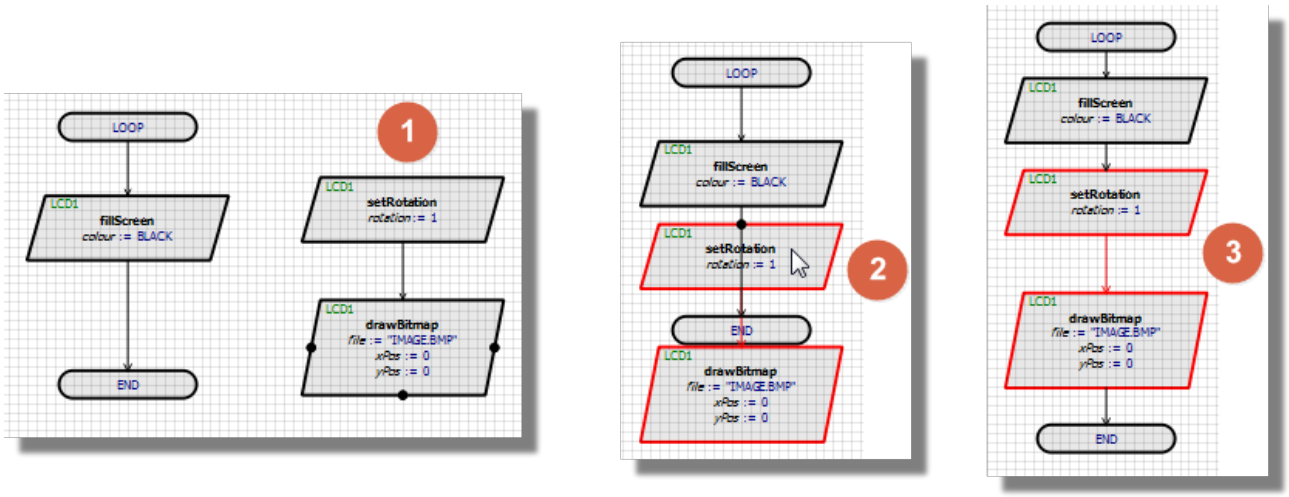
To attach a series of flowchart blocks

1) Make some room at the place in the chart you want to insert.

2) Select the series of 'connected' blocks you want to insert.

3) Move them into place, watching the top connection nodes of the uppermost block. When this appears, check that the bottom connection node is also visible.

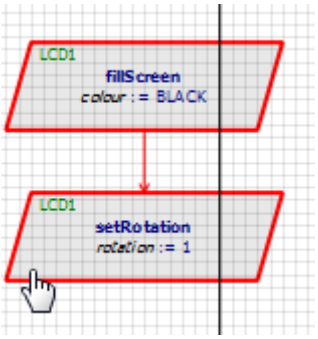
4) Release the mouse to insert.



To detach a series of flowchart blocks

1) Select the series of blocks you want to insert.

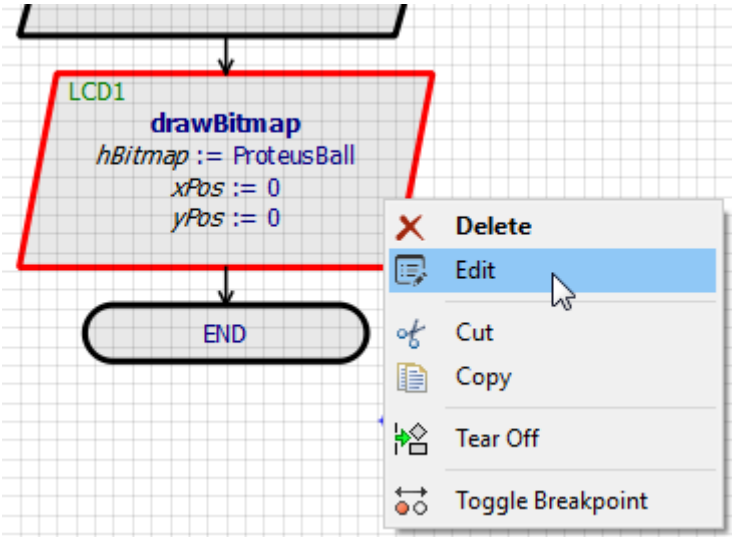
2) Hold the CTRL button down and drag the blocks off the flowchart or use the tear off command on the Edit Menu.



Commands on the context menu apply to a single block whereas commands on the Edit menu apply to all selected blocks. This is an important distinction as some commands (e.g. Edit) are meaningless for multiple blocks.

Editing a Block

To edit a block you can either right click and select Edit from the resulting context menu or you can double click on the block.

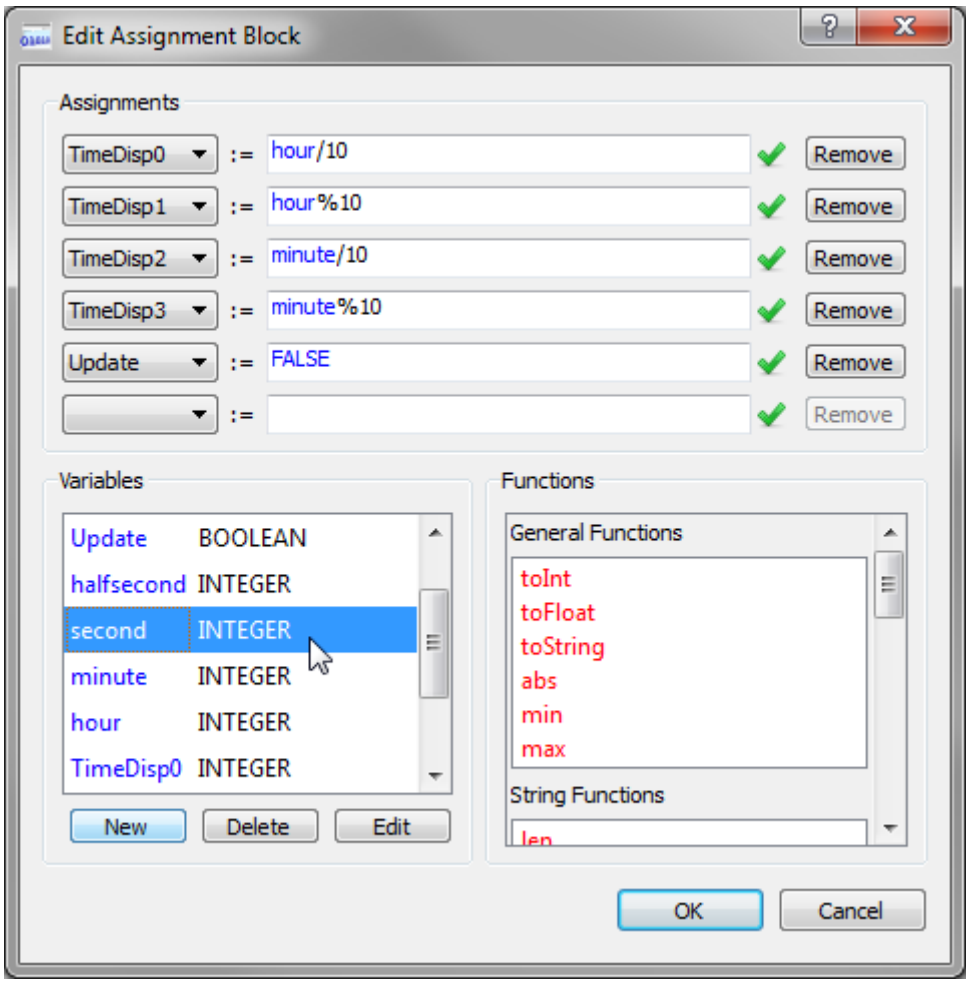


Regardless of the type of block you are editing you are almost certain to want to enter an expression to give the block meaning. In some cases such as calling a subroutine this will be extremely simple but for other block types it can be a little more complex. Fortunately, Visual Designer can help in several different ways:

Variable and Function Listing

Where appropriate, all current variables are displayed in the bottom left of the dialogue form and standard library functions are displayed in the bottom right of the dialogue form. Double clicking on variable/function will add it to the current expression.

You can easily create, edit and remove variables via the buttons at the bottom of the variables display.

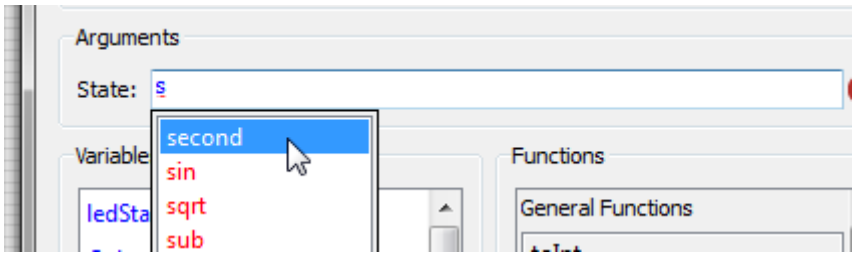


Multiple Expression Assignments using Variables

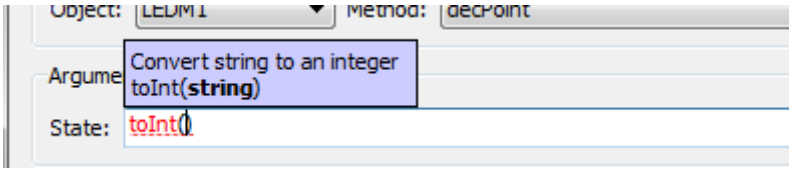
Combo-boxes will be used to restrict selection to variables of the correct type where possible. For example, the loop variable in a For-Next loop must be of type INTEGER.

Auto Complete

The auto-complete system will both prompt you with function hinting during entry and also automatically complete named variables when typing. Press TAB or ENTER to auto-complete.



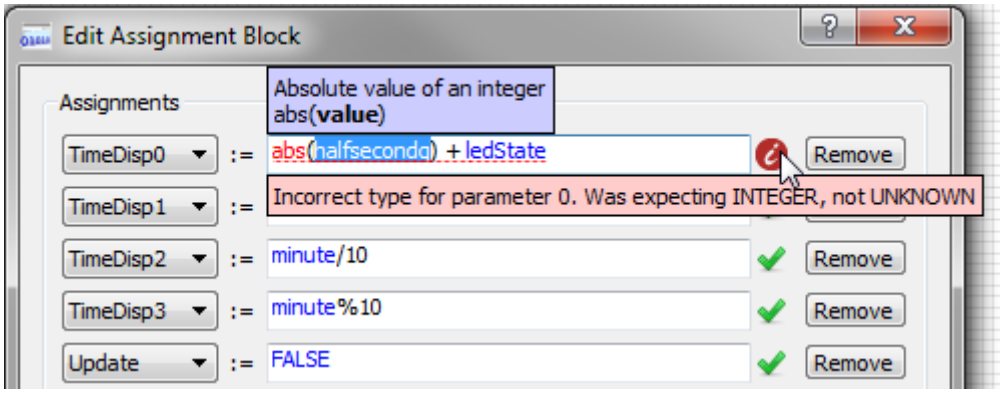
Auto-completion



Function hinting

Syntax and Type Checking

Comprehensive syntax and type checking takes place as you enter the expression. The expression status is indicated at the right hand side and clicking on the red info icon will provide detail on any errors in the expression.



Get more information on incorrect expressions.

Refer to specifics on individual block types for more detailed information.

See Also:

Flowchart Block Types

Wiring blocks together

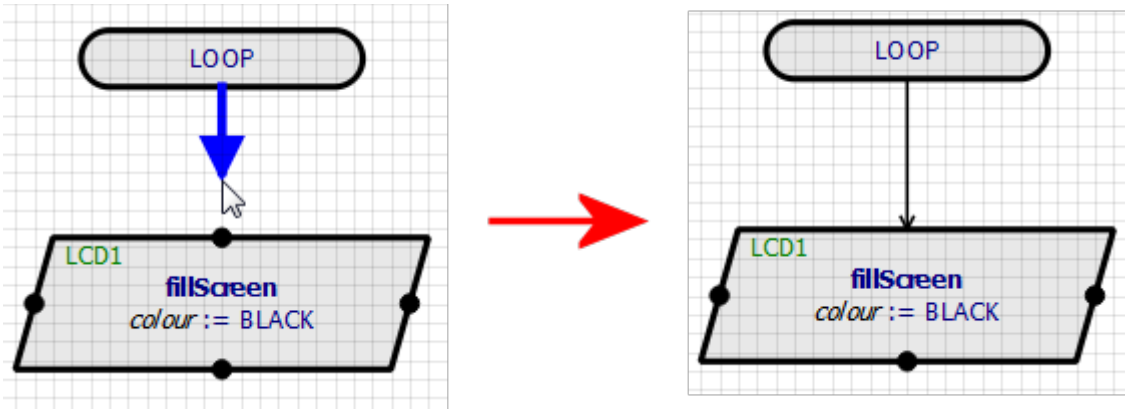
Blocks are connected together with flowlines and Visual Designer provides an point-to-point router to help with the task. The golden rule is that you must connect from an output to an input.

To join blocks together with a flowline

1) Click on the output connection node.

2) Move the mouse to the destination (input) connection node.

3) Click to complete the flowline.



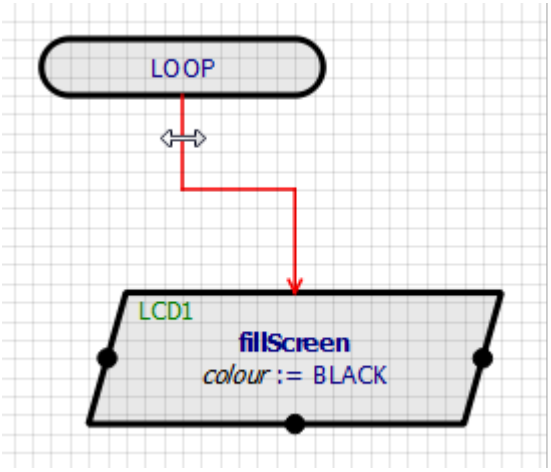
It's good practise to connect flowlines from top downwards to bottom because it makes loops much clearer. Try to avoid connecting from bottom to top in normal flow.

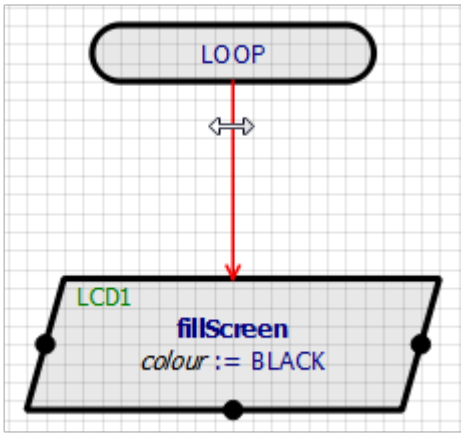
To adjust / move a placed flowline

1) Click to select the flowline.

2) The mouse cursor will change to show you the available direction of movement.

3) Left drag with the mouse to drag.





To delete a flowline

1) Right click on the flowline and select the delete option from the context menu.

Splitting the flowchart into columns

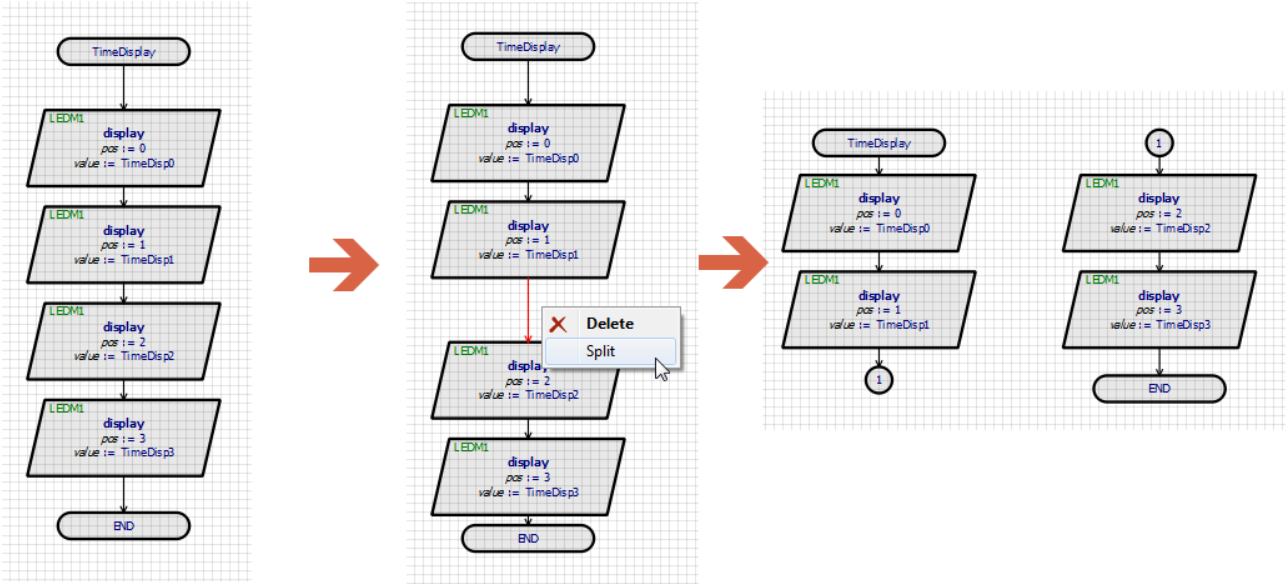
If the flowchart is getting to the bottom of the page inside a single routine you may need to split the flow with interconnect blocks.

1) Right click on the flowline near the bottom of the page and select split from the resulting context menu.

2) Block Move the bottom portion up to the top of the page at the right hand side.

3) Make some space.

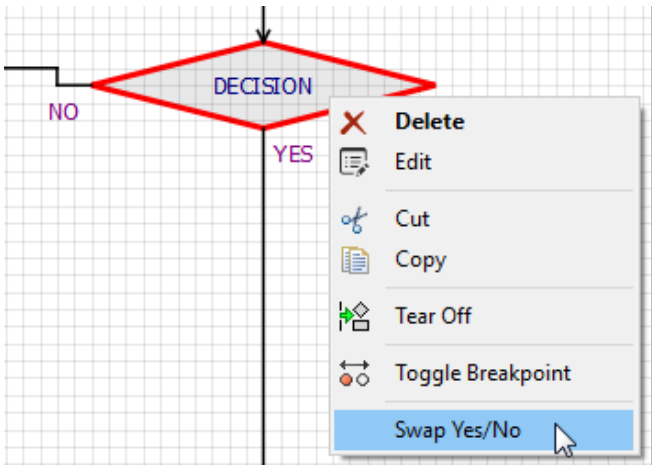
4) Continue inserting blocks.

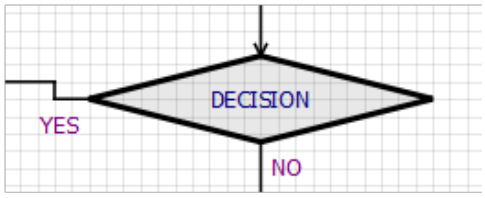


To change the flow of decision blocks

1) Place two flowlines corresponding to the output paths of the decision.

2) If the YES and NO paths are the wrong way around, right click on the decision and swap them.





Clipboard Commands

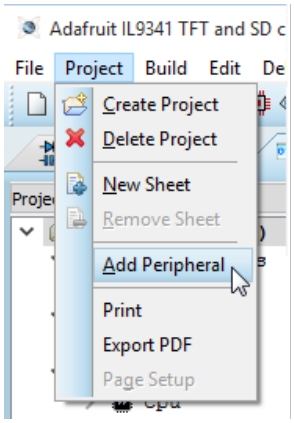
Clipboard commands in Visual Designer work in exactly the same way as other Windows applications and can by invoked in the normal way.

- CTRL+X : Cut to Clipboard.
- CTRL+C : Copy to Clipboard.
- CTRL+V : Paste from Clipboard.
- Menu Icons.
- Commands on Edit Menu.

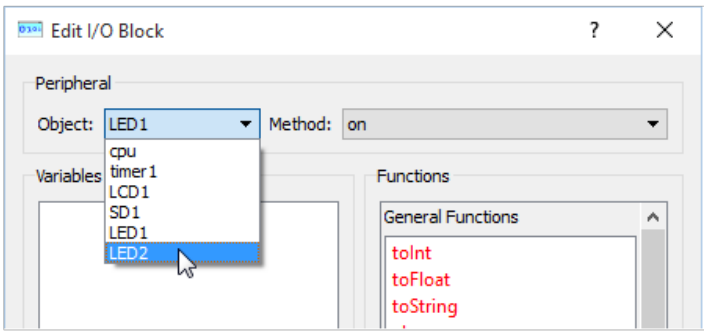


The most common use for the clipboard in Visual Designer is for organizing flowchart routines in the Editing Window and, in particular, for moving routines between sheets.

Remember when copying and pasting that you may be pasting blocks that include peripherals (e.g. hardware shields). In this case the paste will duplicate the flowchart blocks and assignments but will not create new hardware on the schematic. If that is what you want to do, you need to add a peripheral, edit the relevant pasted blocks and assign the method to the correct hardware.







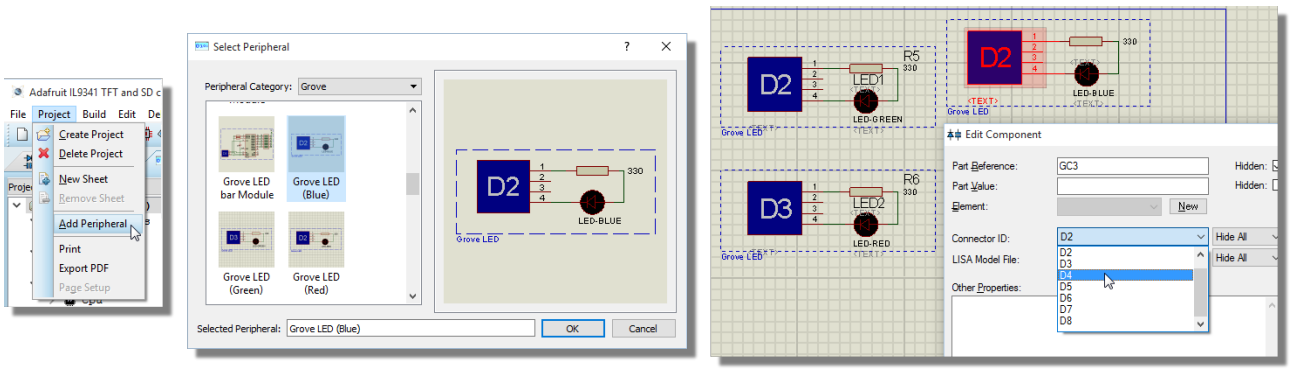
Adding, Using and Removing Peripherals

To add a peripheral device

1) Right click on Project Tree and select Add Peripheral from the resulting context menu.

2) Select the shield or Grove peripheral from the Browser.

3) If you are using Grove peripherals you may need to switch to the schematic module, double click on the label and change the connector to a unique ID.

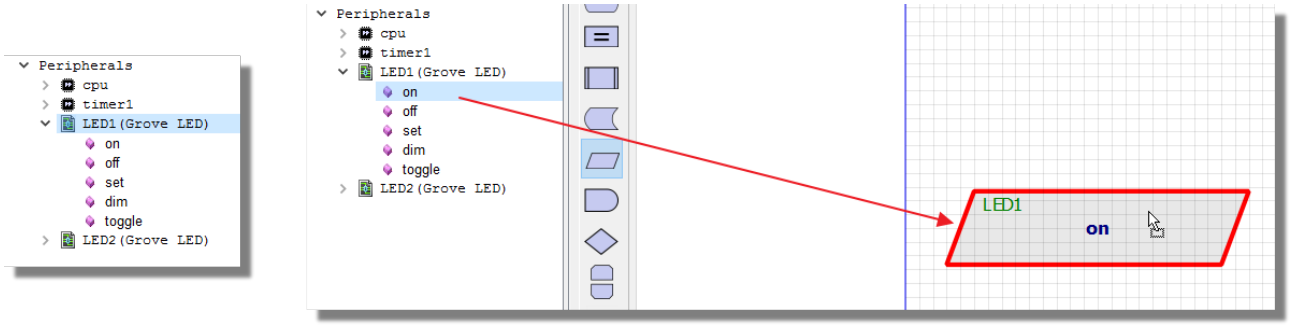


With Arduino, it is quite easy to add shields that are incompatible with each other. For example, if two shields use the same CPU timer, the program is unlikely to work. You need to check this compatibility yourself as Visual Designer does not interfere with the Arduino drivers.

To use a peripheral device

1) Expand the menu in the Project tree to access the methods of the peripheral.

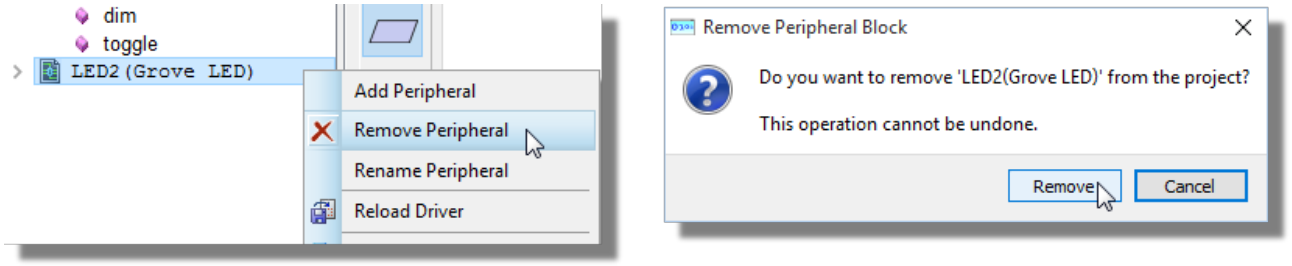
2) Drag and drop the method onto the flowchart.



To remove a peripheral device

1) Right click on the peripheral in the Project tree and select Remove Peripheral from the context menu..

2) Confirm from the resulting dialogue box.



If you remove a peripheral device, it will not remove any flowblocks that referenced the device. You need to adjust or delete all such blocks manually.

See Also:

- Project Tree
- Peripherals and Shields

Adding, Using and Removing Variables

Variables are created and manipulated inside the object editing dialogue forms and are always global to the project. The following types are supported:

- Boolean (TRUE / FALSE)
- Integer (Integer number)

- Float (Floating point number)
- String (String of characters)
- Handle (Handle to a resource).

The expression editor will type-check assignments automatically.

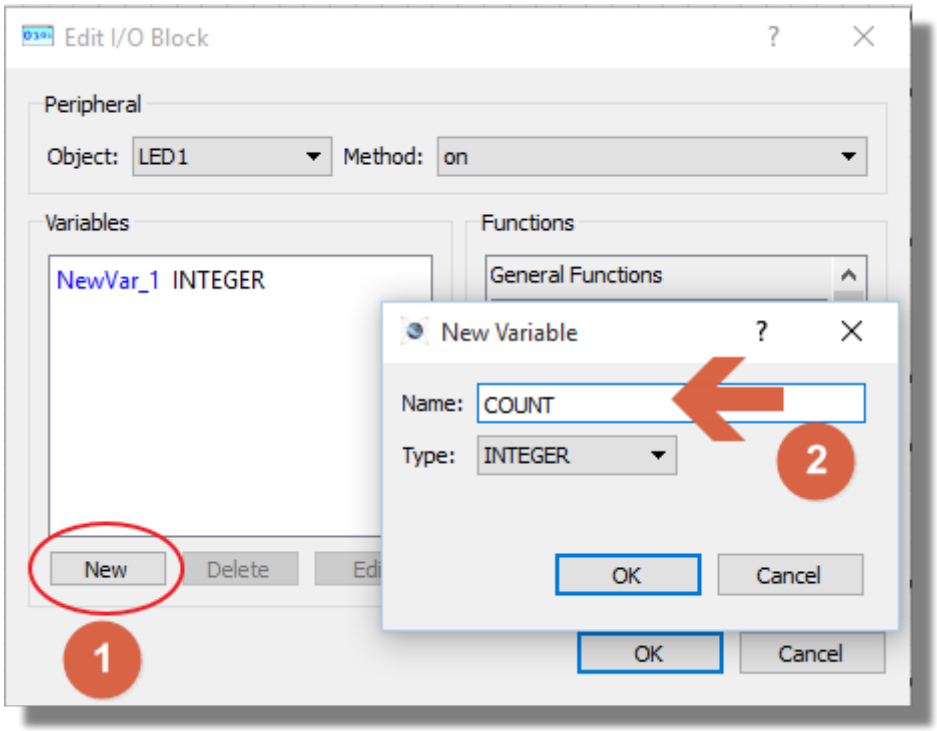
To create a new variable

1) Edit the block you are working on.

2) At the bottom left, use the New button to launch the Create Variable dialogue.

3) Enter the name of the variable and assign it a type.

4) Click OK to exit the dialogue.

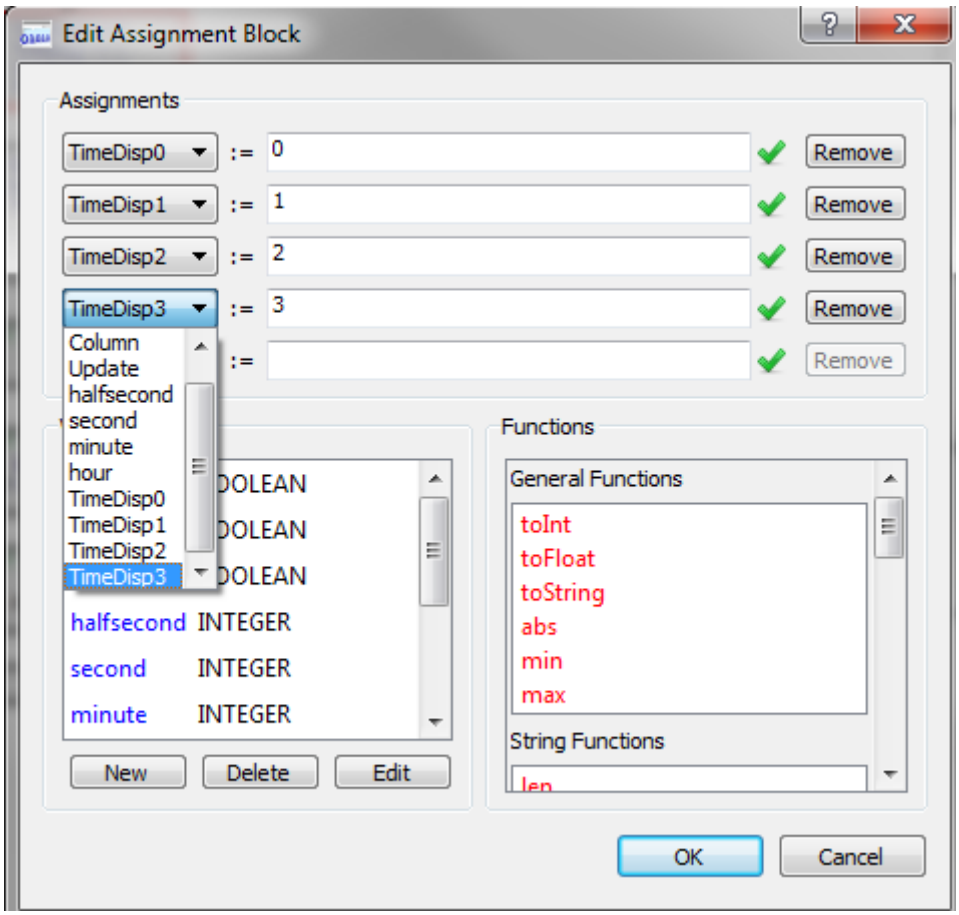


1, Click New. 2, Set the Variable and click OK.

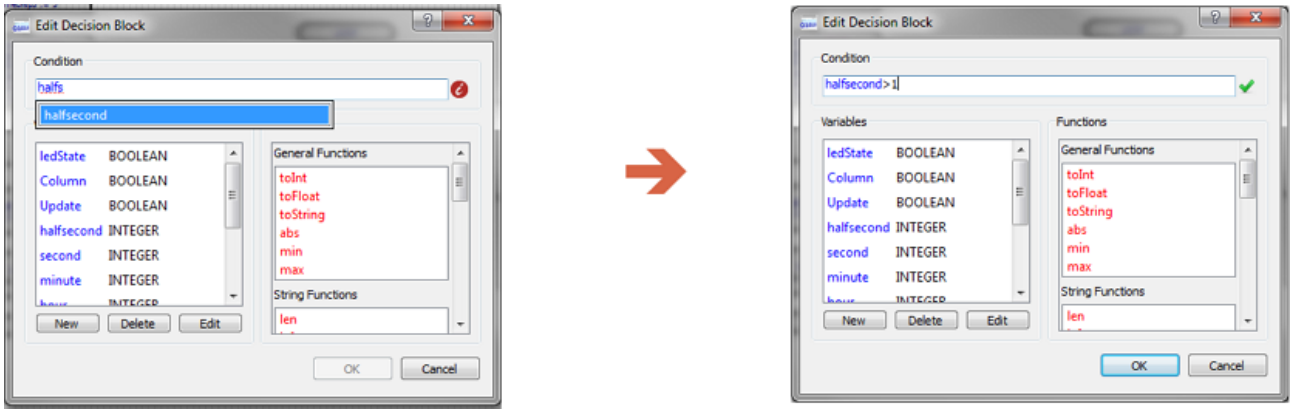
To use an existing variable

Edit the block you are working on and then either:

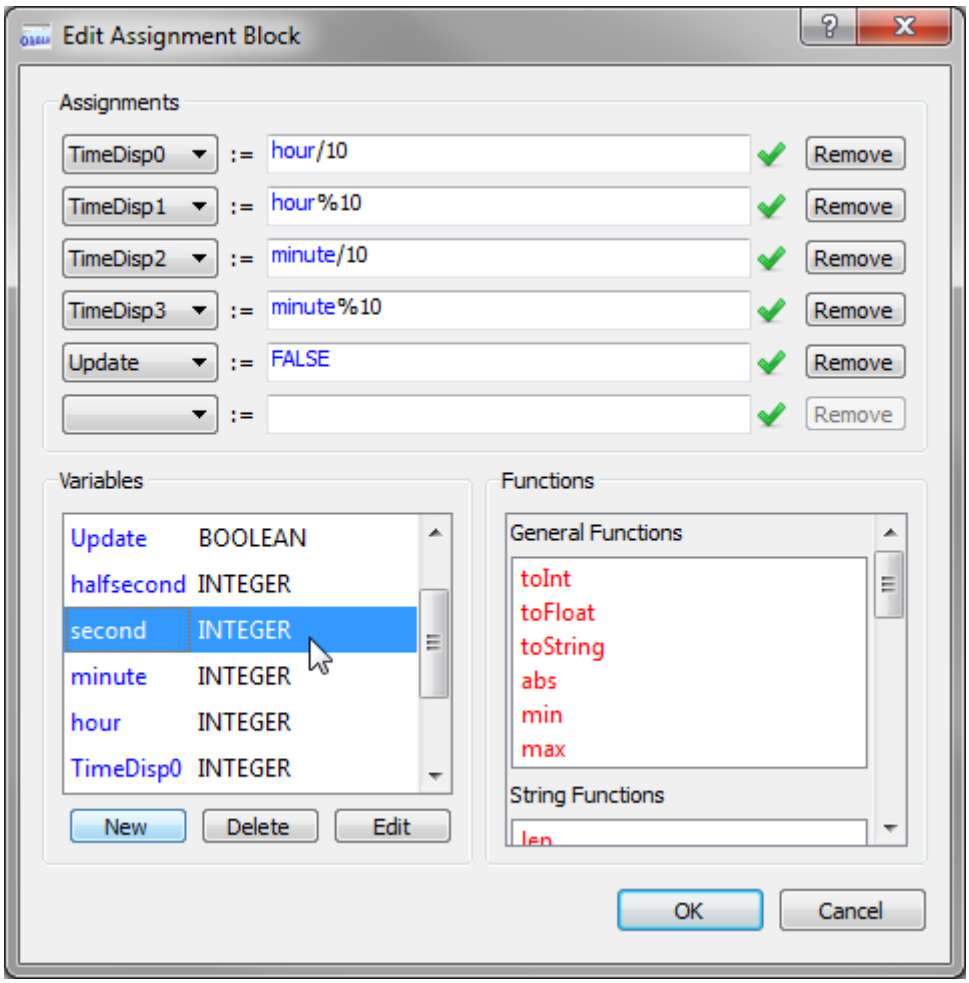
1) Use the variable drop down to select the variable.



2) Type the variable name and use the auto-complete.



3) Double click the variable name from the list to insert into the expression.

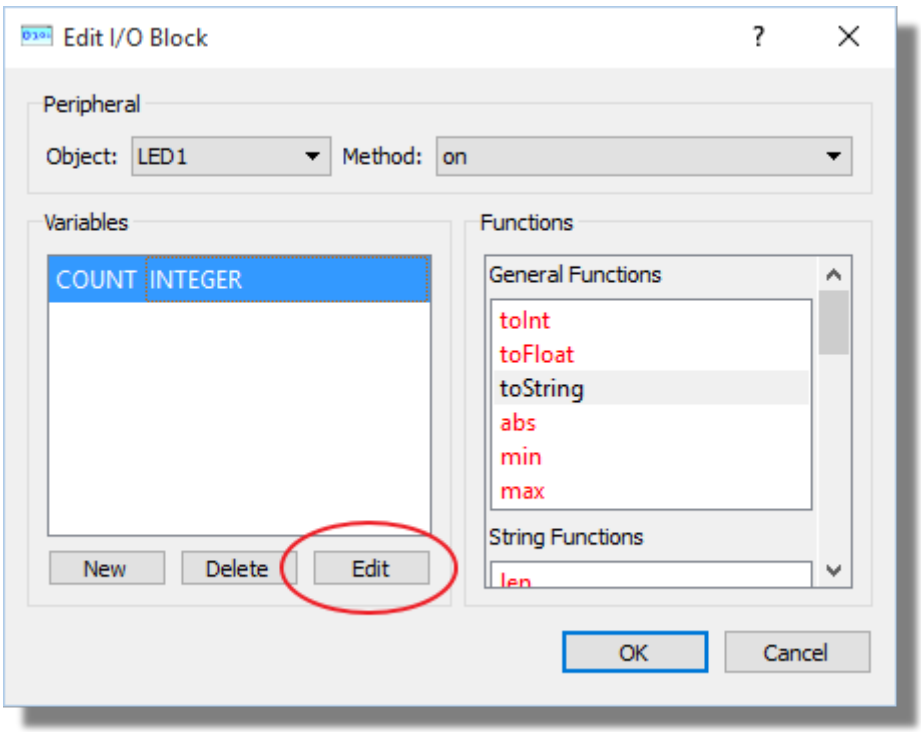


To edit a variable

1) Edit one of the blocks on the flowchart.

2) Select the variable from the list on the left hand side.

3) Click the edit button and change the name and/or type of the variable.

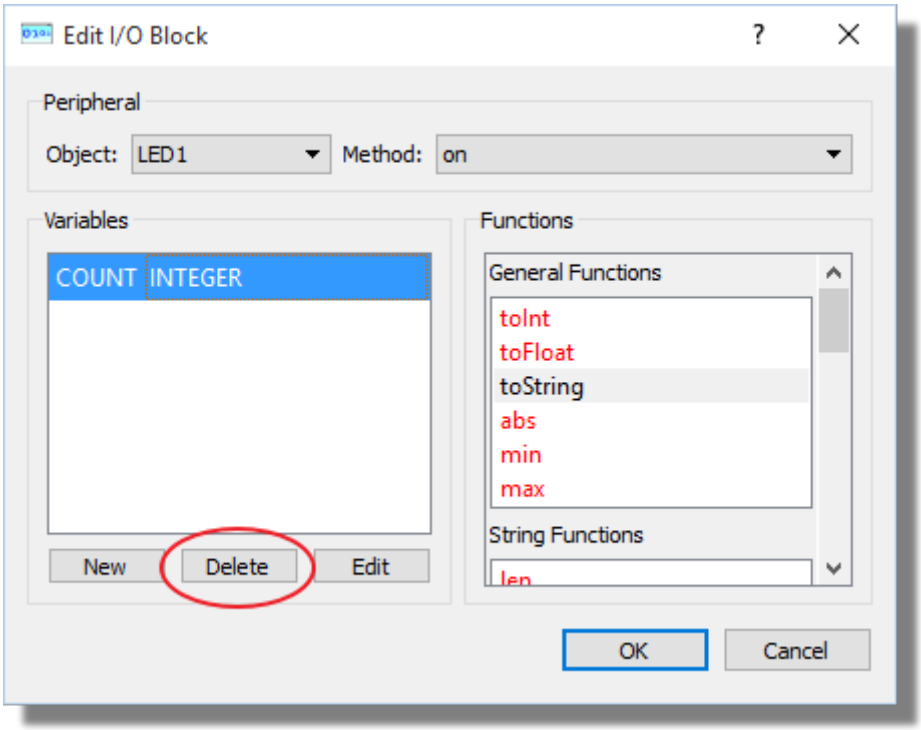


To remove a variable

1) Edit one of the blocks on the flowchart.

2) Select the variable from the list on the left hand side.

3) Click delete to remove the variable



If you edit or remove a variable you need to remember to check and change all the other flowchart blocks in which the variable has been used.

See Also:

Editing Blocks : Expression Editor

FLOWCHART BLOCKS

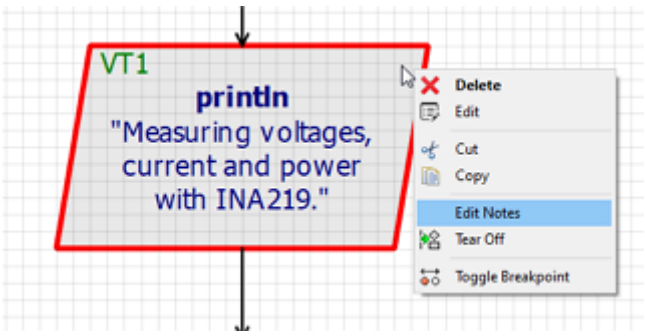
Flowchart Blocks

Visual Designer includes a small set of flowchart blocks which are used as the programming constructs in your firmware. These will be familiar to many users and are listed below in full.

ICON	DESCRIPTION	ICON	DECRIPTION
	Event Start Block: Used together with an end block to define a sub-routine or event handler.		Time Delay: A time delay in program execution.
	End Block: Used together with an event start block to define a sub-routine or event handler.		Decision Block: A boolean decision in the flowchart.
	Assignment Block: Used to assign a value to a variable.		Loop Construct: Used to simplify the configuration of different types of loop.
	Subroutine Call: Used to call a sub-routine.		Interconnector : Used in pairs to connect longer sequences of blocks into separate columns on the editor.
	Stored Data Block: Used to specify an action on a storage object (such as an SD Card).		Comment : Allows you to add a textual comment on the flowchart.
	I/O (Peripheral) Operation: Used to specify an action on the hardware, either internally to the CPU (e.g. timer) or external (e.g. LCD).		

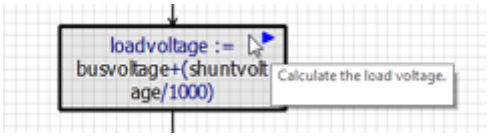
Commenting Blocks

You can add a dedicated tooltip comment to any block via the Edit Notes command on the context menu for the block



Adding a tooltip comment to a block.

Blocks with a comment will be denoted with a little blue triangle at the top right and the comment will be displayed on mouse hover.



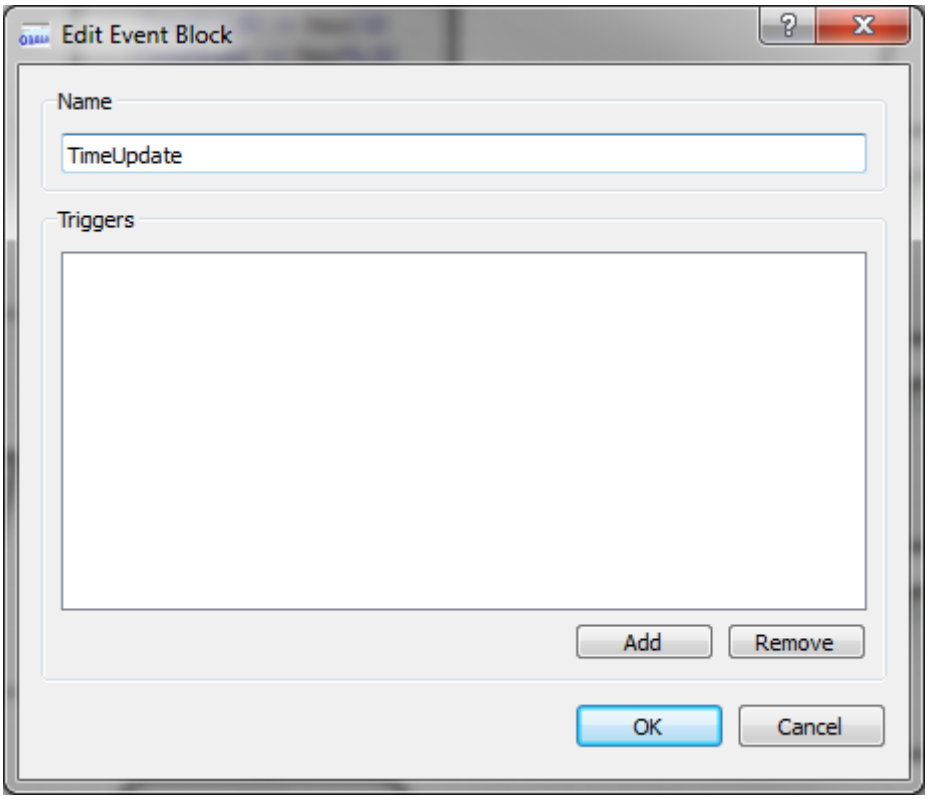
Tooltip on mouse over where a comment is defined.

> If you need a comment which is always visible then you can add text to the chart itself using the comment block.

Event / Subroutine Block

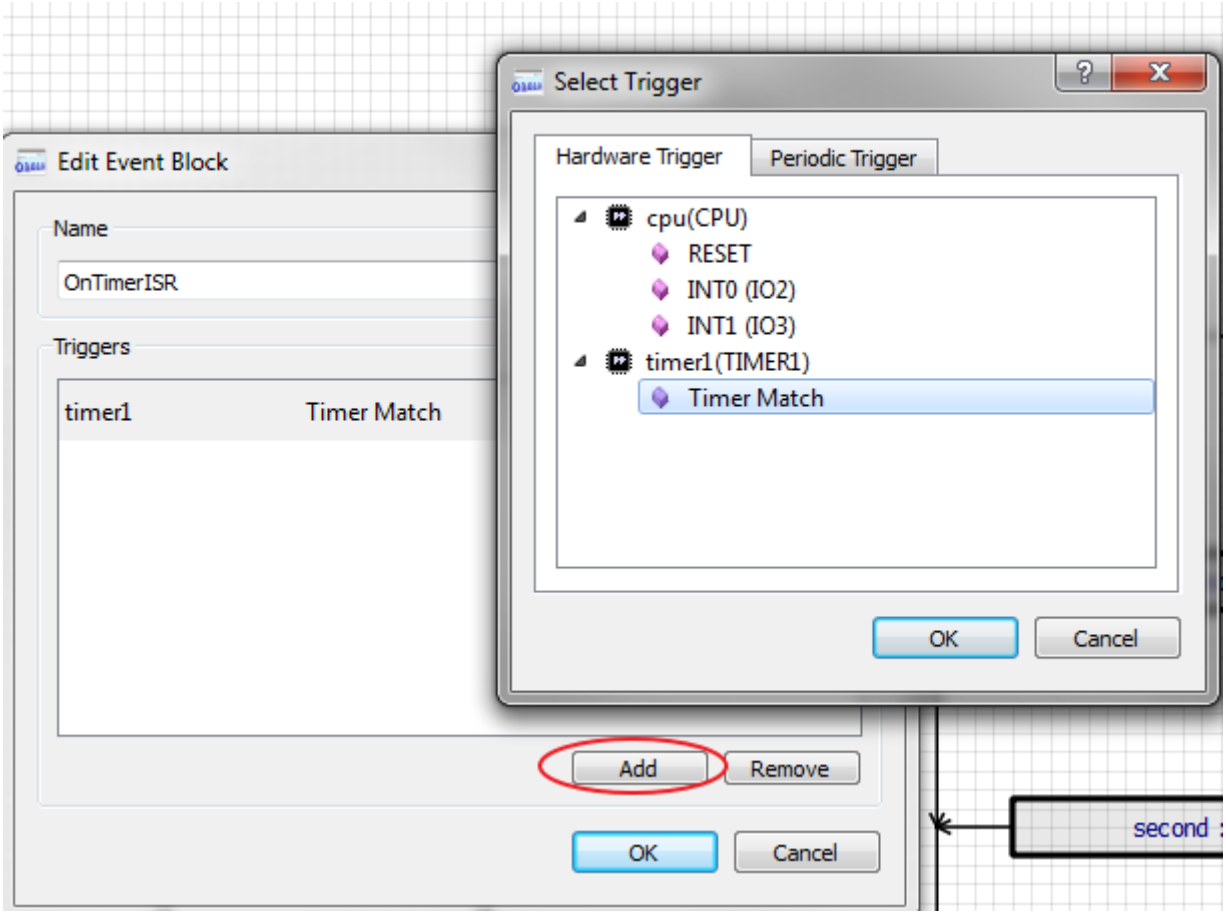
The event block is used together with the end block to define the start and end of sub-routines (e.g. to write to a display) and event handlers (e.g. to handle a timer interrupt).

If you are creating a subroutine then all that is required is a name for the block. This is the name of the routine and will be selectable when placing and editing subroutine call blocks.



Simple Subroutine Config - Only the name is required.

If you are creating a routine to handle a triggerable event such as an interrupt handler then you need to also specify the trigger



Setting up a timer interrupt service routine based on a timer1 match event.

> The periodic trigger provides a very handy way to get a repeatable call at regular time intervals. It does however make use of one of the CPU timers so be careful not to use it elsewhere if you configure it here for use here.

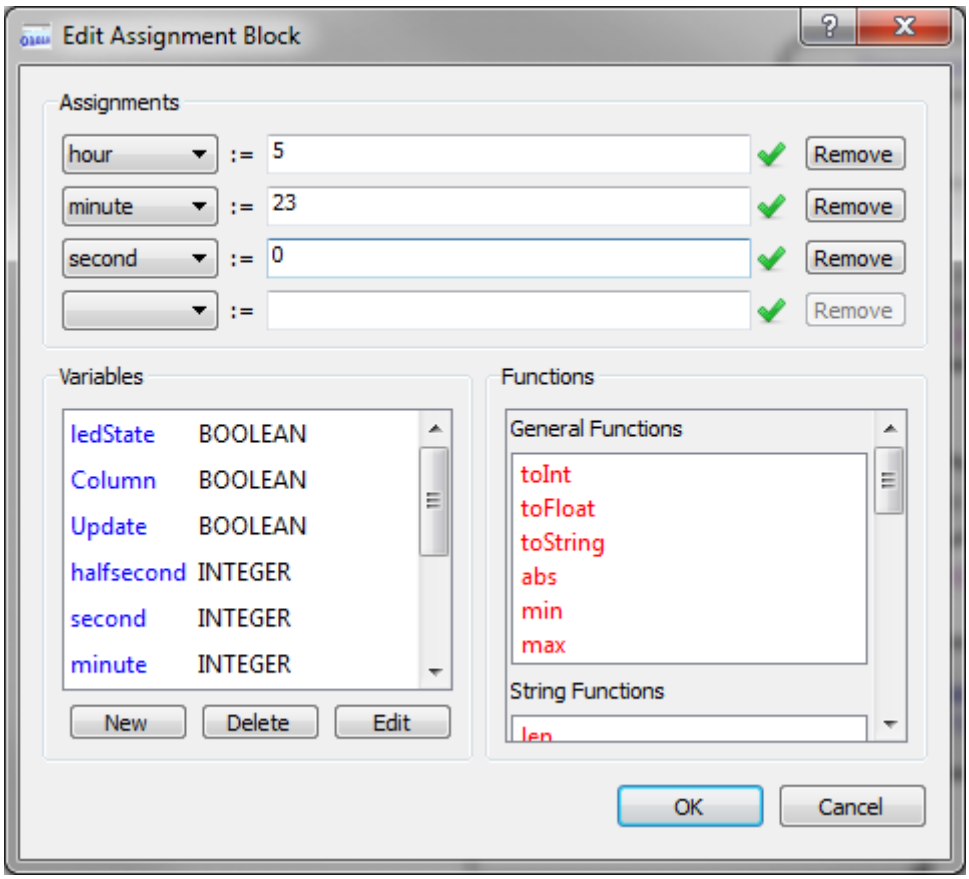
End Block

The end block is used to terminate a routine or event handler and is normally placed together the event block.

> A typical Arduino project will initialize with two event and end blocks, one for program setup and the other for main program execution.

Assignment Block

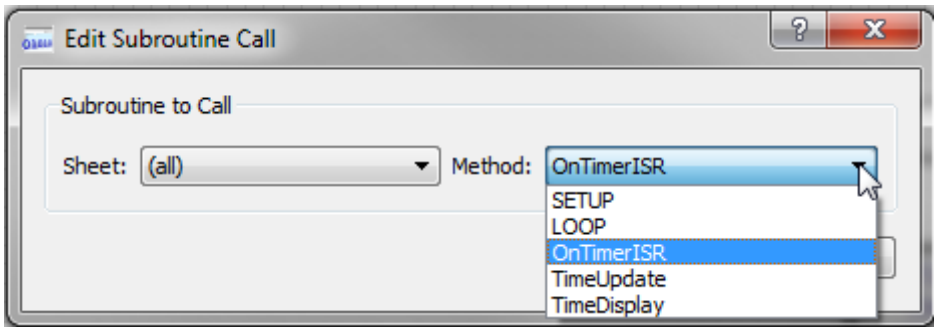
This is the standard block for assigning values to variables. It is also the place to create new, edit and delete variables.



Assigning values to variables for the time.

Subroutine Call Block

The subroutine block lets you call a function defined elsewhere on the flowchart.

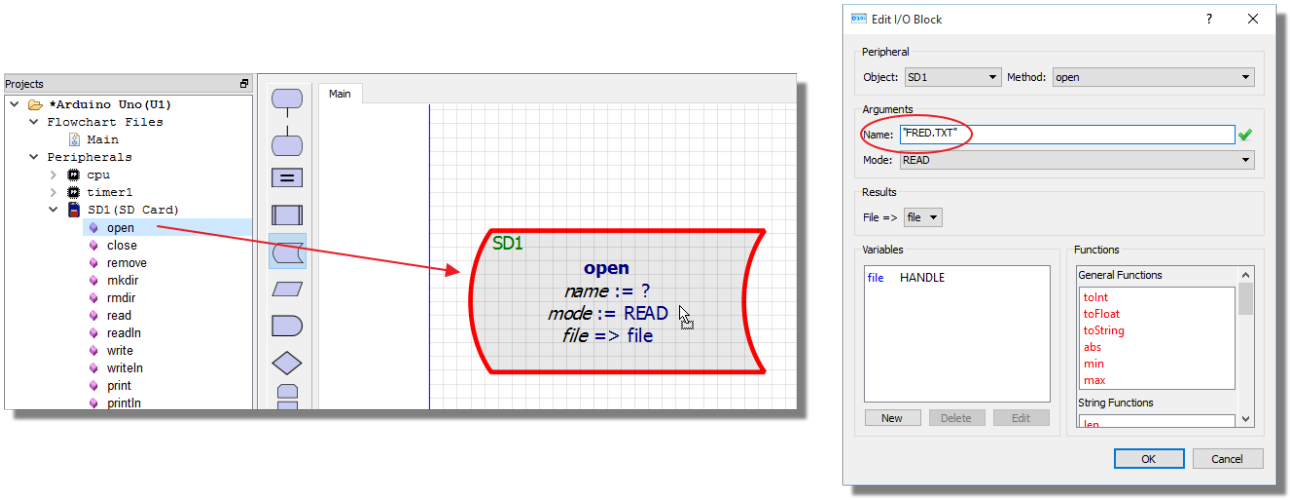


Specifying the subroutine to call.

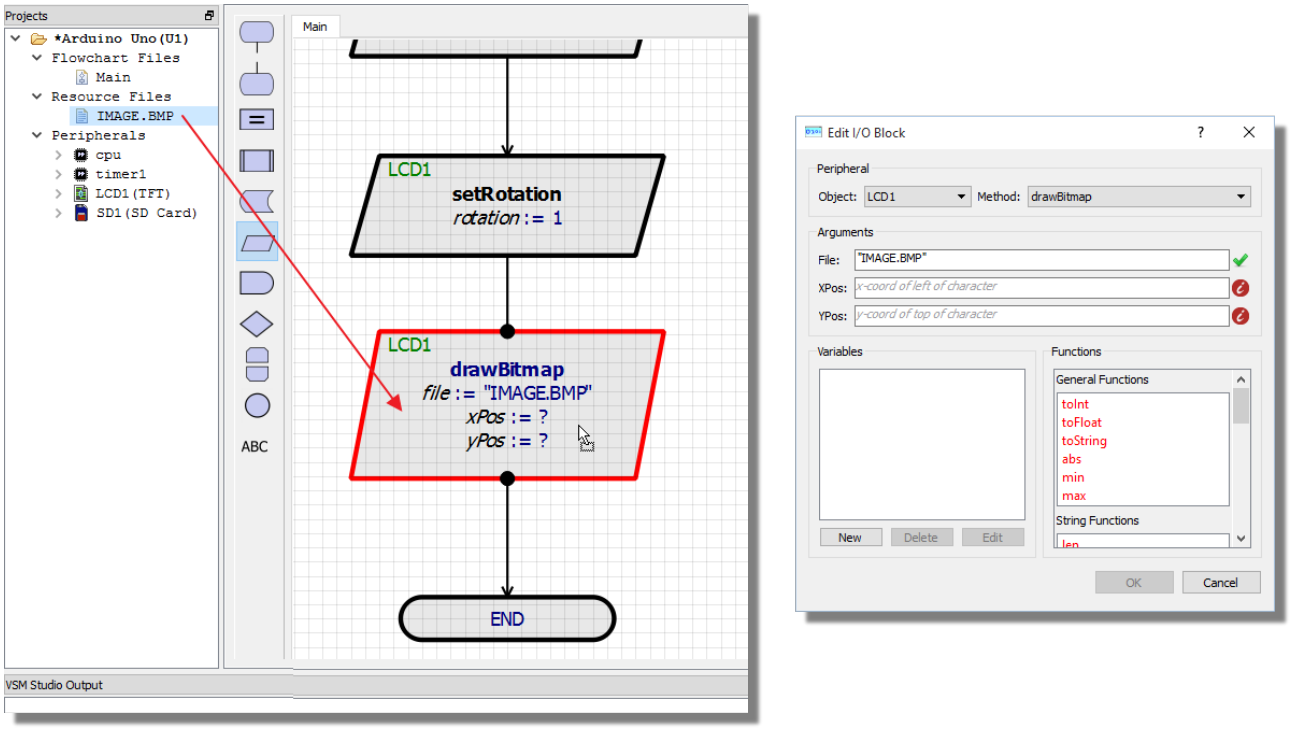
> You must first create and name the subroutine with an event and end block. The combo-box on the subroutine dialogue lists all named routines.

Stored Data Block

The storage block is used to denote an action or method on a storage object (SD Card). The methods are then typically used with resources (e.g a text file) to manipulate the filestore.



When you add a shield that contains an SD card it will almost always have a method for using the contents of the SD Card. For example, a TFT shield has a DrawBitmap() method and a Wave shield has a Play() method. In these cases, you should not use the storage data blocks but rather just drag and drop the resource onto the chart

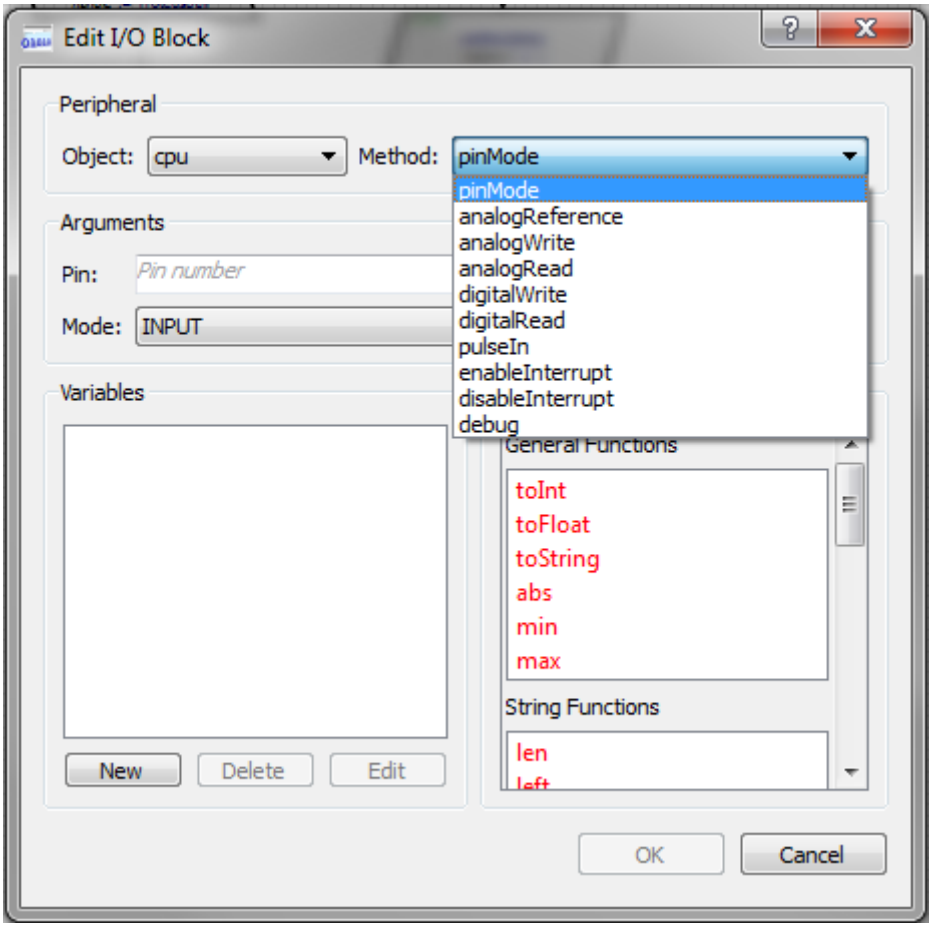


See Also:

- Resources and the Filestore
- Tutorial 3: SD Card and TFT Display.

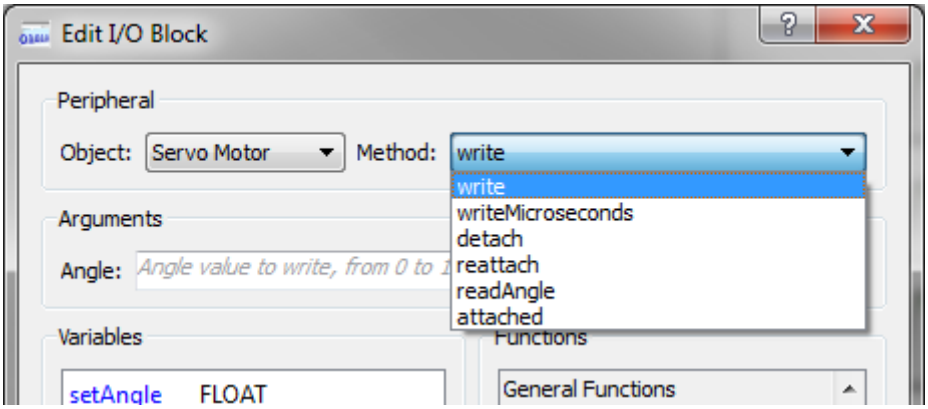
Peripheral Operation Block

A peripheral operation block basically allows you to perform an action on the hardware. With a new project the hardware will consist of only the processor and the available actions are therefore limited.



Available actions on the processor.

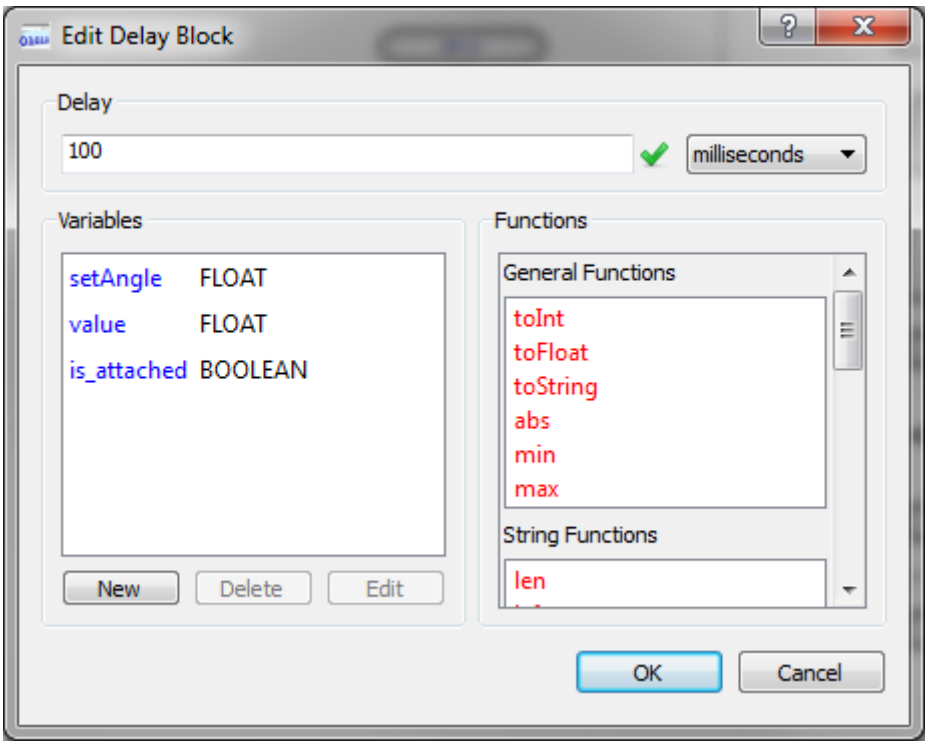
However, as you add peripheral shields to your project you will be able to interact and control their I/O via this block type.



Available methods on the Grove Servo Motor Shield.

Time Delay Block

This block is used to introduce a specific delay in the program.

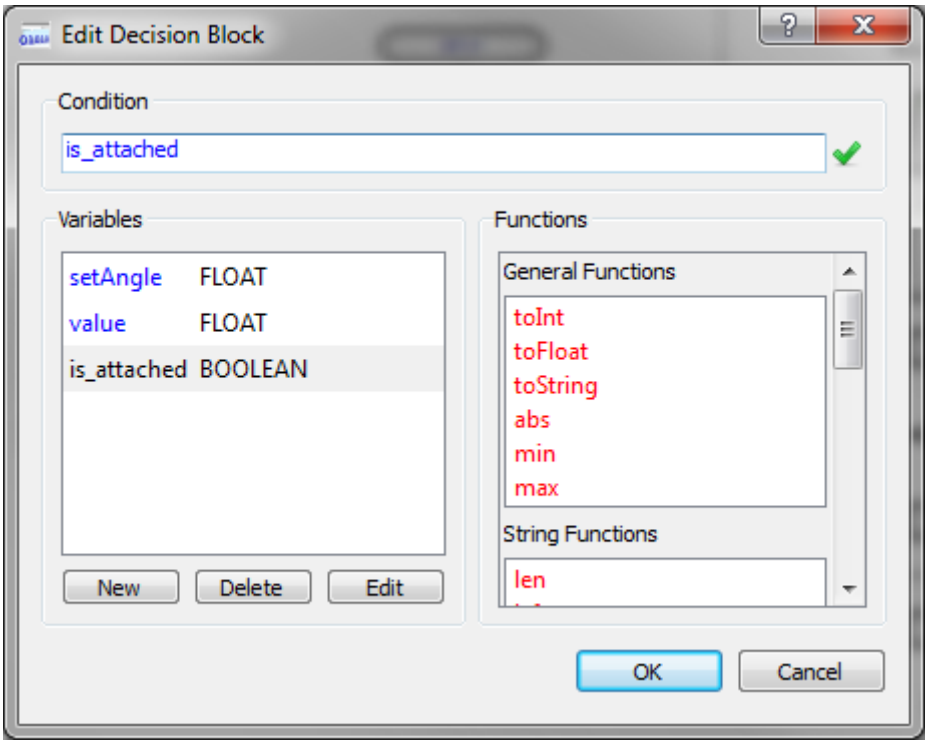


Specifying a 100ms Delay in the program.

> In Arduino, no other reading of sensors, mathematical calculations, or pin manipulation can go on during the delay function, so in effect, it brings most other activity to a halt. Interrupts and other specific functions will however still work. Refer to the Arduino documentation on Delay() for more detail.

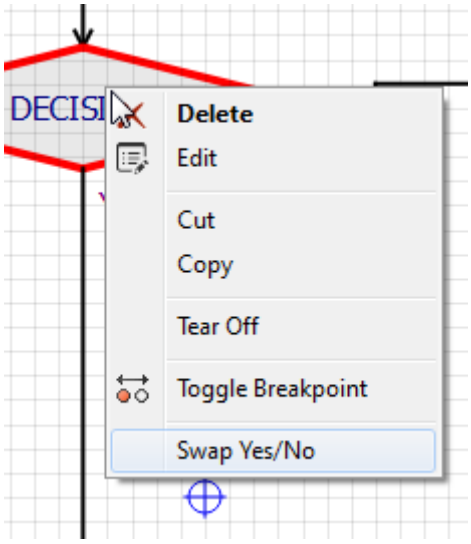
Decision Block

The decision block is the basic yes/no question that determines desired program flow. The dialogue form is fairly self-explanatory and requires a boolean expression.



Testing the is_attached variable in a decision block.

When a decision block is linked on the flowchart, the YES and NO labels are placed in default locations. You can swap them over via the context menu if that better suits the flowchart.



Swap the YES and NO flowlines on a decision block.

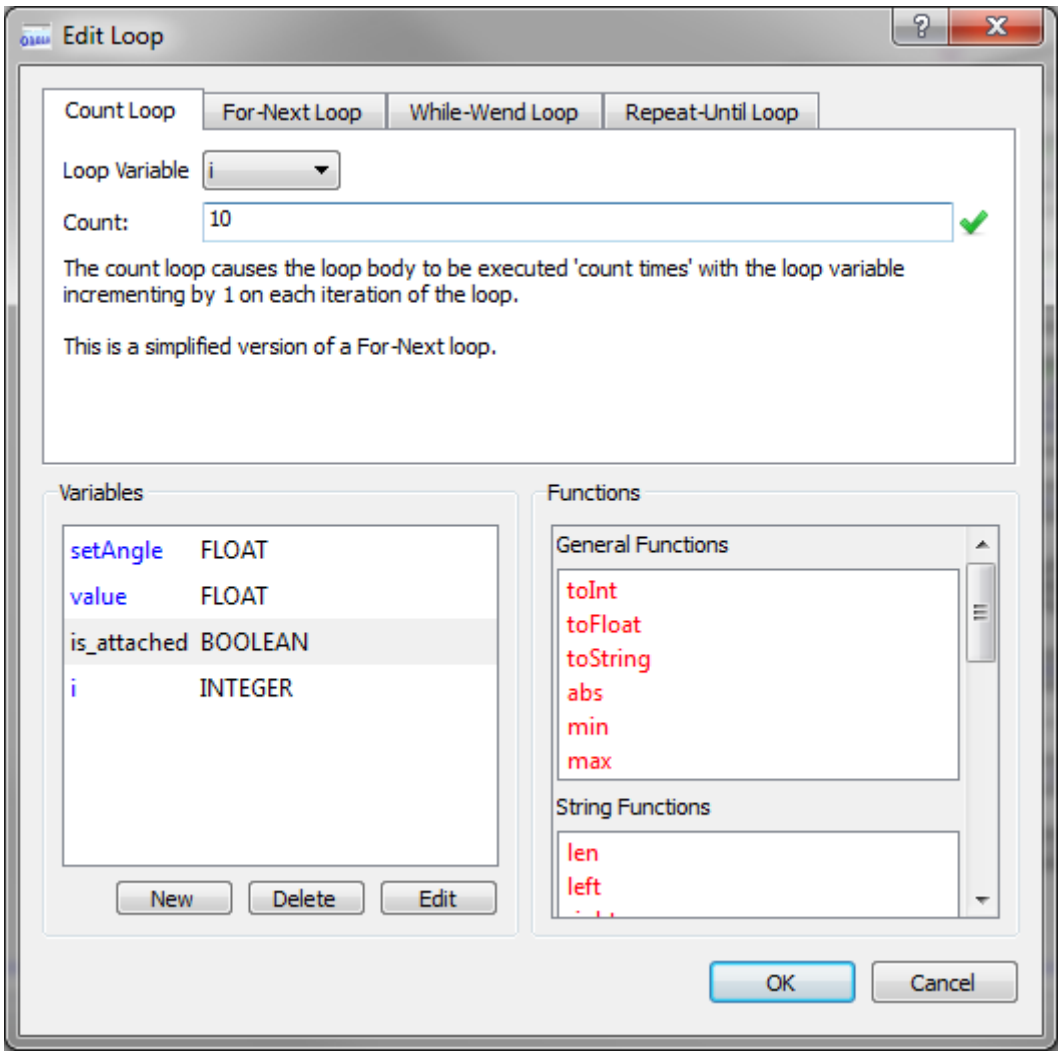
Loop Construct Block

The loop construct provides a simple, dialogue driven way to configure some common types of program loop. It is placed as a pair and the required logic blocks are then attached inside the loop.

The edit loop dialogue then provides several options:

Count Loop

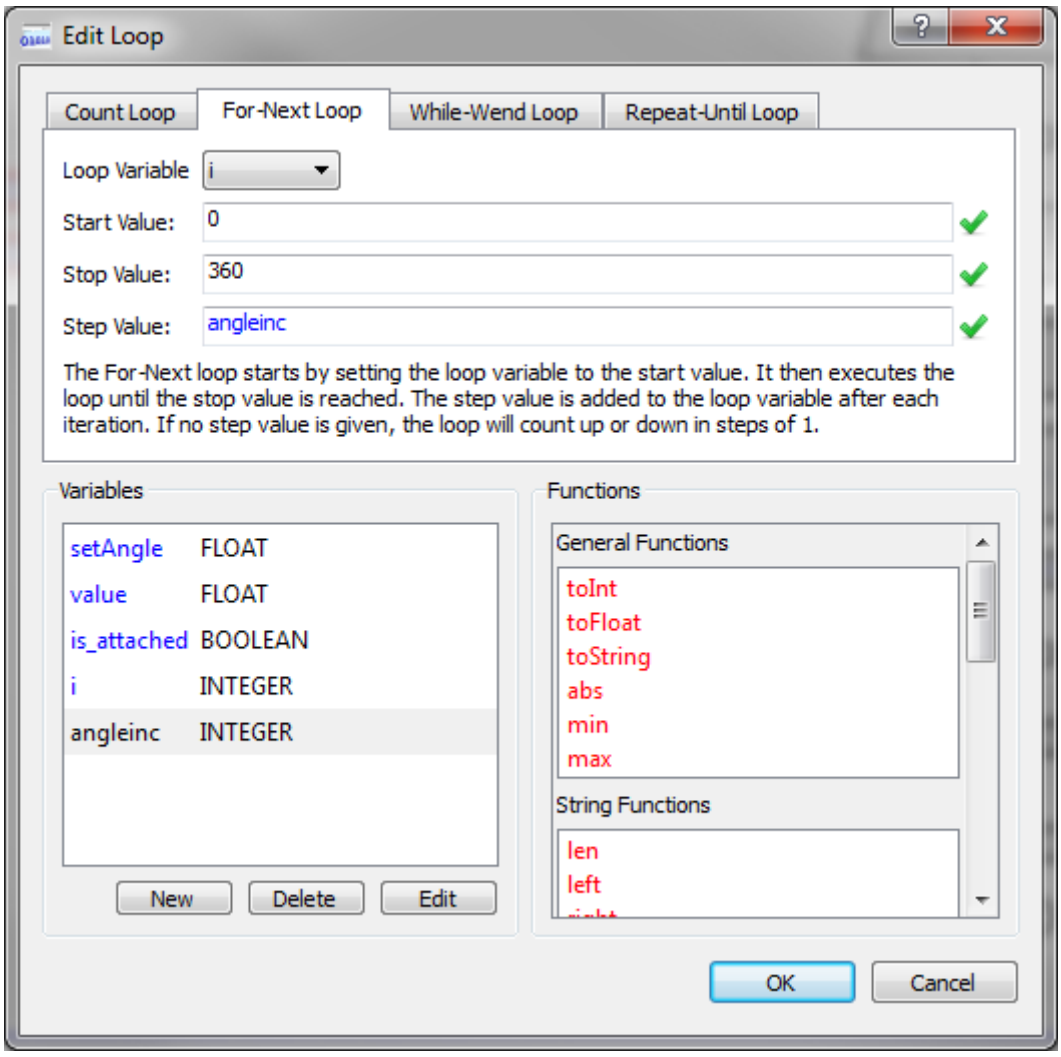
The count loop is simply a way of executing the loop body a specified number of times.



Count Loop

[For-Next Loop]

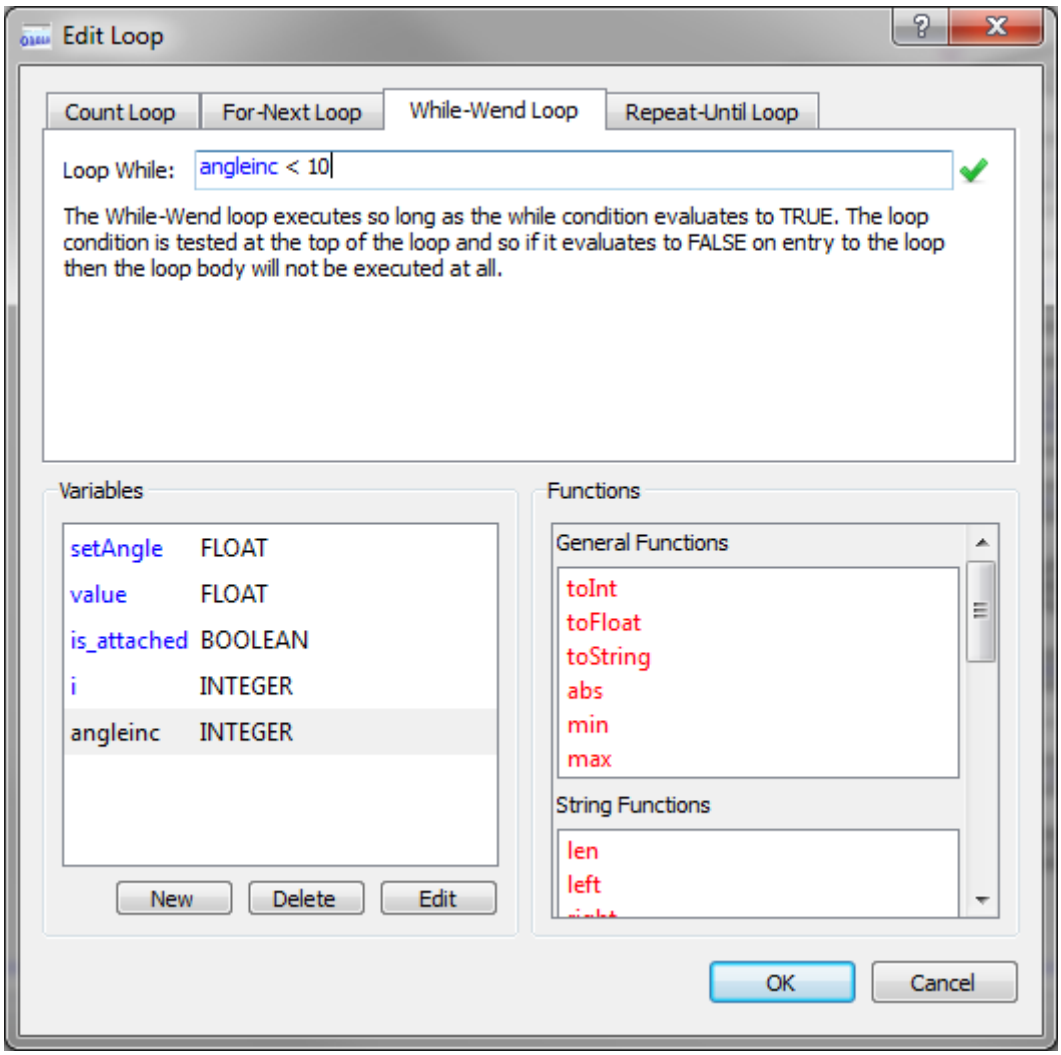
A For-Next Loop is an extension of the count loop as you need to specify the start, stop and step to determine how often the loop body will execute.



For-Next Loop.

[While-Wend Loop]

This construct executes the body of the loop while the tested condition evaluates as TRUE.



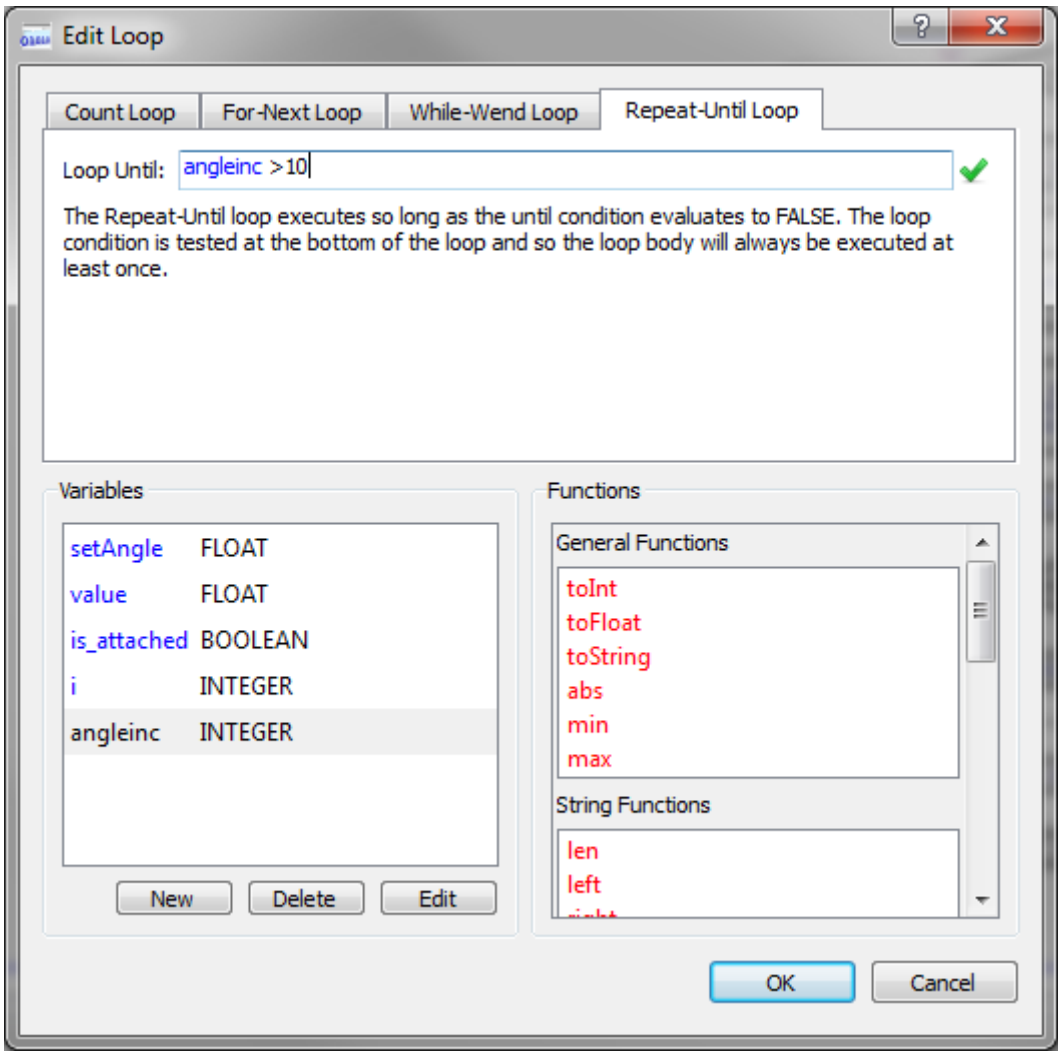
While-Wend Loop

> Note that the test takes place at the top of the loop and so if the test is FALSE to begin with the loop contents will not be executed at all

Repeat-Until Loop

Repeat-Until is very similar to While-Wend except:

- It executes until the tested condition evaluates to FALSE
- The test takes place at the bottom of the loop and so the loop body always executes at least once.

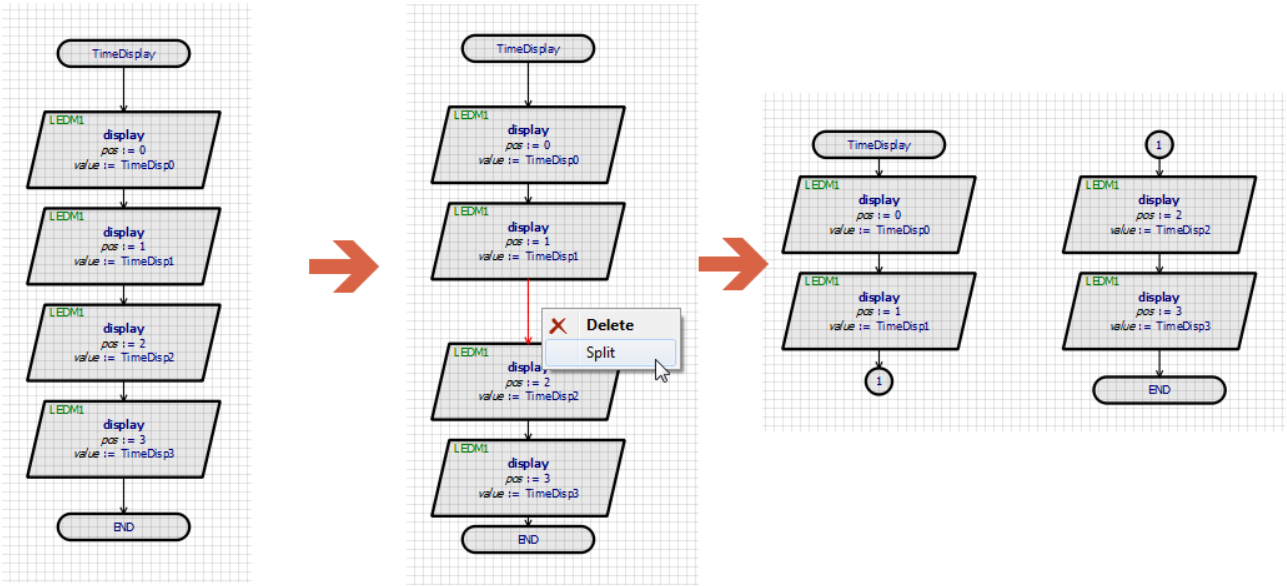


Repeat-Until Loop

Interconnector Block

The interconnect blocks are basically 'virtual connections' and need to be in pairs. If you have two interconnect blocks with the same number then you can imagine an invisible wire joining them together. The purpose of the interconnect blocks is to let you split flowchart logic into multiple columns.

You can either drag and drop two interconnect blocks, link them on the flowchart and then re-number appropriately, or you can simply right click and split a flowline.

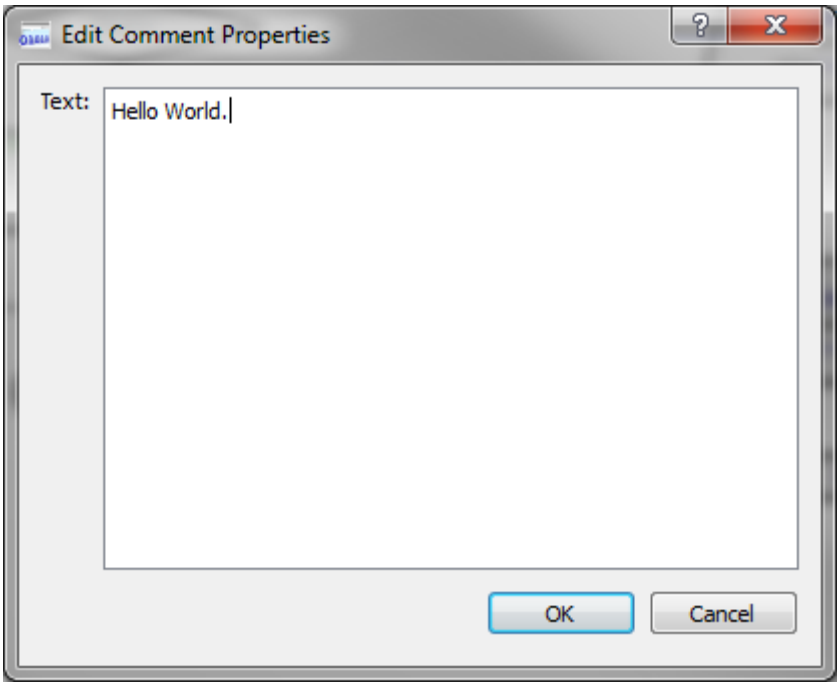


Splitting into two separate lines with interconnect

You can join two interconnects together into a single line by dragging one over the top of the other.

Comment Block

The comment block enables a free format block for the entry of descriptive text.



Comment Block Dialog

ARDUINO HARDWARE

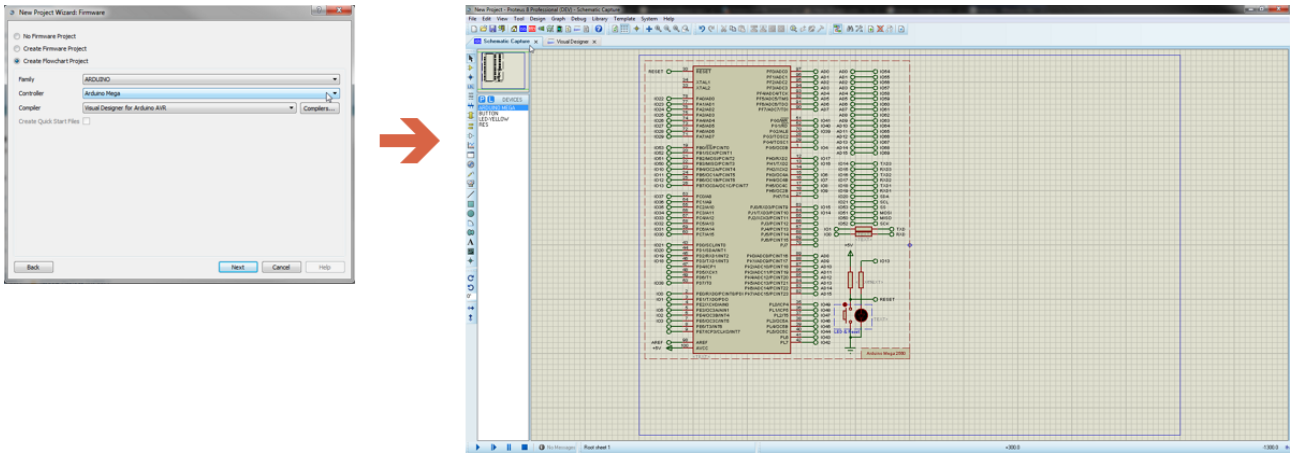
Introduction

Designing 'virtual hardware' for your project is greatly simplified in Visual Designer. The various options available to you are described in detail below.

Arduino Base Board

In hardware, Arduino is a system consisting of a fairly simple base board with an AVR microcontroller, some supporting electronics and Arduino pin headers (connectors). There are then a wealth of peripheral boards (shields) which plug into the pin headers to form the hardware for a particular project.

In Visual Designer for Arduino, the creation of the project is the point at which you make your decision on the base board (E.g. Arduino Uno, Arduino Mega). This is automatically placed on the schematic when the project is created.



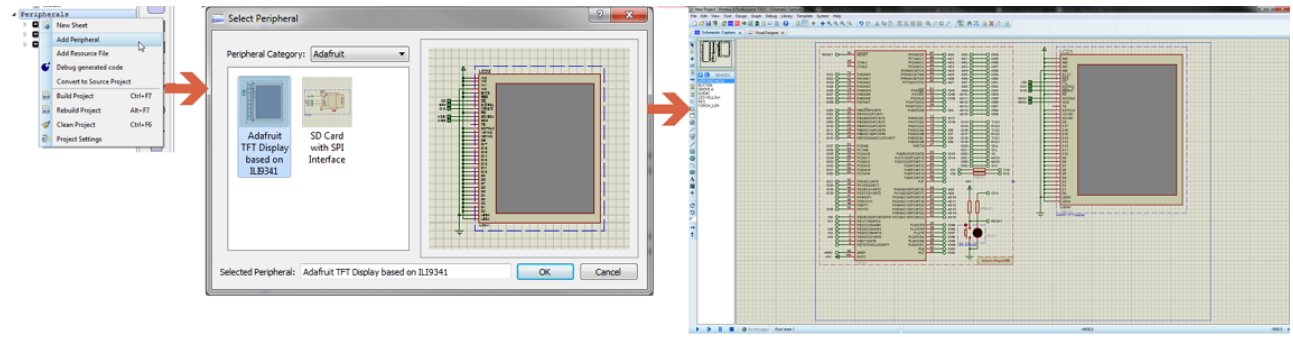
Automatic placement of the Arduino Mega on the Schematic when the Project is created.

Arduino Shields

Arduino shields are modular circuit boards that piggyback onto your Arduino base-board to instill it with extra functionality. They are the simplest way to add hardware functionality to your project because the circuitry and connectivity on the schematic happens automatically.

Adding a Shield

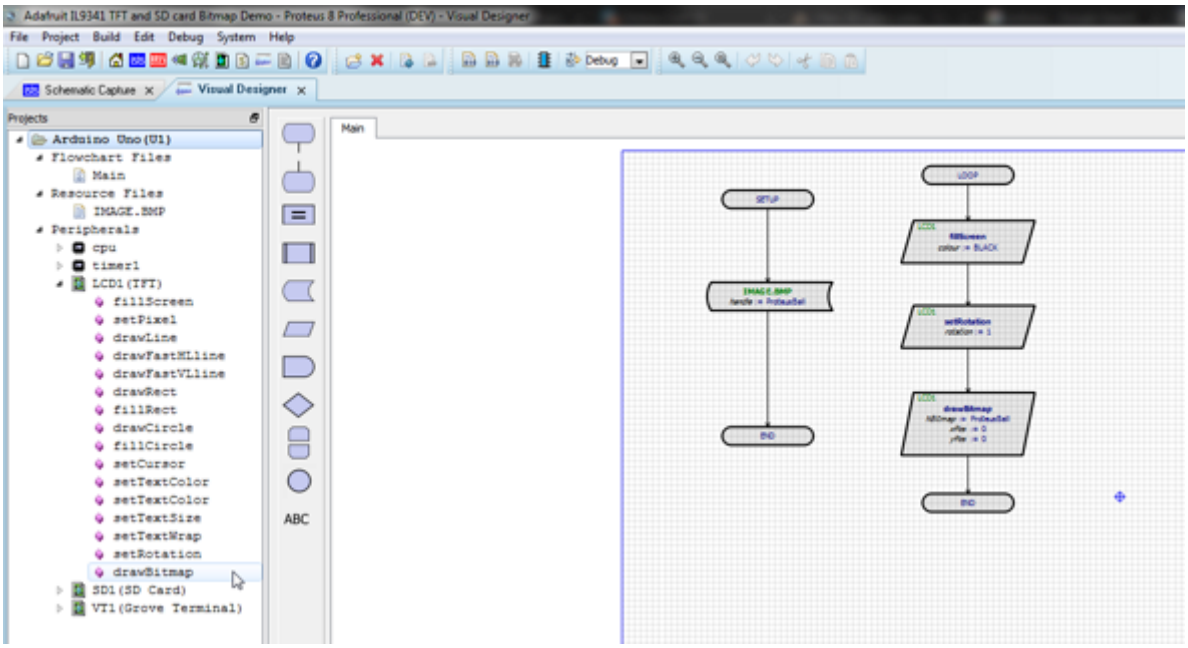
When you then want to add a shield in Visual Designer you right click on the Project Tree and select 'Add Peripheral' from the resulting context menu. Select the 'Adafruit' category on the peripheral browser and then select the shield you want to add. This will auto-place the shield electronics on the schematic.



Adding a Peripheral will auto-place and auto-connect on the schematic.

Writing a Program to Control a Shield

Shields appear in the project tree in Visual Designer once picked from the Peripheral Gallery. They include high level methods that allow you to quickly control the connected electronics via flowchart blocks.



Drag and Drop methods make it simple to drive the shields.

Potential Pitfalls

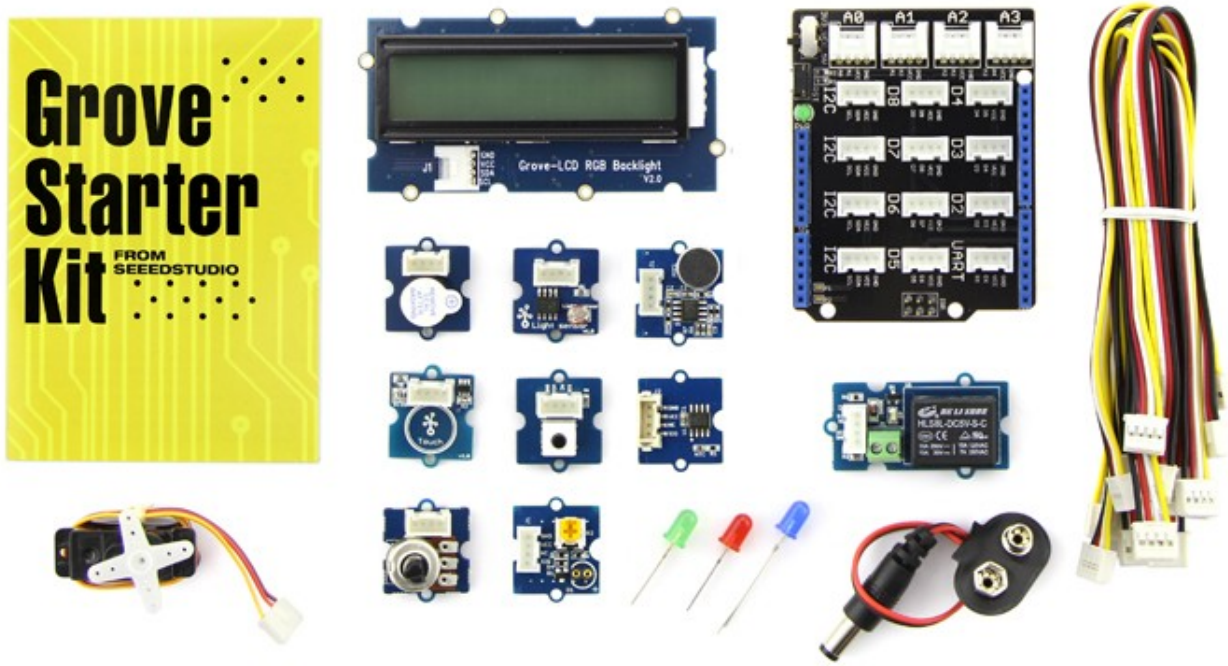
Working with Viirtual Shields is the simplest method to quickly create projects in Visual Designer. You do however still have to take care with what hardware you choose to use. For example:

- You need to make sure that your use of Arduino shields is compatible with hardware. For example, some shields are incompatible because they make use of the same resource on the CPU (e.g. Timer1). Others are pin incompatible.
- You need to make sure that your target Arduino base board has enough memory to handle the shields you are adding to it. In particular, with an Arduino Uno you can run out of memory if you add shields with large and complex software stacks.

Grove Sensors and Modules

Grove is a modular electronic platform for quick prototyping. Every module has one function, such as touch sensing, creating audio effect and so on. You simply plug the modules you need into the base shield and then connect the base shield to the Arduino base board in the normal way.

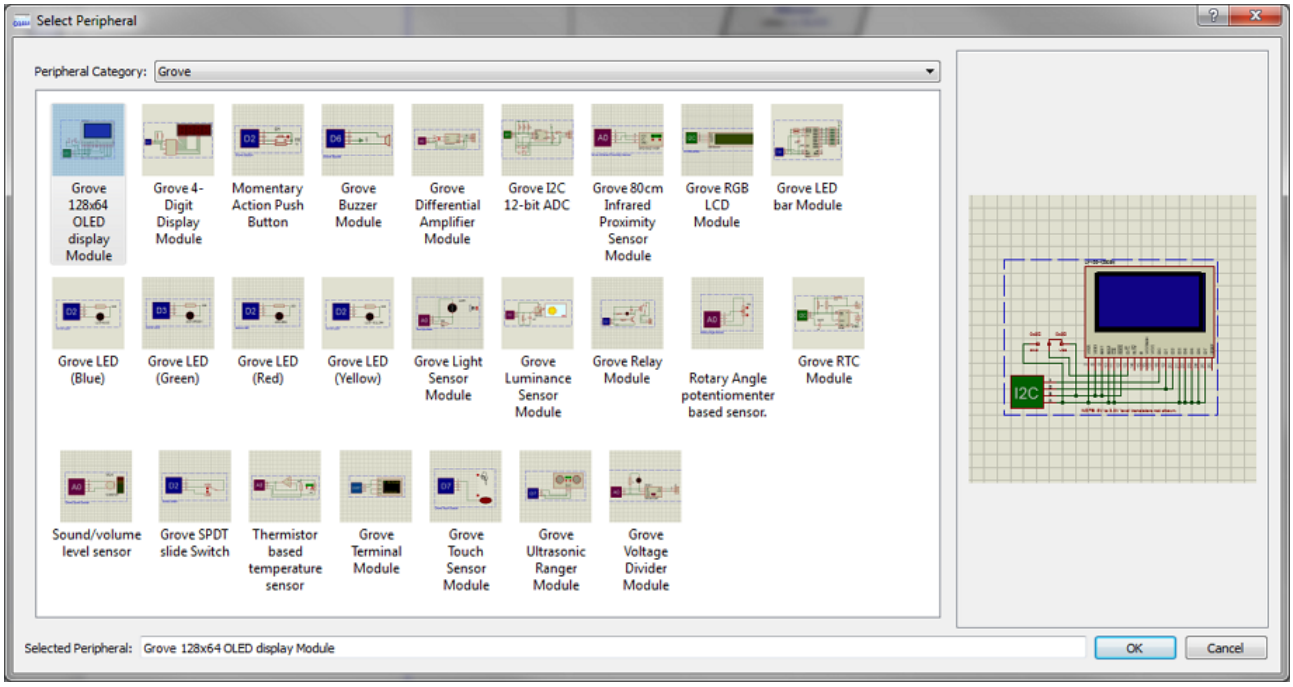
The Grove Starter Kit Plus is a great way for beginners and students to get started with Arduino. The Grove Base Shield drops onto your existing Arduino and lets you connect up to 16 Grove modules without any fuss. The Arduino headers are still broken out on the top of the shield making it very easy to connect other breakout boards and sensors to your Arduino.



Grove Starter Kit Hardware.

Adding a Grove Module

You can add a Grove module in the same way as adding a shield, except that you select the Grove category in the Peripheral browser.

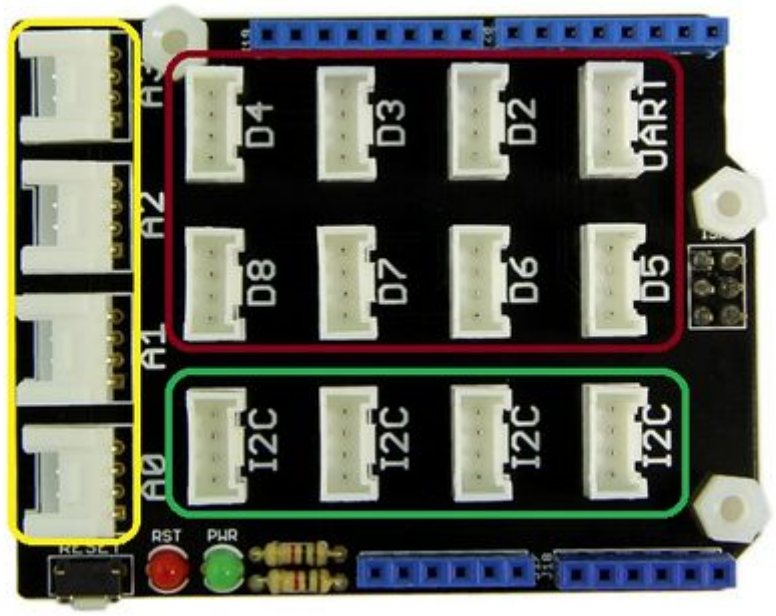


Adding Grove Starter Kit modules in Visual Designer.

You can then control the Grove peripherals via the high level methods available from the Project Tree. Multiple peripherals and sensors can easily be added to a single project which provides an extra level of flexibility over using standard Arduino shields.

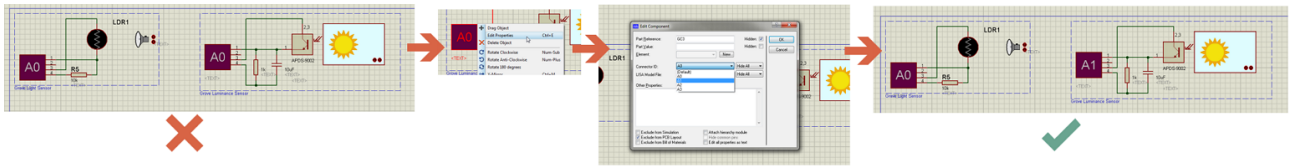
Potential Pitfalls

When you are working with Grove modules in Visual Designer the process is identical to working with complete Arduino shields. However, since the modules themselves plug into an Arduino shield, you need to take care not to re-use a connector on the Grove board. The real Grove base-board shield will look something like the following:



Grove Shield - the sensor modules plug-in to the exposed header pins.

When you add a Grove peripheral it will be assigned an ID on the schematic corresponding to a connector on the real board. If you get a duplicate, you'll need to edit and rename one of the connectors to a free slot.

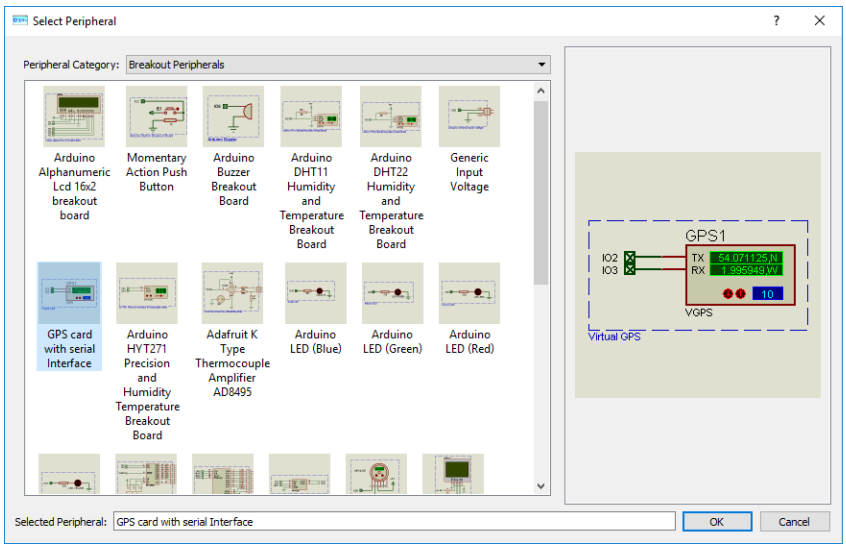


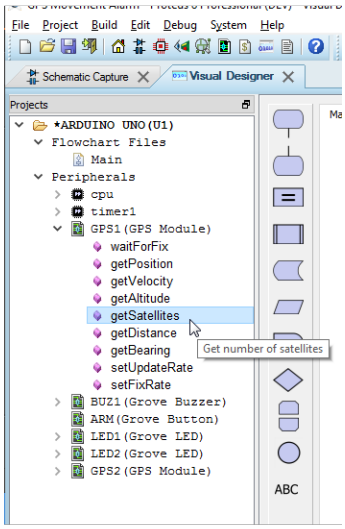
If two grove modules have the same connector, you need to edit one and assign to an unused header slot on the Grove Shield.

> ⚠ You need to make sure that your use of Grove modules is compatible with hardware. Since the pins used in each socket overlap it is quite easy to mistakenly use the same pin twice. A good example of this is discussed in Tutorial 2

Breakout Boards and Peripherals

The third category of hardware that you will find in the Peripheral Gallery is breakout boards. These are useful and interesting pieces of electronics (often single components) for which we have provided control methods inside Visual Designer. So, when you add a breakout peripheral to your project, you can drive it from the methods it adds to the Project Tree.





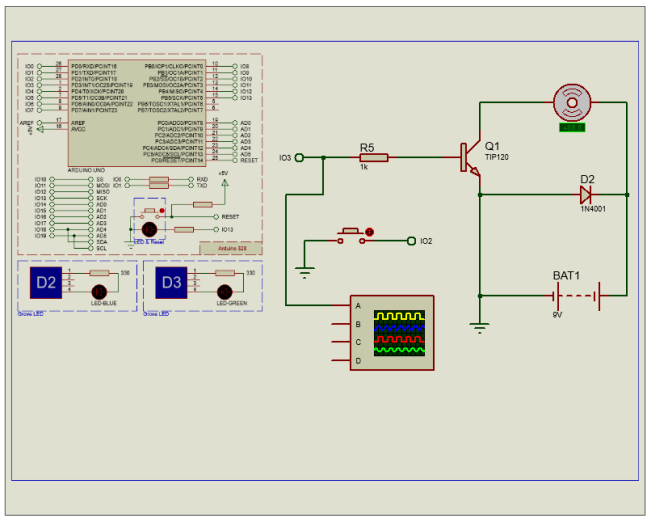
Breakout boards provide even more flexibility on the schematic. Indeed, they often connect to other schematic components that you want to place and wire yourself.

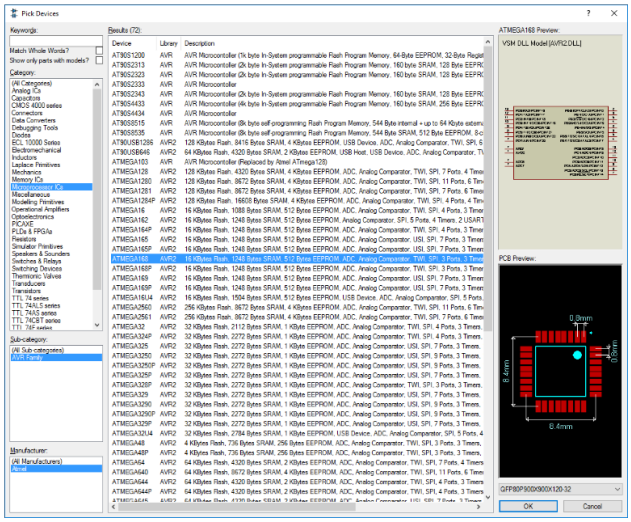
Potential Pitfalls

CPU resources and pin assignments need careful consideration.

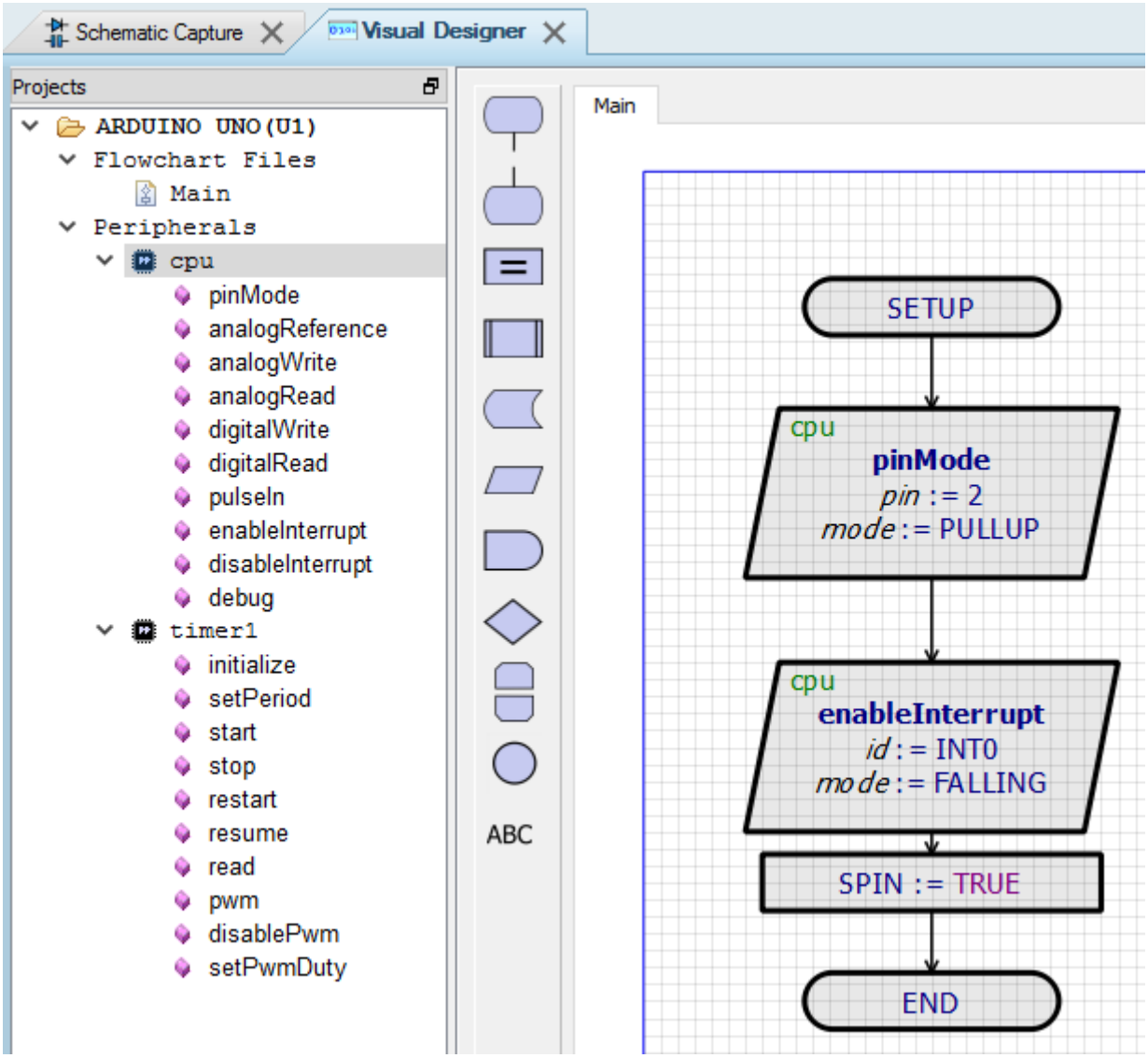
Schematic Design

With Visual Designer you have access to the full power of the Proteus VSM simulation environment. This means that you have a professional schematic capture tool with tens of thousands of embedded components that you can place, wire and simulate.





This is the most flexible design method of all but involves a significantly higher learning curve. The programming in Visual Designer will have to be done via the CPU methods rather than specific methods for peripherals.



See Also :

Tutorial 5: Introducing Schematic Design

Potential Pitfalls

Schematic design for simulation is a large and relatively complex topic. In addition to the basic of placement and connectivity there will also be power considerations and you will need to program for interconnect protocols where appropriate. At this stage, the transition from Visual Designer to Proteus VSM should be considered as there is a point beyond which C programming is required to effectively design the embedded system.

RASPBERRY PI HARDWARE

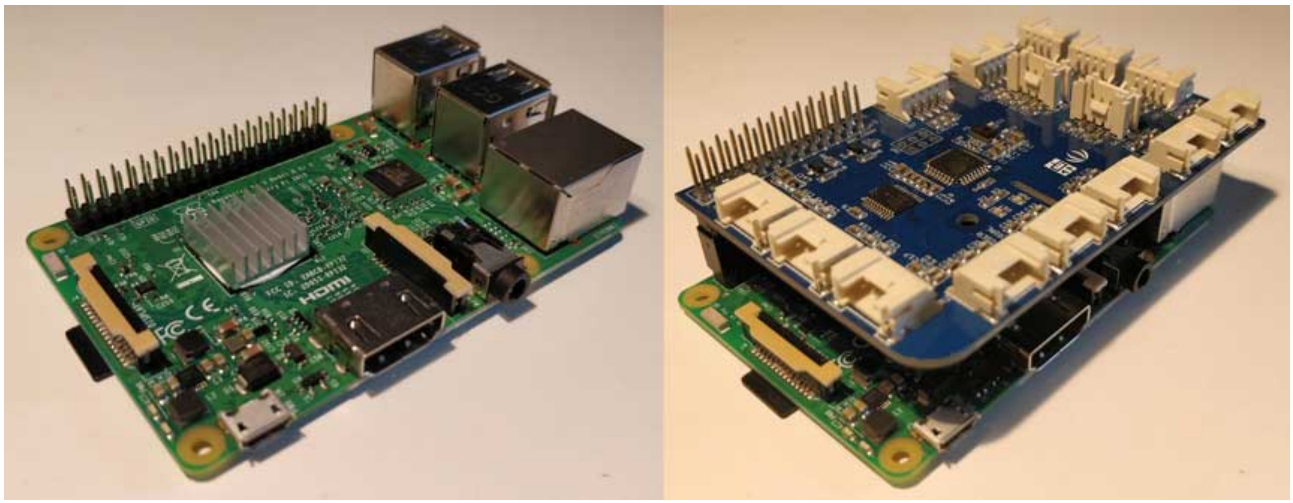
Introduction

Designing 'virtual hardware' for your project is greatly simplified in Visual Designer. The various options available to you are described in detail below.

Raspberry Pi Base Board

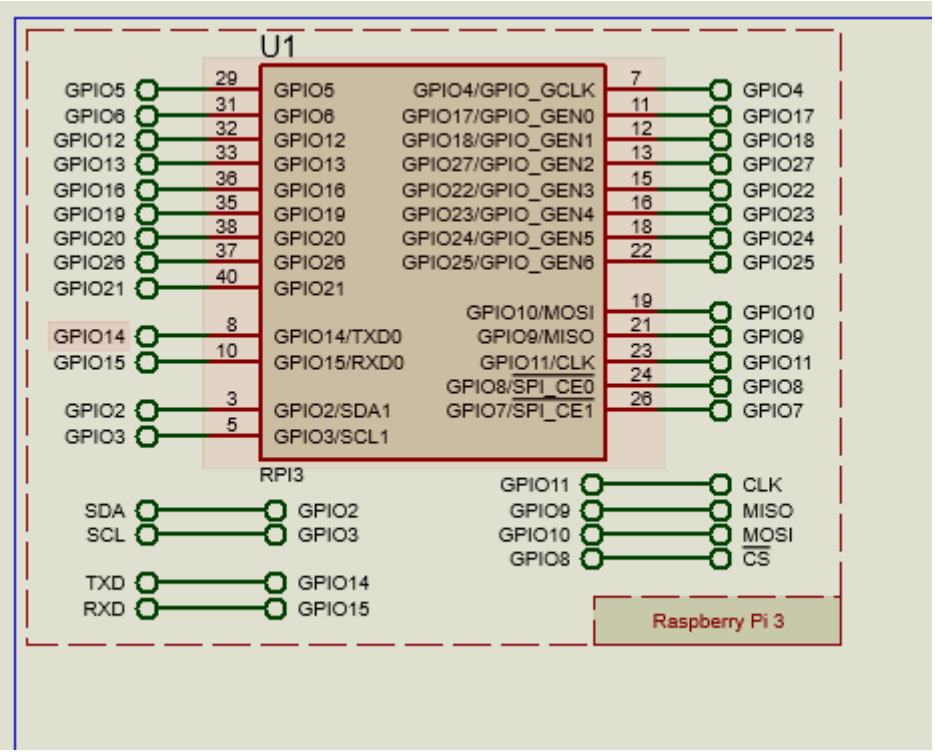
In hardware, Raspberry Pi projects start with the Pi computer board and then sometimes will have additional electronics attached via the header block. These can be either complete functional electronics blocks (known in the Pi world as 'hats') or loose electronics components such as LED's or buttons.

In Visual Designer for Raspberry Pi the creation of the project is the point at which you make your decision on the base board. You can choose between the Raspberry Pi 3 or the Raspberry Pi 3 with Grove. The latter can be thought of as the Pi3 with the Grove hat stacked on top of it



Equivalent Hardware: Raspberry Pi on the Left and Raspberry Pi with Grove Hat on the right

Your choice will be automatically placed on the schematic when the project is created.



Automatic placement of the Raspberry Pi 3 on the Schematic when the Project is created.

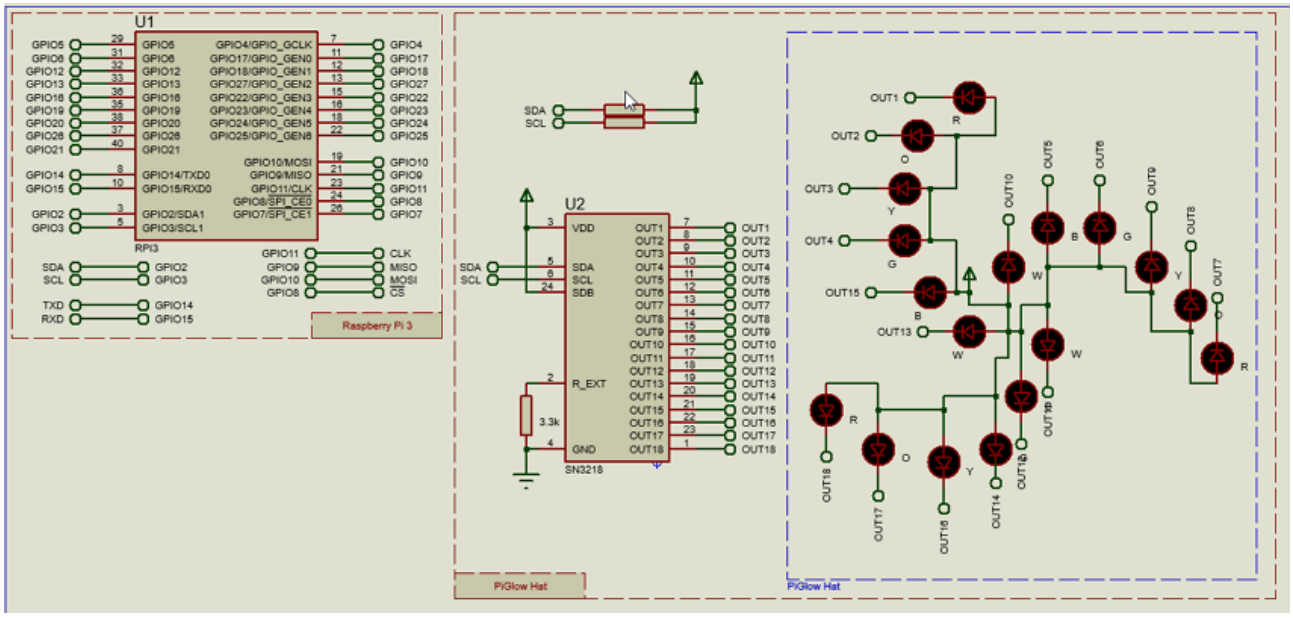
> In theory, the Raspberry Pi Zero W should also work as the pinout is identical and it contains the wi-fi chip. If you are working with this device select Pi 3 in the configurator. Raspberry Pi 2 will not work with Visual Designer because the pinout is completely different.

Raspberry Pi Hats

Raspberry Pi Hats are modular circuit boards that sit on your Pi3 base-board to instill it with extra functionality. They are the simplest way to add hardware functionality to your project because the circuitry and connectivity on the schematic happens automatically.

Adding a Hat

When you then want to add a hat in Visual Designer you right click on the Project Tree and select 'Add Peripheral' from the resulting context menu. Select either the 'Adafruit' or the Pimoroni category on the peripheral browser and then select the hat you want to add. This will auto-place the electronics on the schematic and connect it to the Pi baseboard.

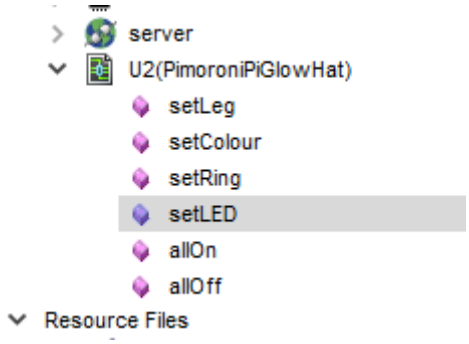


Adding a Peripheral will auto-place and auto-connect on the schematic.

> The connections are made by terminal name. Two terminals with the same name can be thought of as being connected with an invisible wire.

Writing a Program to Control a Hat

The Raspberry Pi Hat will appear in the project tree in Visual Designer once picked from the Peripheral Gallery. They include high level methods that allow you to quickly control the connected electronics via flowchart blocks.



Drag and Drop methods make it simple to drive the hats.

Potential Pitfalls

Working with Viirtual Shields is the simplest method to quickly create projects in Visual Designer. You do however still have to take care with what hardware you choose to use. For example:

- You need to make sure that your use of hats is compatible with hardware. For example, some hats are incompatible because they make use of the same resource on the CPU (e.g. Timer1). Others are pin incompatible.

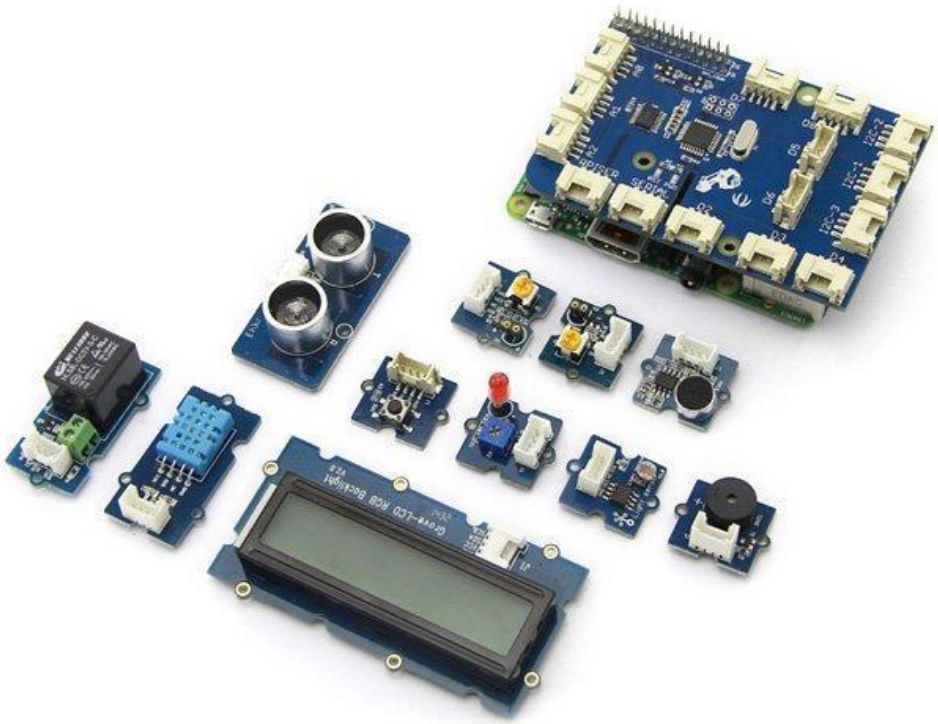
Grove Sensors and Modules

Grove is a modular electronic platform for quick prototyping. Every module has one function, such as touch sensing, creating audio effect and so on. You simply plug the modules you need into the base shield and then connect the base shield to the Raspberry Pi base board in the normal way.

The GrovePi+ is the Pi Hat which attaches to the Pi 3 baseboard and includes 15 Grove 4-pin interfaces that then let you directly connect Grove sensors to the Raspberry Pi.

In Visual Designer if you want to want to work with Grove and Raspberry Pi you should choose the Raspberry Pi 3 with Grove option during project creation.

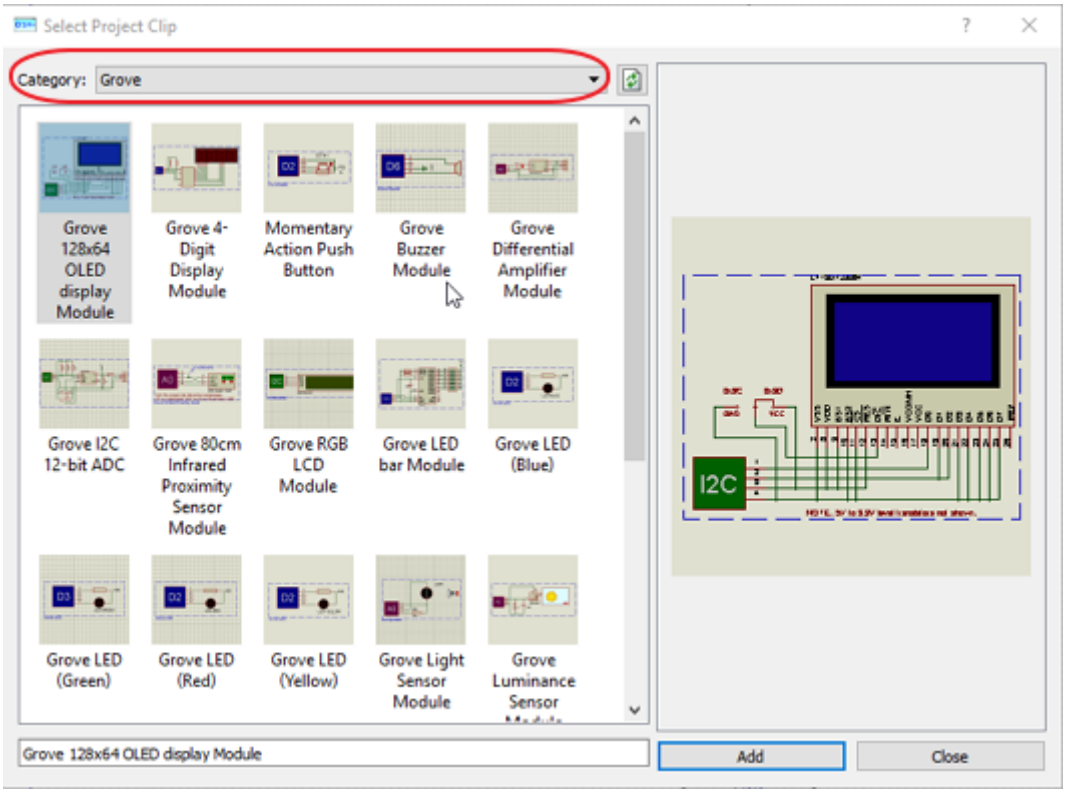
Grove Pi+ Starter Kit is a great way for beginners and students to get started with Raspberry Pi. The Grove Base Shield drops onto your existing Pi and lets you connect up to 16 Grove modules without any fuss. The headers are still broken out on the top of the shield making it very easy to connect other breakout boards and sensors to your Pi.



Grove Pi+ Starter Kit Hardware.

Adding a Grove Module

You can add a Grove module in the same way as adding a shield, except that you select the Grove category in the Peripheral browser.

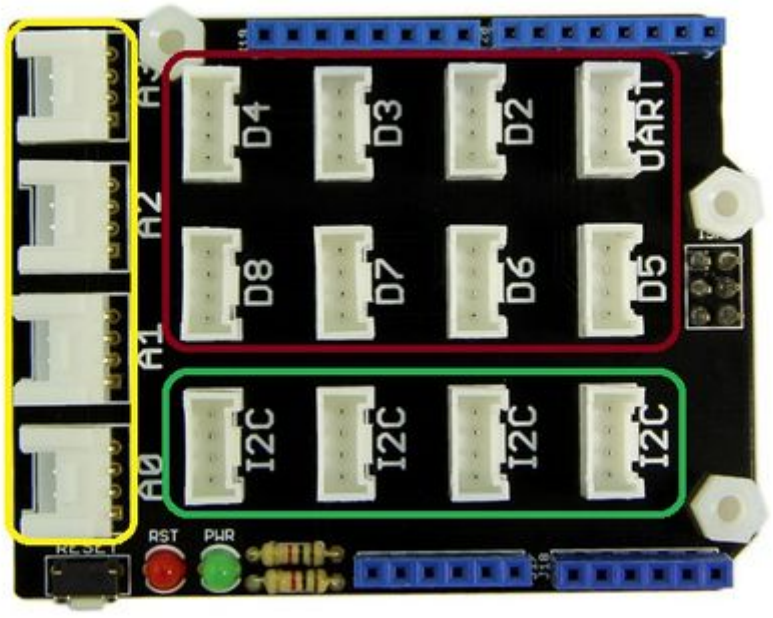


Adding Grove Starter Kit modules in Visual Designer.

You can then control the Grove peripherals via the high level methods available from the Project Tree. Multiple peripherals and sensors can easily be added to a single project which provides an extra level of flexibility over using standard Raspberry Pi shields.

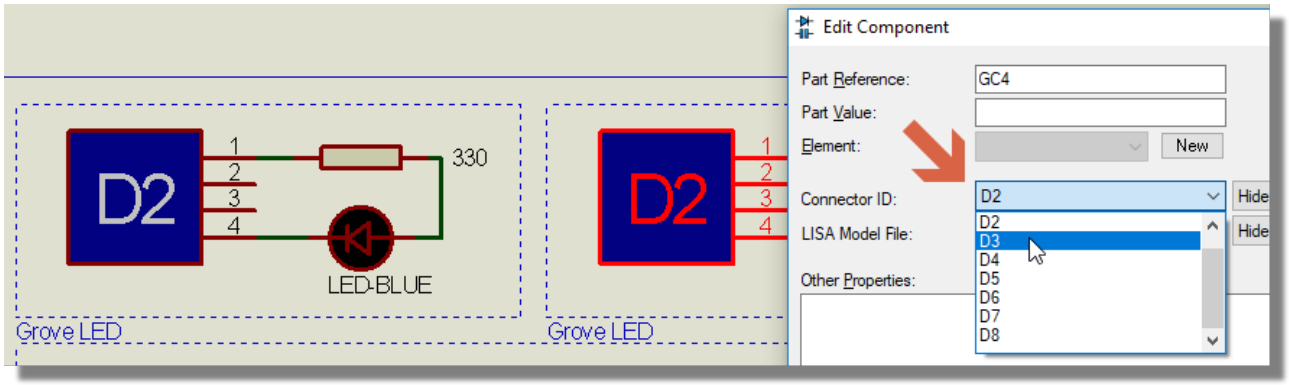
Potential Pitfalls

When you are working with Grove modules in Visual Designer the process is identical to working with complete hats. However, since the modules themselves plug into a hat, you need to take care not to re-use a connector on the Grove board. The real Grove base-board shield will look something like the following:



Grove Shield - the sensor modules plug-in to the exposed header pins.

When you add a Grove peripheral it will be assigned an ID on the schematic corresponding to a connector on the real board. If you get a duplicate, you'll need to edit and rename one of the connectors to a free slot.

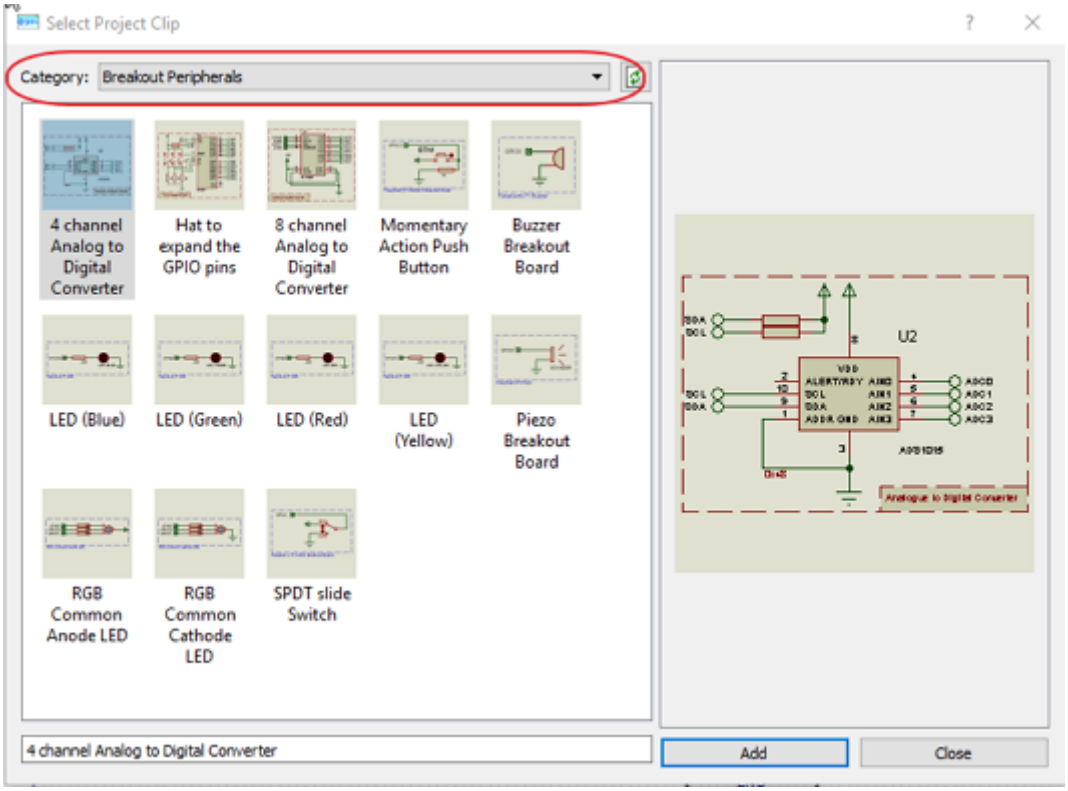


If two grove modules have the same connector, you need to edit one and assign to an unused header slot on the Grove Shield.

> ⚠ You need to make sure that your use of Grove modules is compatible with hardware. Since the pins used in each socket overlap it is quite easy to mistakenly use the same pin twice.

Breakout Boards and Peripherals

The third category of hardware that you will find in the Peripheral Gallery is breakout boards. These are useful and interesting pieces of electronics (often single components) for which we have provided control methods inside Visual Designer. So, when you add a breakout peripheral to your project, you can drive it from the methods it adds to the Project Tree.



Breakout boards provide even more flexibility on the schematic. Indeed, they often connect to other schematic components that you want to place and wire yourself.

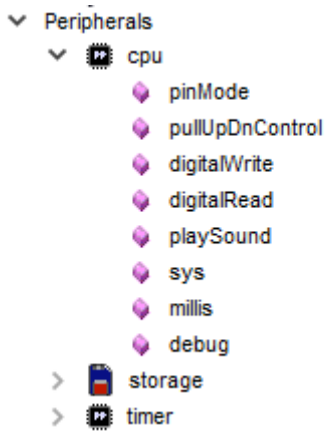
Potential Pitfalls

CPU resources and pin assignments need careful consideration.

Schematic Design

With Visual Designer you have access to the full power of the Proteus VSM simulation environment. This means that you have a professional schematic capture tool with tens of thousands of embedded components that you can place, wire and simulate.

This is the most flexible design method of all but involves a significantly higher learning curve. The programming in Visual Designer will have to be done via the CPU methods rather than specific methods for peripherals.



Potential Pitfalls

Schematic design for simulation is a large and relatively complex topic. In addition to the basic of placement and connectivity there will also be power considerations and you will need to program for interconnect protocols where appropriate. It should also be noted that the Raspberry Pi simulation model in Proteus is a high level behavioural model. Since the real rPi runs a full operating system it is quite easy to run Python code that is not supported in simulation. You'll find a list of supported libraries here.

DEBUGGING SKILLS

Overview

When developing your embedded system you may want to test your program as it evolves and you will certainly want to test it when it is complete. It is a rare occasion when everything works perfectly first time and you'll therefore also need to debug your program in order to find and fix errors.

Visual Designer makes this process extremely simple because it all happens in software. The following topics summarize the essential skills needed to simulate, debug and measure with Visual Designer. It's assumed at this stage that you are familiar with all the required editing skills and have a project ready for testing.

System Level Simulation

When you add peripherals and control them in your flowchart program you are building an embedded system. Loosely speaking, there will be the microcontroller board and some hardware pieces representing the peripherals. Running a system level simulation is the real power of Proteus because you are simulating the microcontroller together with all of the analog and digital electronics connected to it. This allows you to interact with ,test and debug your project entirely inside the Proteus software suite.

To start a simulation

To begin simulating either press the F12 shortcut key or the play button on the animation control panel. Your program will compile and simulation progress will be reported on the status bar.



When you start the simulation the time elapsed is shown on the status bar.

To stop a simulation

To end a simulation either press the ESCAPE shortcut key or the stop button on the animation control panel. The program will stop and Visual Designer will return to the Editing Layout



Stop the simulation via the Control Panel.

To pause a simulation

Pausing the Proteus system level simulation is an important concept; you are effectively stopping time. For example, your capacitors wont discharge and your motors will maintain their angular position and their momentum. This allows you to inspect both your program and your virtual hardware and when you single step debug the entire system will advance in time synchronously. Your embedded system works to the clock on the status bar, not the clock on the wall and you are in charge of the clock ! It's enormously powerful and we'll see many uses in both these topics and more practically in the tutorials.

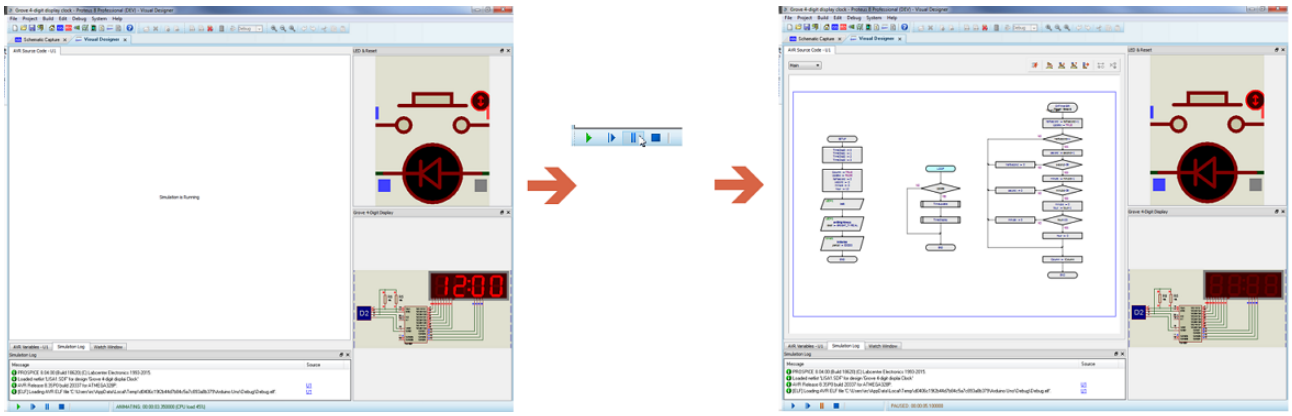
To pause a running simulation either press the PAUSE button on the keyboard or click the pause button on the animation control panel.



Pausing a Simulation

A simulation will also be automatically paused at a breakpoint. Breakpoints are discussed in the Debugging Skills topic below and demonstrated in the various tutorial topics.

Whenever the simulation is paused, Visual Designer will switch to full debugging layout and both the flowchart and the various debug windows will become available for inspection and interaction.



Pausing the Simulation prepares Visual Designer for debugging.

> You don't see the flowchart during free running simulation because the program is executing so quickly that actually showing it's current position would be impossible.

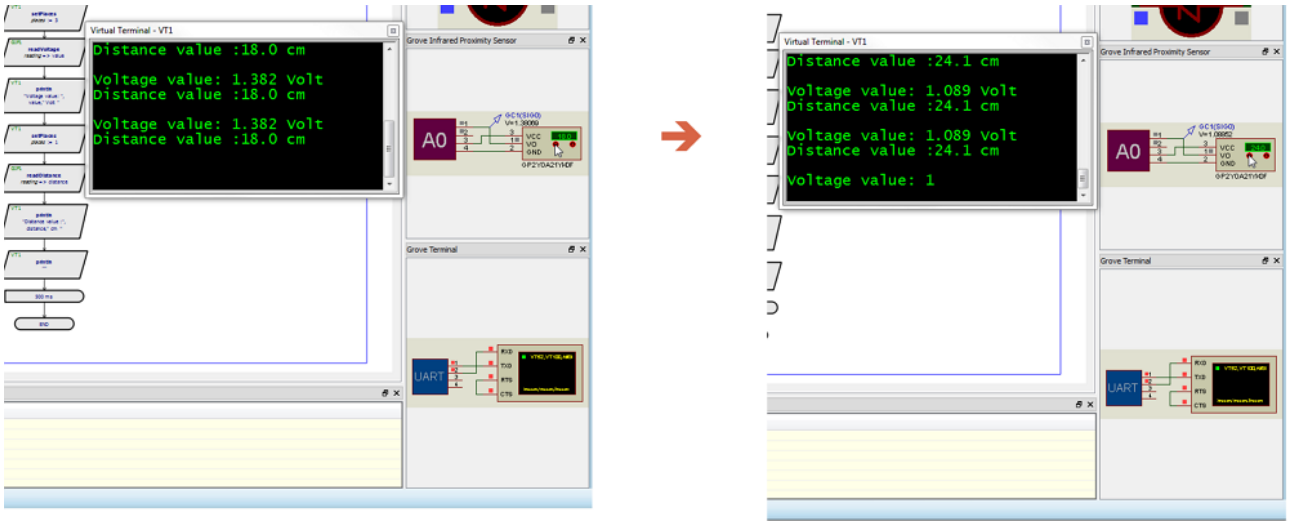
To interact with a simulation

When the simulation is running you will see Visual Designer in it's debugging layout, with all the interactive elements aligned on the right hand side of the application. These will almost certainly be a mix of indicators (led's, LCD's, etc.) that provide you with feedback and actuators (buttons, switches, keyboards, etc.) that you click to provide stimuli to the simulation. Here's an example where we adjust an infra-red proximity sensor:

1) Start the Simulation

2) Move the mouse over the adjustment arrows on the sensor.

3) Click up or down as required.



Clicking on the IR sensor adjustment controls mimics the behaviour of a physical test.

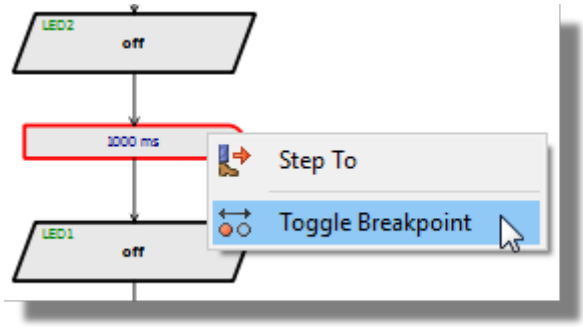
You can just as easily switch to the schematic tab and interact on that screen if you prefer. However, it's handy to have everything in one place and particularly so during a debugging session when you want to view hardware while single stepping your flowchart.

> The virtual terminal is slightly different as you can simply click and type when the simulation is running. This means you can use the terminal as a command interface to your program which has many, many uses.

Debugging Skills

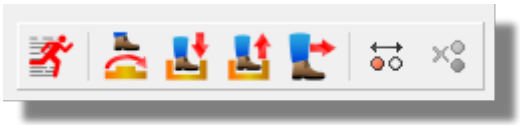
To set a software breakpoint

Right click on the flowchart block either before simulation or when the simulation is paused and select the toggle breakpoint command from the context menu.



To single step debug your flowchart

When the simulation is paused, use the step commands at the top of the window or the F10 and F11 shortcut keys to single step through your flowcode.



> To watch the program flow it is often helpful to try a simulation run by pressing pause and then the animated single step command from the Debug Menu. This is slow enough that you can see the path your program executes through decision blocks and is often a quick way to spot a problem.

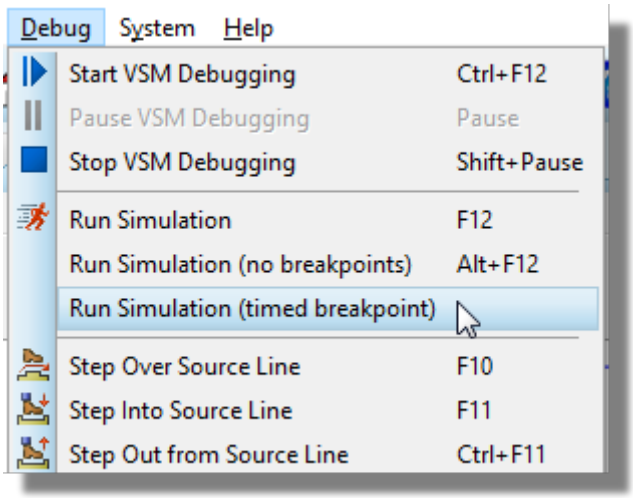
To step to a block on the flowchart

To step to a specific block on the flowchart, first pause the simulation, then right click on the block and select the Step to command from the context menu.



To set a timed breakpoint

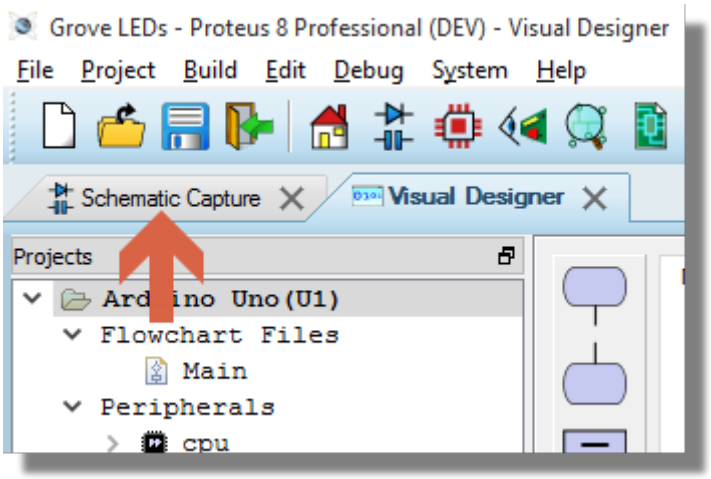
It's occasionally useful to pause a simulation after a period of time rather than on a particular flowblock. You can do this by selecting the Run Simulation with Timed Breakpoint command from the Debug Menu.



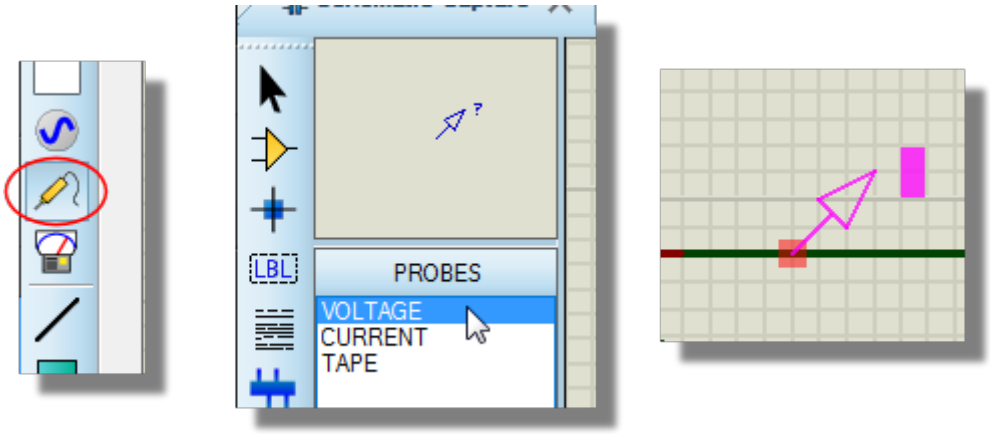
To set a hardware breakpoint

You can also set a breakpoint on a hardware condition as opposed to a software condition. The procedure is:

1) With the simulation stopped switch to the schematic capture tab.



2) Place a voltage probe on the wire of interest.



> Terminals with the same name are considered connected by an invisible wire. It is a simple way to keep shields and breakouts self-contained and not have wires criss-crossing all over the screen.

3) Edit the probe and configure the Real Time Breakpoint settings.

4) Run the simulation

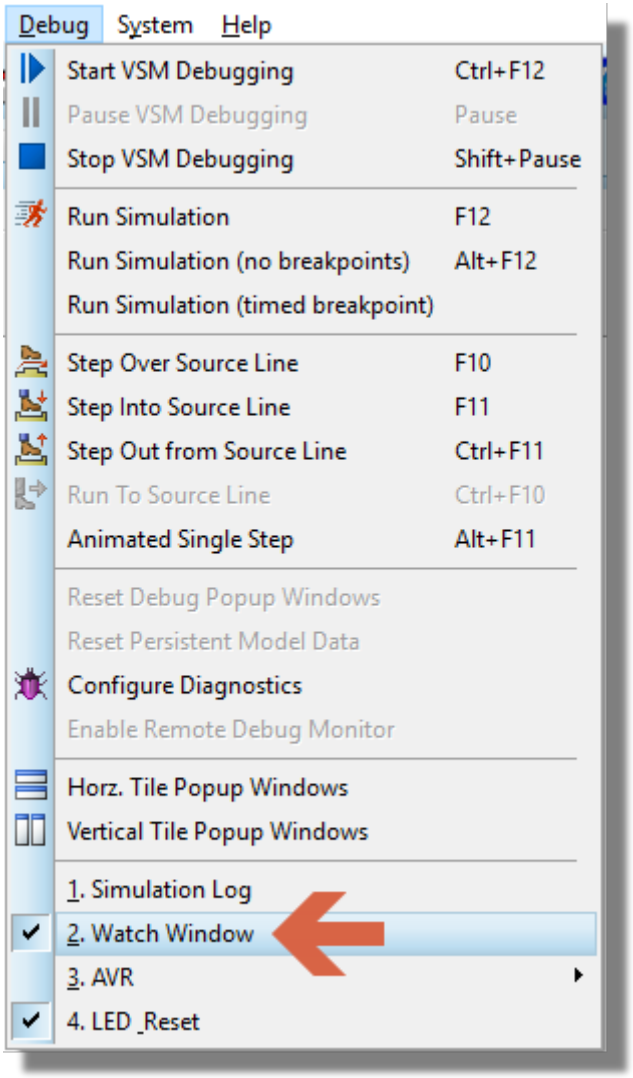
See Also:

- Hardware Breakpoints (Proteus VSM Reference Manual)

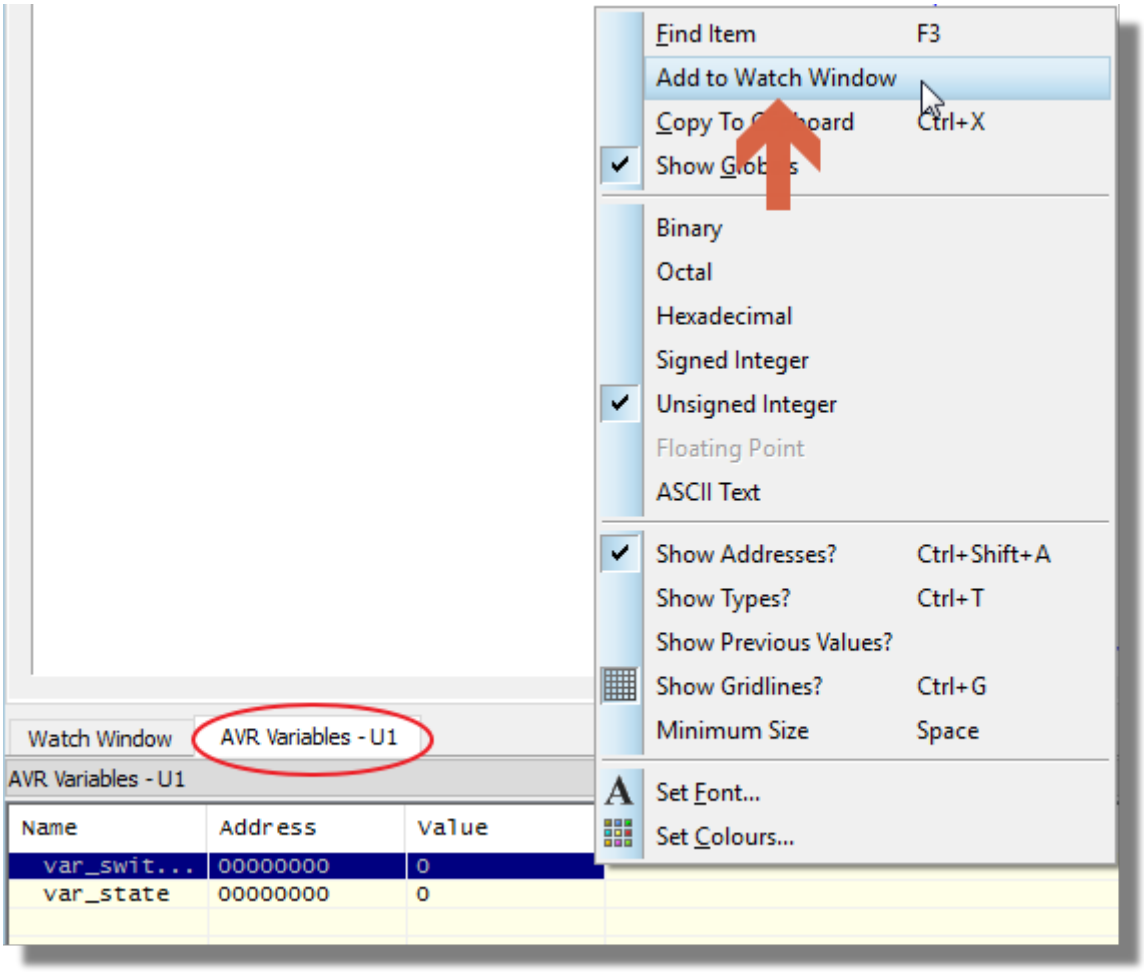
To add a variable to the watch window

The watch window is the only debugging window available during a free running simulation. You may therefore choose to add a program variable to the watch window in order to monitor the value as the program executes. To do this:

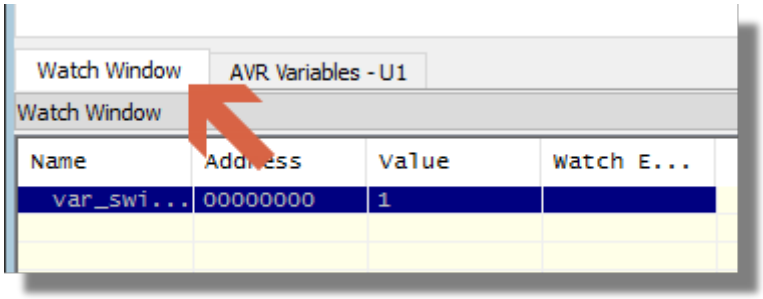
1) With the simulation paused, open the watch window from the Debug Menu. By default, the window will dock beside the variables window.



2) Switch to the variables window, right click on the variable of interest and select the 'Add to Watch Window' command.



3) Switch back to the Watch Window and run the simulation.

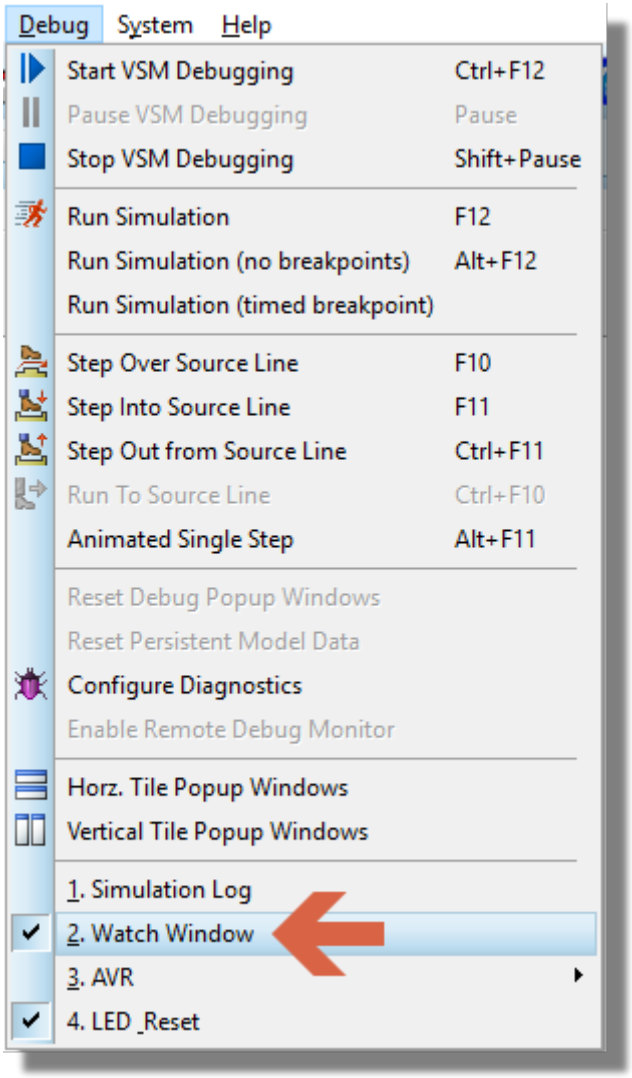


⚠ Variables are identified in the Watch Window by address. If you change your program between simulation runs there is a chance that the compiler will assign a different address to the variable and you will not be monitoring the variable on subsequent simulation runs. This can be extremely confusing and it's best to remove and re-add the variable when debugging after programming changes.

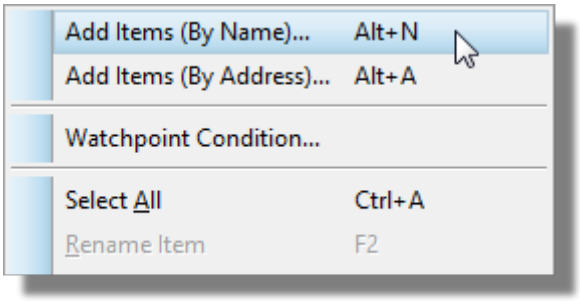
To add other items to the watch window

The watch window is simply a monitor on memory so you can add AVR registers or other program locations. For example, to add a register:

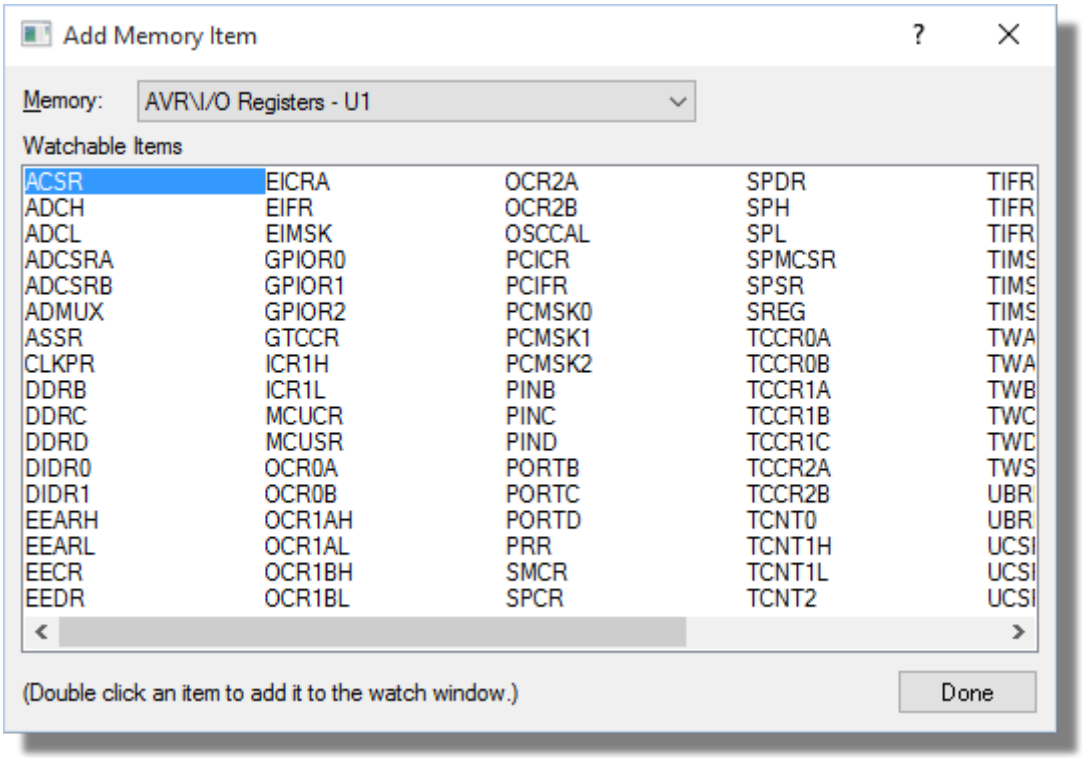
1) With the simulation paused make sure the Watch Window is open from the Debug Menu.



2) Right click on the Watch Window and select add item by name.



3) Select the register you want to watch and add it.



The process is very similar when adding memory locations by address.

See Also:

The Watch Window

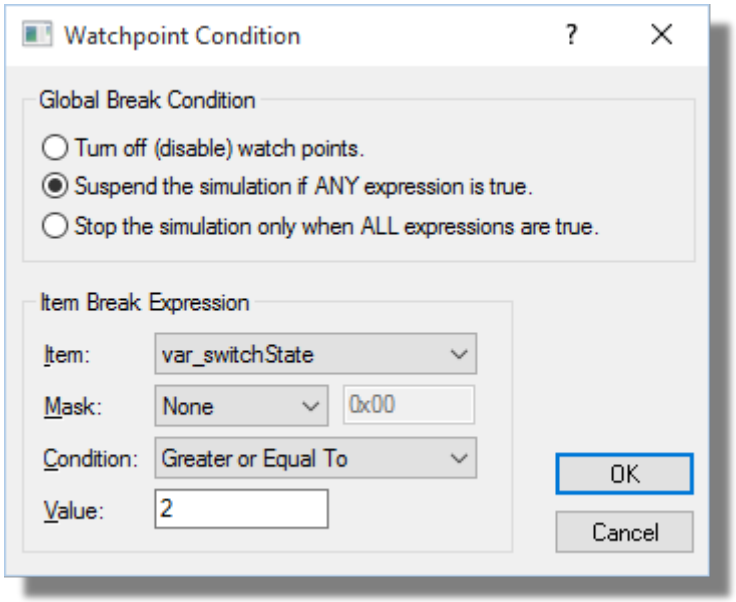
To set a conditional breakpoint

A conditional breakpoint (a watchpoint) is where you set a breakpoint to occur when a preset condition is met on an item in the watch window. It is often under used as a debug tool - for example, it is the perfect way to pause the simulation every time a timer overflows. To set a watchpoint condition:

1) Use the methods described above to add either a variable or a register to the watch window.

2) Right click on the item in the Watch Window and select Watchpoint Condition from the context menu.

3) Configure as required.



4) Run the simulation.

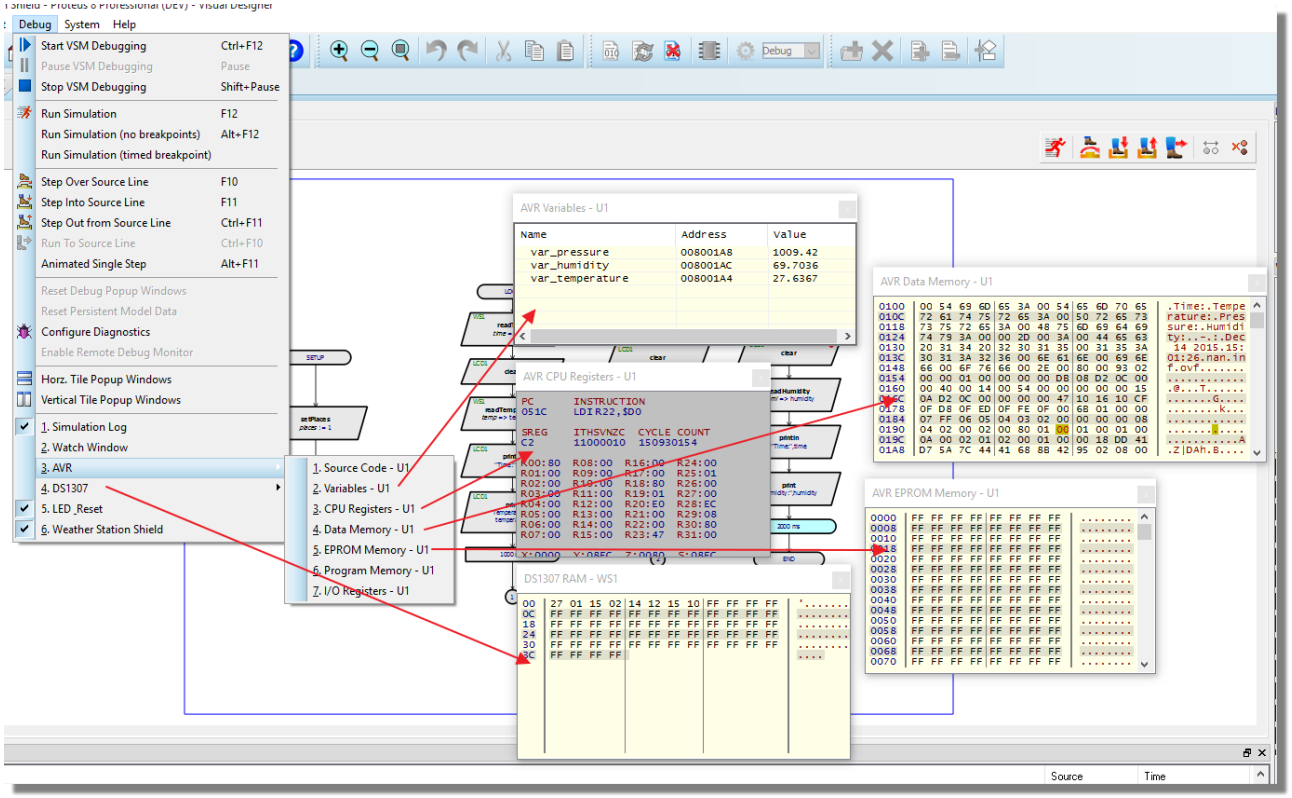
There is an option on the dialogue to specify whether the watchpoint is a Global Break Condition. This just determines whether the simulation suspends when any item expression is true, or only when all watchpoints are true.

To open a debug window

In addition to the source, variables and watch windows there are a host of additional debug windows that we encourage users to explore. After all, the more familiar you are with the available tools the better at finding problems you will become and the debugging toolset in Proteus is unparalleled. Often, you can even open debug windows on the shield peripheral models as well as the processor model. The process is:

1) Pause the simulation.

2) Launch the debug window you want from the Debug Menu.



If you find yourself with a very busy screen you can close debug windows and re-open them when needed from the Debug Window. Additionally, with the exception of the Watch Window, they will all disappear during free running simulation.

RESOURCE FILES & DATA STORAGE

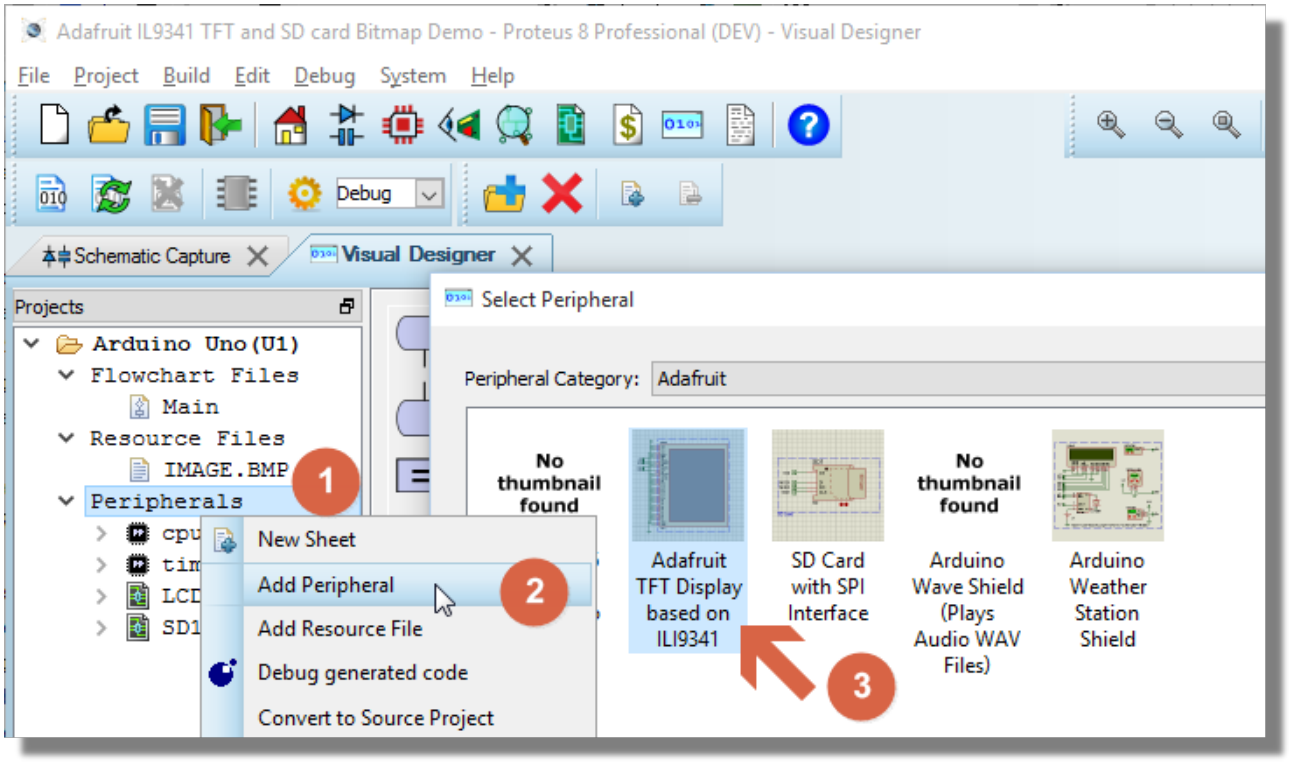
Introduction

Visual Designer includes a very powerful yet simple storage model. This takes the form of resource files which can be added to the current project directly from the Project menu or via the context menu in the Project Tree. Resource files can be anything from a text file to a picture to an audio clip.

How it Works

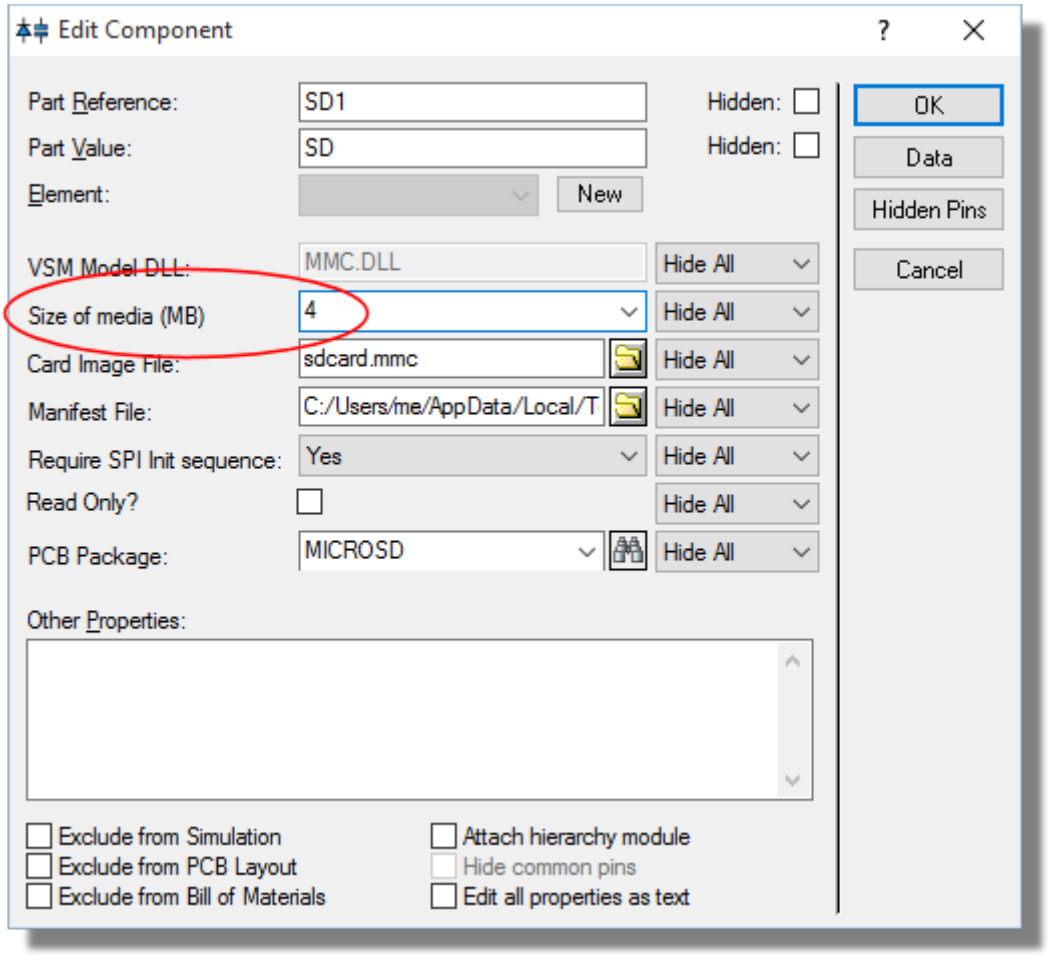
It is very common to have a shield containing both an SD Card and other electronics for using the contents of the SD Card. Obvious examples include the TFT shield which can render pictures stored on the SD card and the Wave shield which can play WAV files stored on the SD Card. Like everything else in Visual Designer the process for working with these shields is fairly straightforward:

1) Add the shield containing the resource file (e.g. Wave shield, TFT Shield)



1, right click and on Peripherals, 2, Add Peripheral, 3, Select Peripheral.

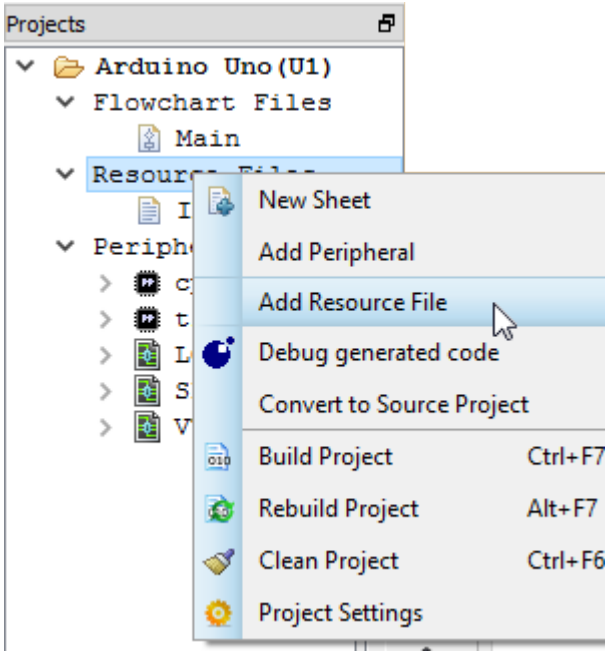
2) Check that the SD Card image size is big enough for your intended purposes. Clearly, the SD Card image size for wave files needs to be larger than that for pictures.



Change the size to cope with what you are loading. i.e. a WAV file will be larger than an image file.

In software, Proteus works some magic here and will create a FAT image for you of the correct size. If you are programming the real hardware you'll need to make sure you have a FAT16 or FAT32 SD Card of a suitable size in the SD Card slot of the shield before you hit the program button !

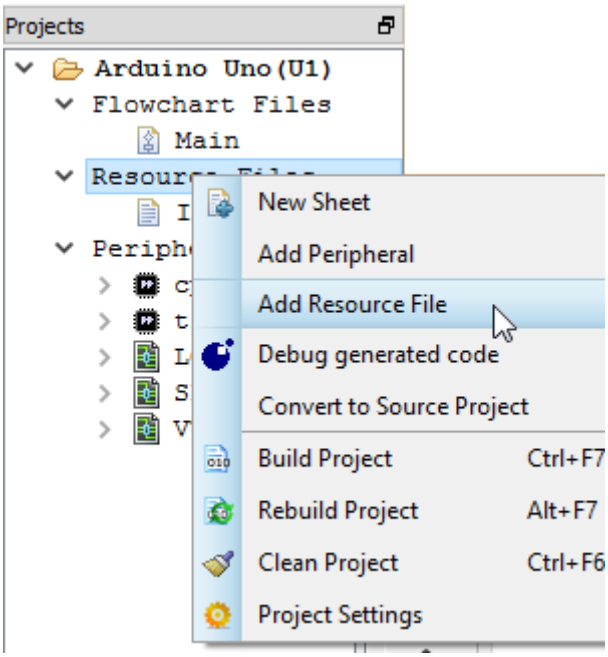
3) Add the resource file(s)



Be careful with the naming of resource files. The Arduino SD stack by default supports only 8.3 filenames in order to support FAT16 cards. This means that the namestem must be 8 characters or less and the extension must be 3 characters or less. For example, a picture called ProteusLogo.bmp will not load whereas renaming it to logo.bmp or image.bmp will work fine.

4) Drag and drop the resource file into your program. Visual Designer will automatically:

- Detect a destination IO routine for the resource (e.g. DrawBitmap() or Play())
- Set the source for the routine to be the resource.



See Also:

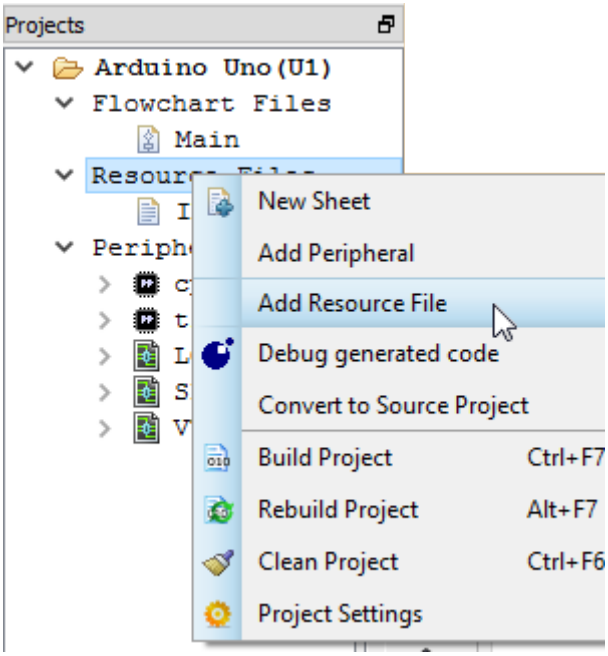
Tutorial 3: SD Card and TFT

If you add a resource file there must be either an SD Card in an included shield or an SD Card breakout peripheral otherwise you have no place to store the resource in hardware !

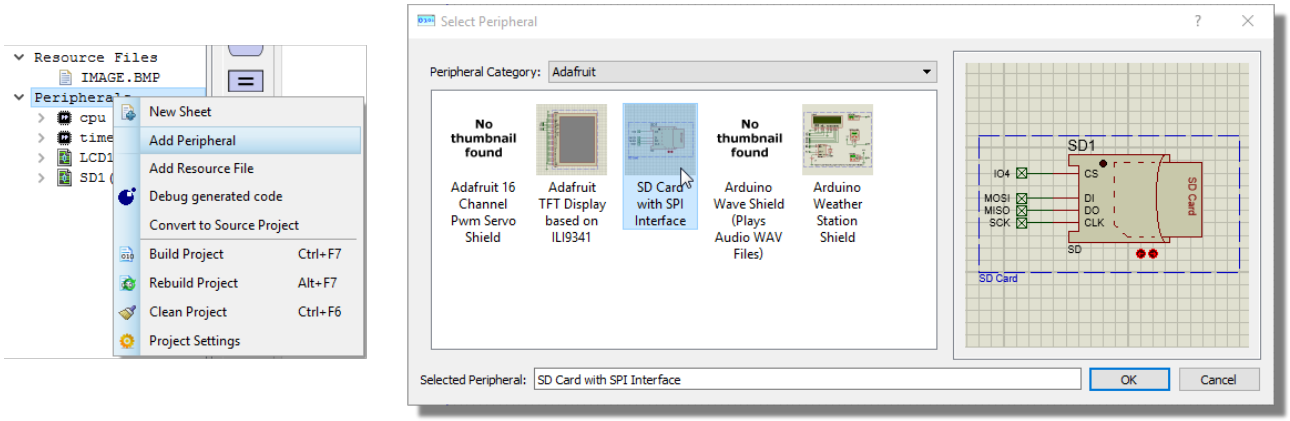
The Filestore

Sometimes you just need to store and manipulate data in your program. The filestore gives you this capability. To use the filestore:

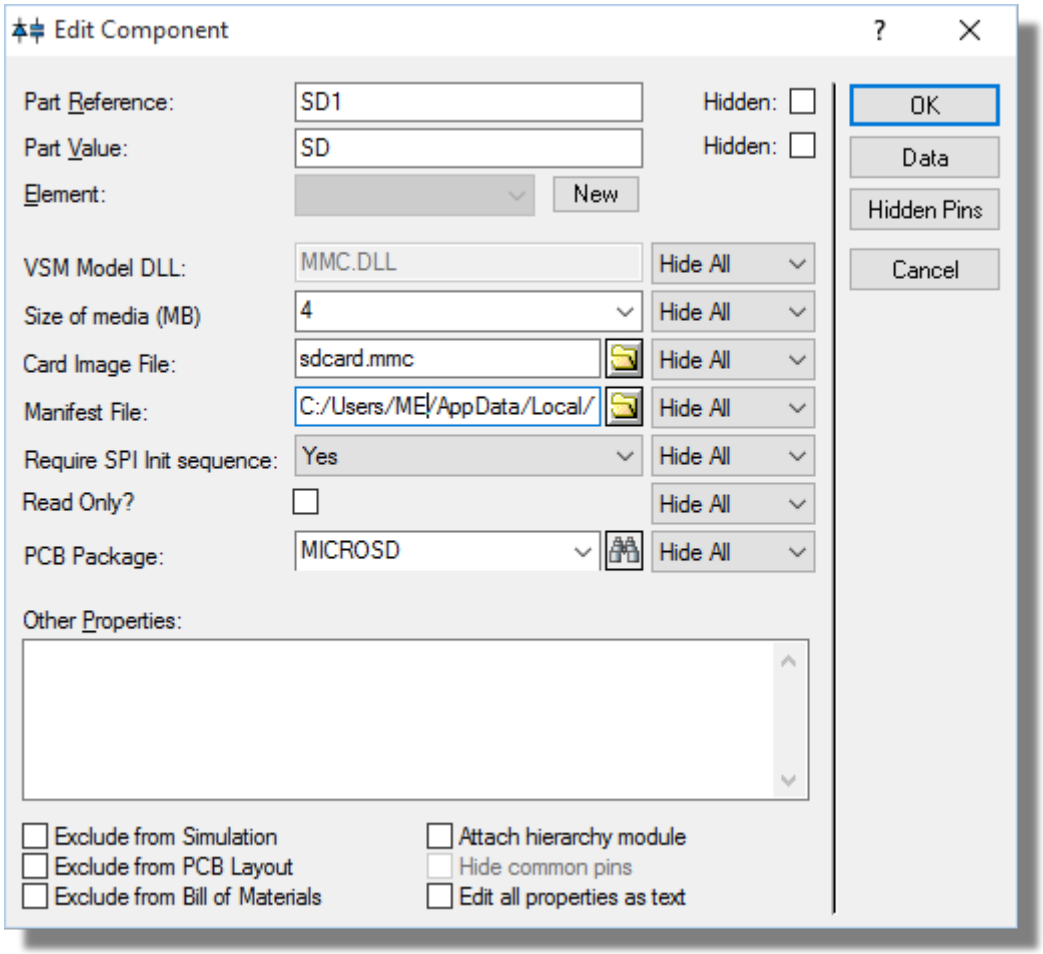
1) Add the resource(s). This is the data being stored (e.g. text file).



2) Add the SD Card peripheral. This is the filestore.



3) Use the methods on the filestore (storage data blocks) to open, read and otherwise manipulate the data in your program.



When you program the physical hardware the programmer will handle the transfer of the resources to the SD Card (as long as there is a suitable SD card in the hardware!).

ARDUINO HARDWARE

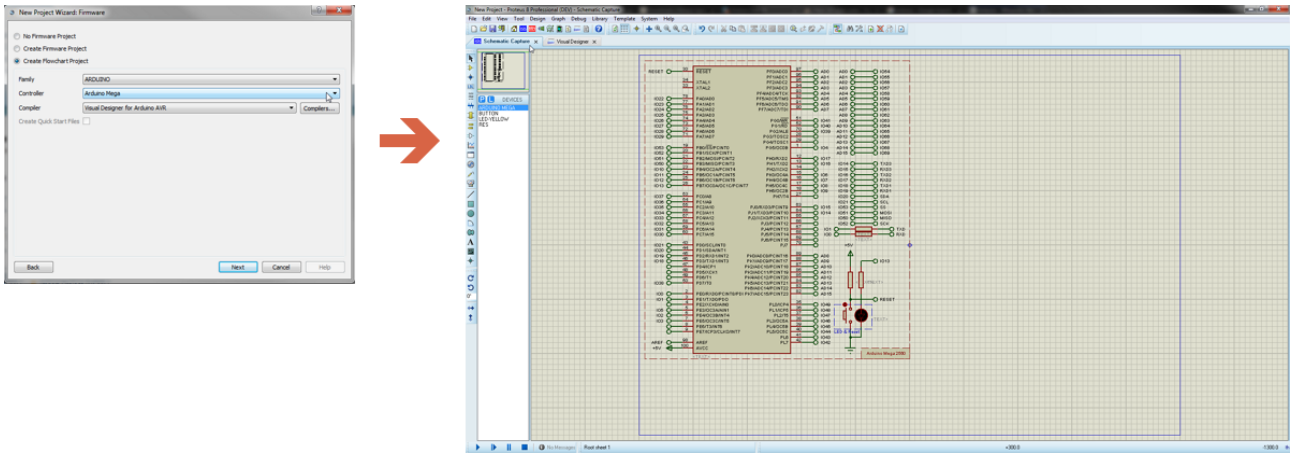
Introduction

Designing 'virtual hardware' for your project is greatly simplified in Visual Designer. The various options available to you are described in detail below.

Arduino Base Board

In hardware, Arduino is a system consisting of a fairly simple base board with an AVR microcontroller, some supporting electronics and Arduino pin headers (connectors). There are then a wealth of peripheral boards (shields) which plug into the pin headers to form the hardware for a particular project.

In Visual Designer for Arduino, the creation of the project is the point at which you make your decision on the base board (E.g. Arduino Uno, Arduino Mega). This is automatically placed on the schematic when the project is created.



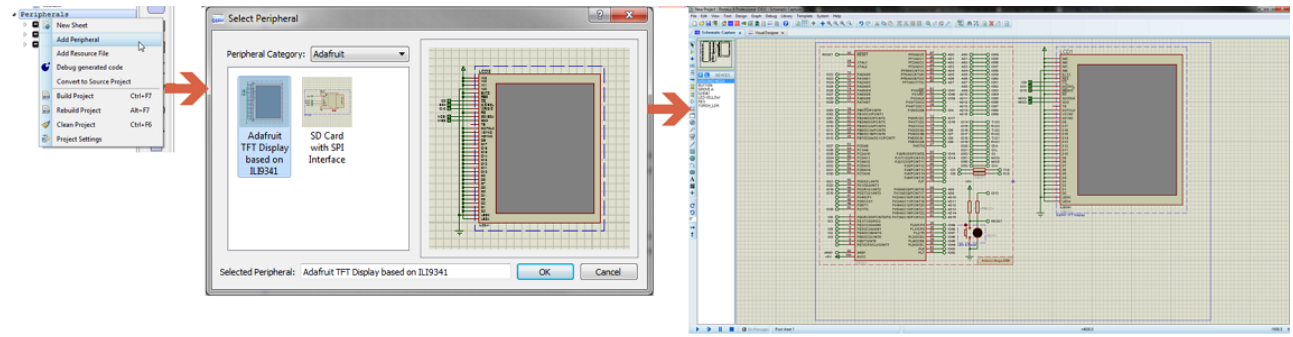
Automatic placement of the Arduino Mega on the Schematic when the Project is created.

Arduino Shields

Arduino shields are modular circuit boards that piggyback onto your Arduino base-board to instill it with extra functionality. They are the simplest way to add hardware functionality to your project because the circuitry and connectivity on the schematic happens automatically.

Adding a Shield

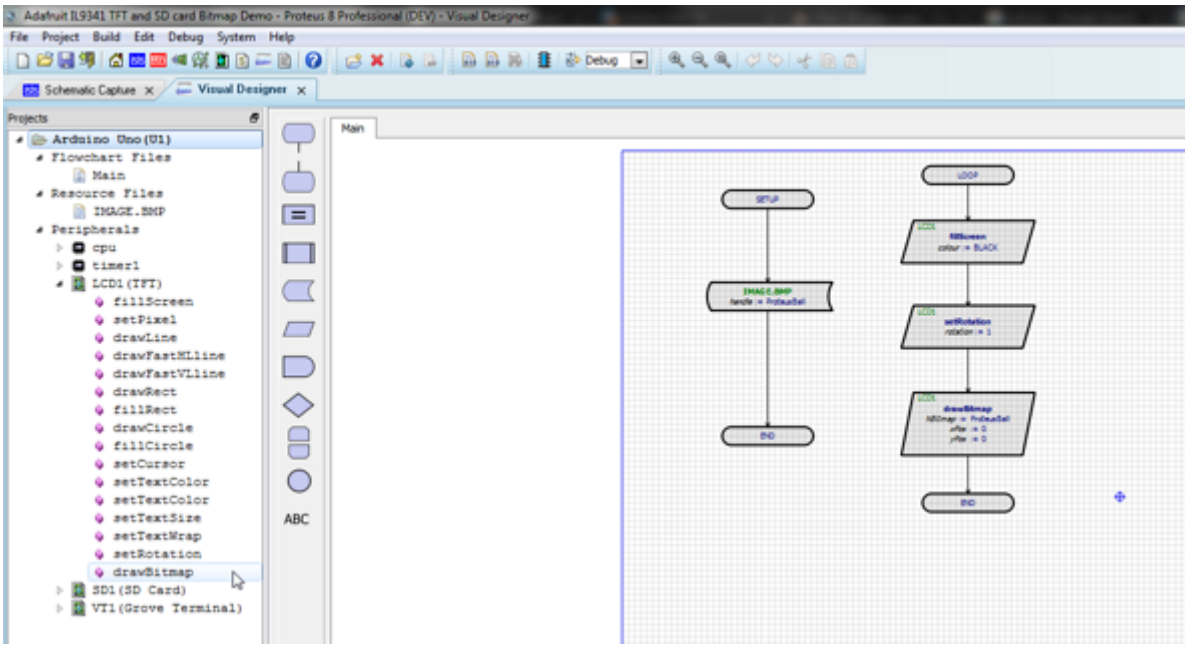
When you then want to add a shield in Visual Designer you right click on the Project Tree and select 'Add Peripheral' from the resulting context menu. Select the 'Adafruit' category on the peripheral browser and then select the shield you want to add. This will auto-place the shield electronics on the schematic.



Adding a Peripheral will auto-place and auto-connect on the schematic.

Writing a Program to Control a Shield

Shields appear in the project tree in Visual Designer once picked from the Peripheral Gallery. They include high level methods that allow you to quickly control the connected electronics via flowchart blocks.



Drag and Drop methods make it simple to drive the shields.

Potential Pitfalls

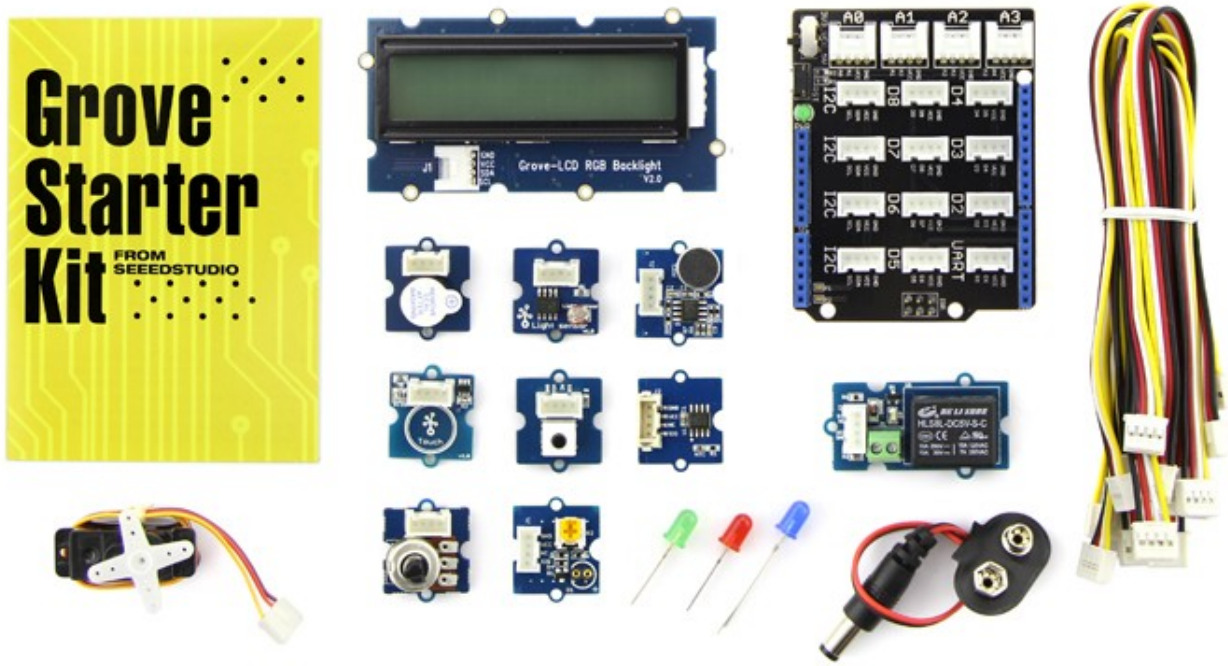
Working with Viirtual Shields is the simplest method to quickly create projects in Visual Designer. You do however still have to take care with what hardware you choose to use. For example:

- You need to make sure that your use of Arduino shields is compatible with hardware. For example, some shields are incompatible because they make use of the same resource on the CPU (e.g. Timer1). Others are pin incompatible.
- You need to make sure that your target Arduino base board has enough memory to handle the shields you are adding to it. In particular, with an Arduino Uno you can run out of memory if you add shields with large and complex software stacks.

Grove Sensors and Modules

Grove is a modular electronic platform for quick prototyping. Every module has one function, such as touch sensing, creating audio effect and so on. You simply plug the modules you need into the base shield and then connect the base shield to the Arduino base board in the normal way.

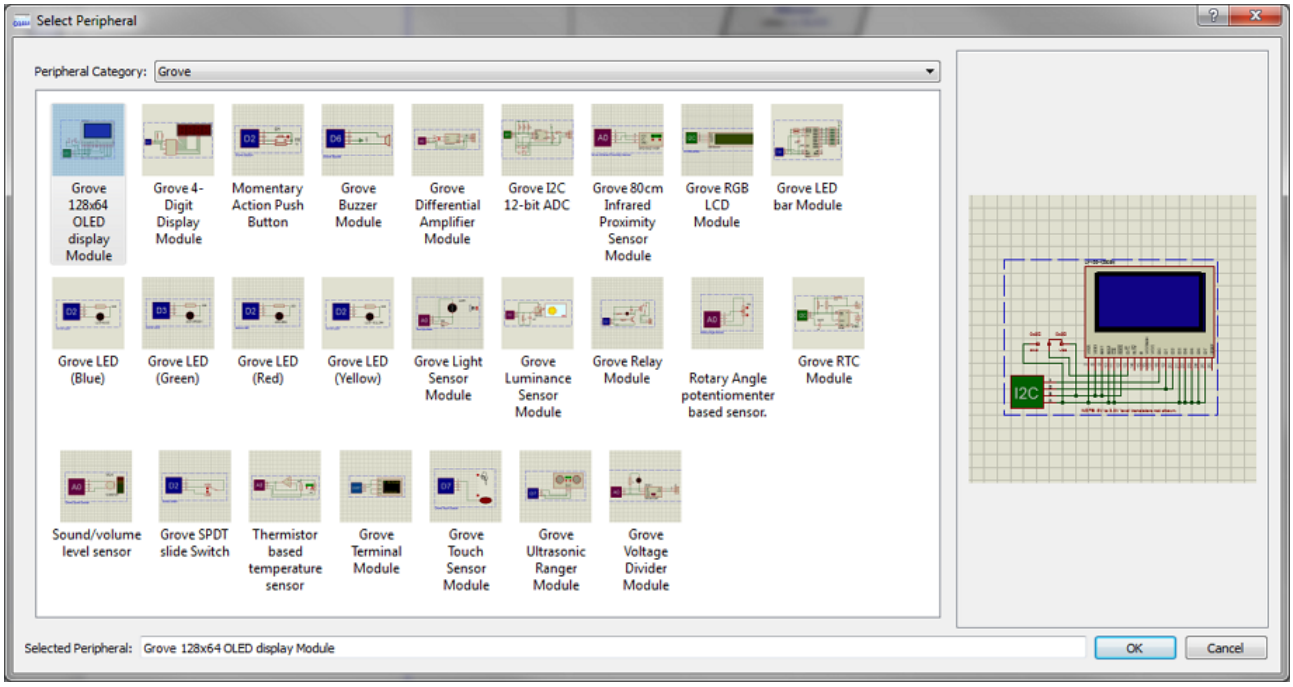
The Grove Starter Kit Plus is a great way for beginners and students to get started with Arduino. The Grove Base Shield drops onto your existing Arduino and lets you connect up to 16 Grove modules without any fuss. The Arduino headers are still broken out on the top of the shield making it very easy to connect other breakout boards and sensors to your Arduino.



Grove Starter Kit Hardware.

Adding a Grove Module

You can add a Grove module in the same way as adding a shield, except that you select the Grove category in the Peripheral browser.

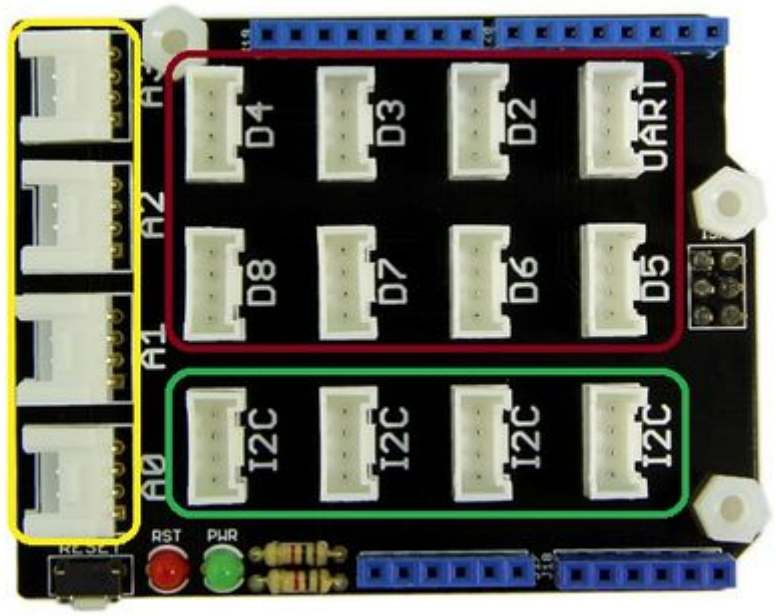


Adding Grove Starter Kit modules in Visual Designer.

You can then control the Grove peripherals via the high level methods available from the Project Tree. Multiple peripherals and sensors can easily be added to a single project which provides an extra level of flexibility over using standard Arduino shields.

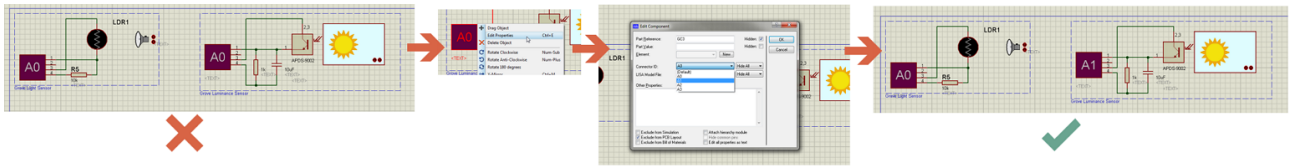
Potential Pitfalls

When you are working with Grove modules in Visual Designer the process is identical to working with complete Arduino shields. However, since the modules themselves plug into an Arduino shield, you need to take care not to re-use a connector on the Grove board. The real Grove base-board shield will look something like the following:



Grove Shield - the sensor modules plug-in to the exposed header pins.

When you add a Grove peripheral it will be assigned an ID on the schematic corresponding to a connector on the real board. If you get a duplicate, you'll need to edit and rename one of the connectors to a free slot.

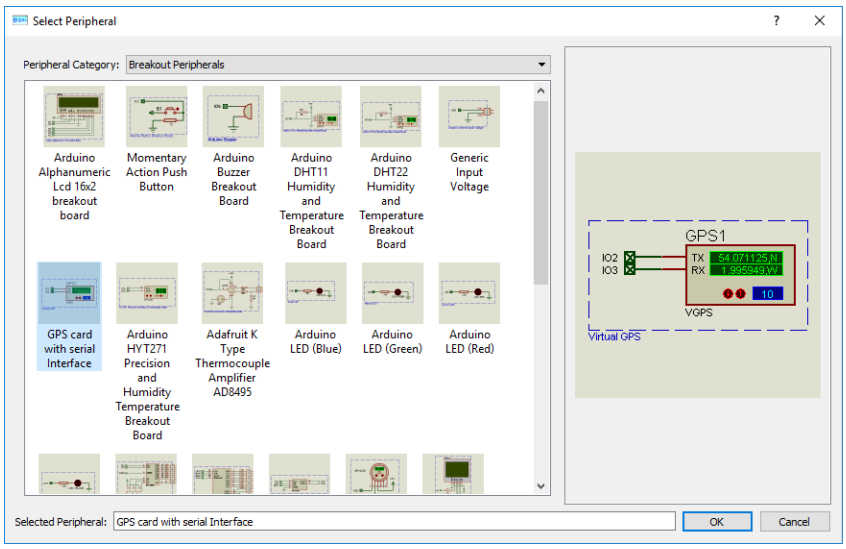


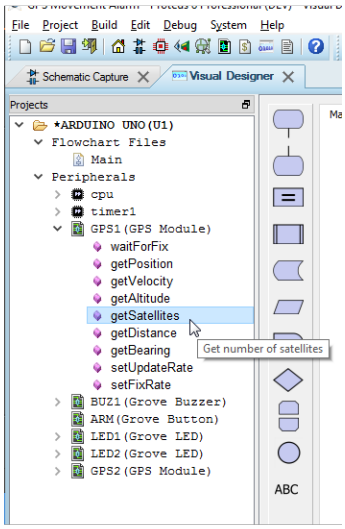
If two grove modules have the same connector, you need to edit one and assign to an unused header slot on the Grove Shield.

> ⚠ You need to make sure that your use of Grove modules is compatible with hardware. Since the pins used in each socket overlap it is quite easy to mistakenly use the same pin twice. A good example of this is discussed in Tutorial 2

Breakout Boards and Peripherals

The third category of hardware that you will find in the Peripheral Gallery is breakout boards. These are useful and interesting pieces of electronics (often single components) for which we have provided control methods inside Visual Designer. So, when you add a breakout peripheral to your project, you can drive it from the methods it adds to the Project Tree.





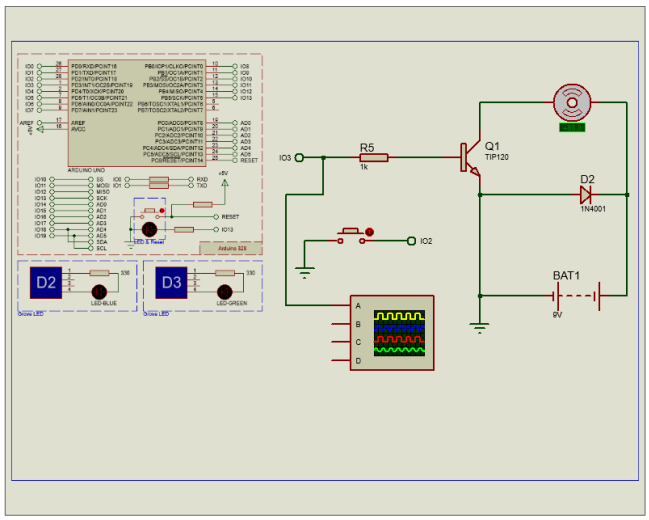
Breakout boards provide even more flexibility on the schematic. Indeed, they often connect to other schematic components that you want to place and wire yourself.

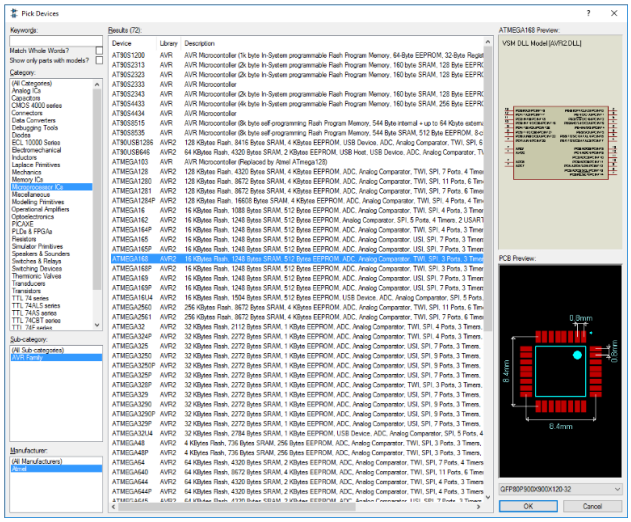
Potential Pitfalls

CPU resources and pin assignments need careful consideration.

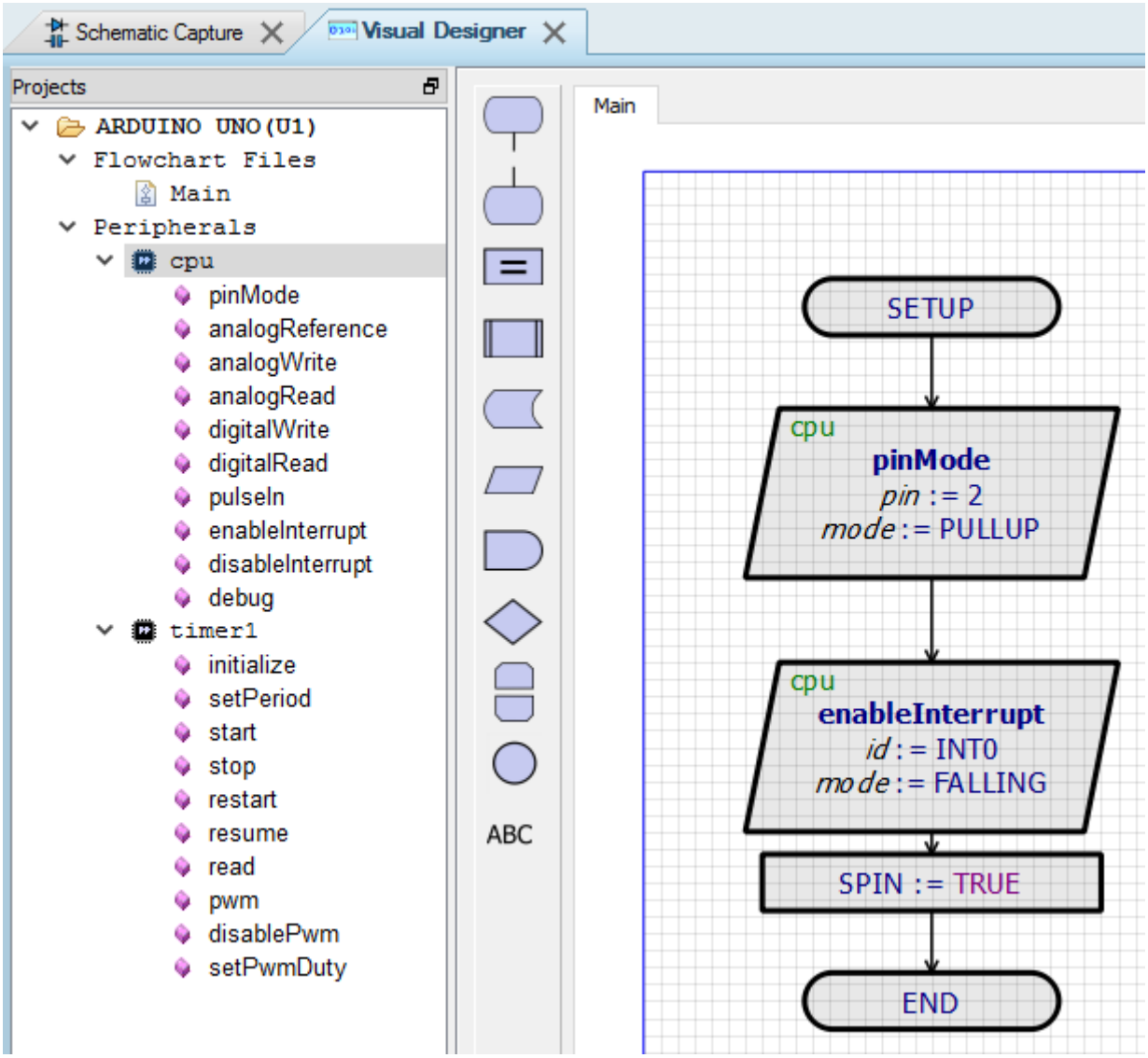
Schematic Design

With Visual Designer you have access to the full power of the Proteus VSM simulation environment. This means that you have a professional schematic capture tool with tens of thousands of embedded components that you can place, wire and simulate.





This is the most flexible design method of all but involves a significantly higher learning curve. The programming in Visual Designer will have to be done via the CPU methods rather than specific methods for peripherals.



See Also :

Tutorial 5: Introducing Schematic Design

Potential Pitfalls

Schematic design for simulation is a large and relatively complex topic. In addition to the basic of placement and connectivity there will also be power considerations and you will need to program for interconnect protocols where appropriate. At this stage, the transition from Visual Designer to Proteus VSM should be considered as there is a point beyond which C programming is required to effectively design the embedded system.



Introduction

This topic discusses how to program the Arduino hardware from inside Proteus Visual Designer so that you can see your projects working in the real world.

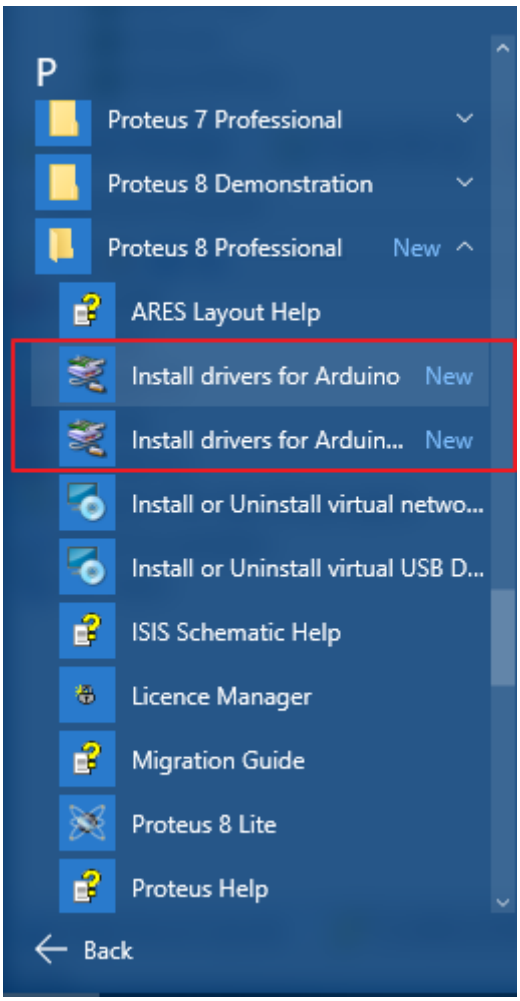
For technical support or problems with programming please refer to either the troubleshooting guide below or to the following Arduino resources:

- https://www.arduino.cc/en/Main/FAQ
- https://forum.arduino.cc
- https://forum.arduino.cc/index.php?topic=261445.0

A Quick Guide to Programming

For convenience, Proteus Visual Designer includes an interface to the Arduino AVR programmer. To program your hardware from inside Proteus you will need to :

1. Make sure you have the Arduino drivers installed. If you don't you can install them from the Labcenter Program Group or by installing the Arduino IDE from their website.



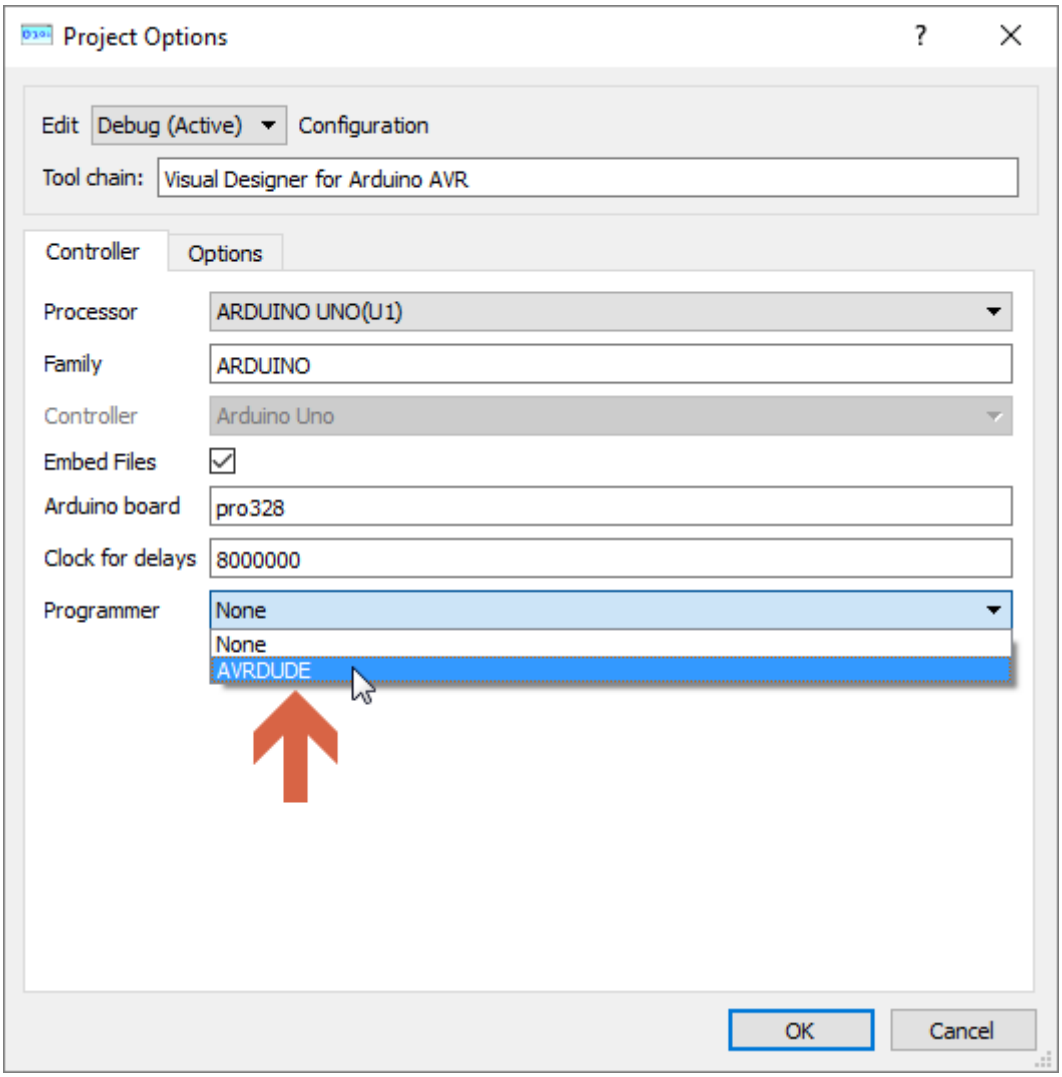
Installing Drivers from the Labcenter Program Group in the Start Menu.

2. Go to the Project Settings dialogue



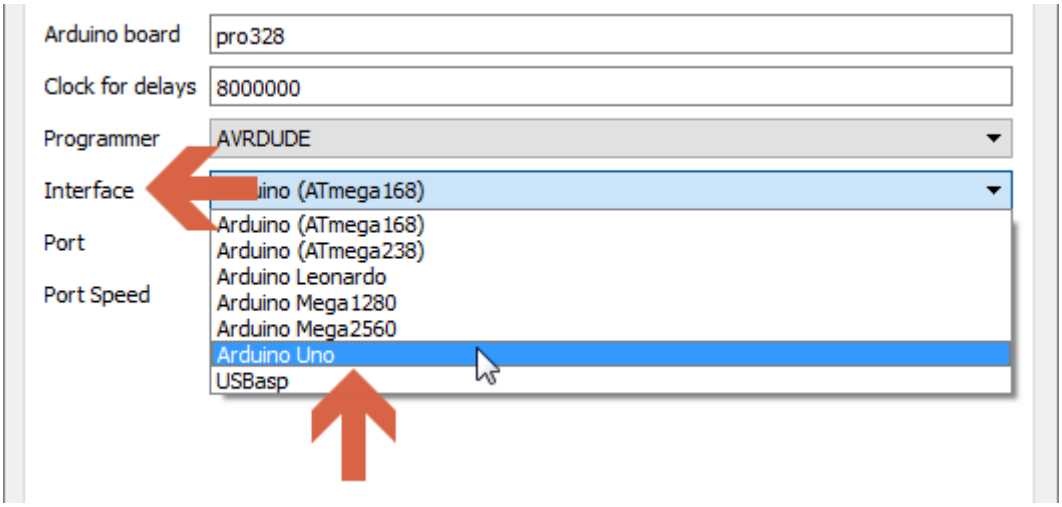
Launching the Project Settings Dialog in Visual Designer

3. Specify the programmer (AVRDUDE) and the board (eg. UNO)



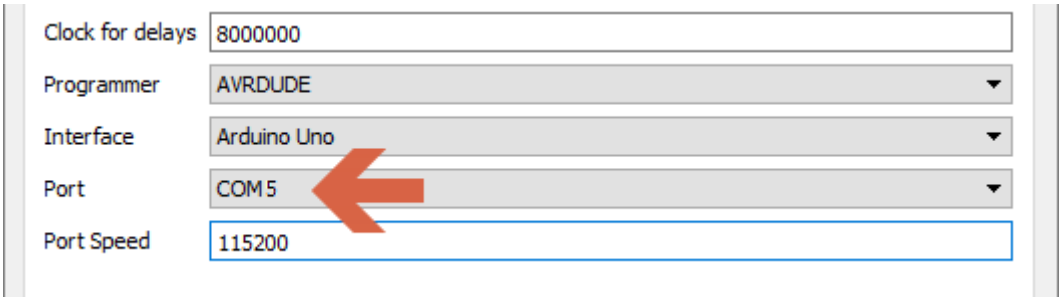
Specify the Programmer.

4. Specify the correct Interface based on the physical hardware you have connected.



Specifying the Interface.

5. Plug in the hardware, check device manager and enter the COM Port that your PC is using for the USB here. Refer to the instructions below if you are unsure what to select



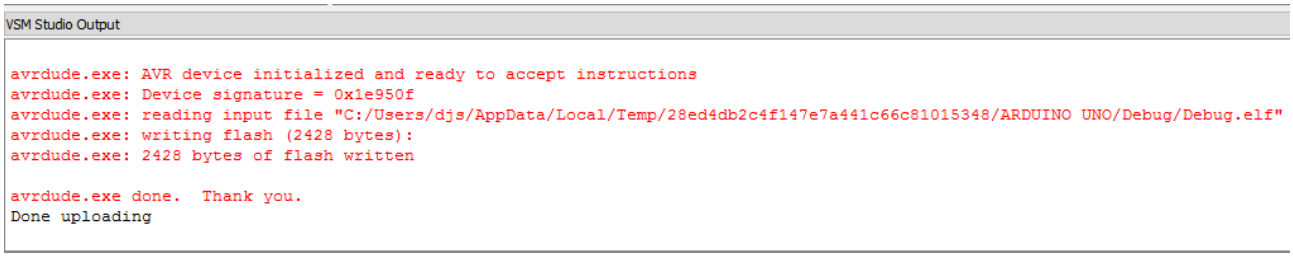
Set the Correct COM Port.

6. Compile your program and press the program button to transfer to the real hardware



Build and Program the Hardware.

You should receive the following message:



Completed Upload to Hardware.

The Controller Dialog

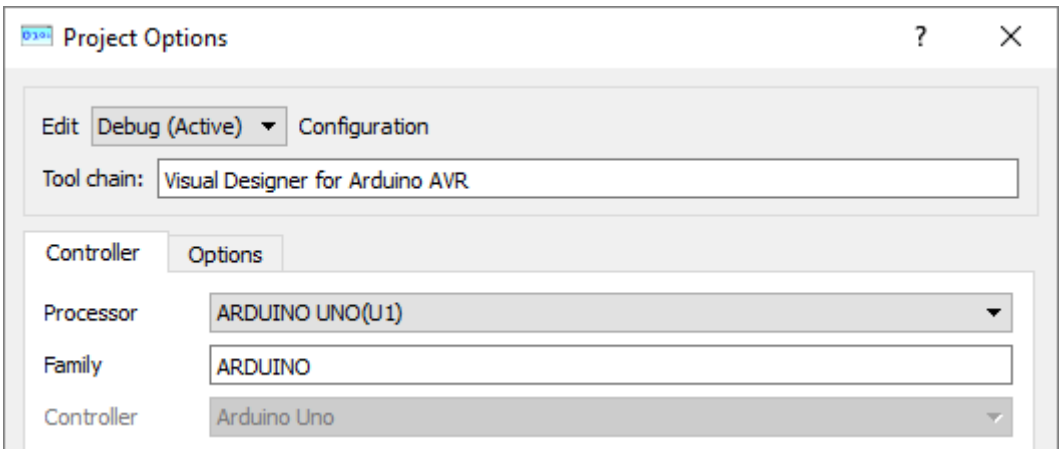
Apart from the basic programming options there are a number of additional options on the dialogue form which are discussed in full below.



Launch the Project Settings Dialogue Form.

Processor, Family and Controller

The Processor, Family and Controller fields are set when you create the project and will be automatically populated for you in this dialogue form.



Processor, Family and Controller Options are preset when you create the project.

Embed Files Option

The Embed Files Option keeps all source and debug files inside the .PDSPRJ file container. This avoids common problems with relative paths to libraries and files as well as making the project easily portable. We do not recommend turning this option off without good reason.



Arduino Board Option

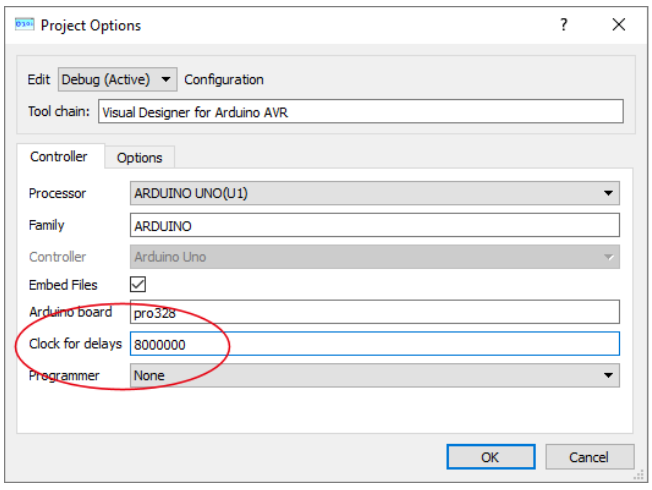
The Arduino board option is a string which is passed to, and used by the Arduino compiler and linker to identify the processor on the board. Do not change this option unless you are confident that you know what you are doing.

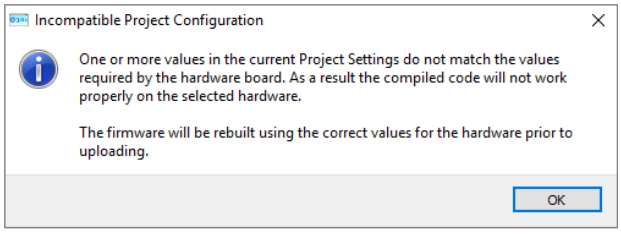


Clock for Delays Option

The clock for delays option is basically the processor clock frequency. For simulation, this is halved to 8MHz to ensure good performance even on modest computers. For the real hardware it needs to be changed to 16MHz but there are two ways to do this.

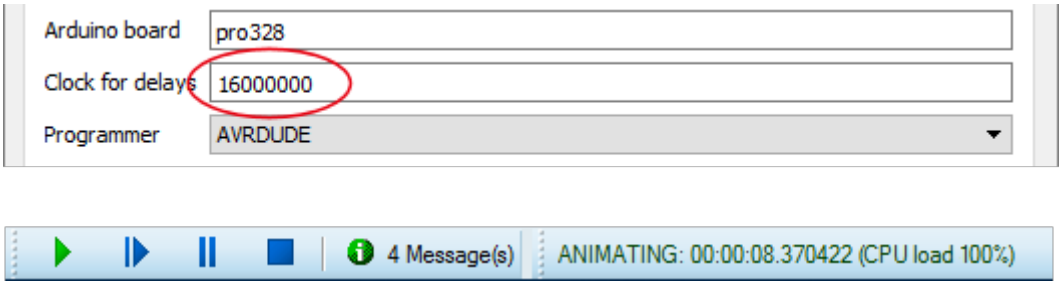
We recommend that you leave the value in the field alone. Each time you upload you will get a message explaining the discrepancy and Visual Designer will then program the real board with the correct 16MHz clock. This option give the best of both worlds as you will simulate with an 8MHz clock and program with a 16MHz clock.





Leave the clock option alone for best simulation performance while maintaining correct hardware compilation.

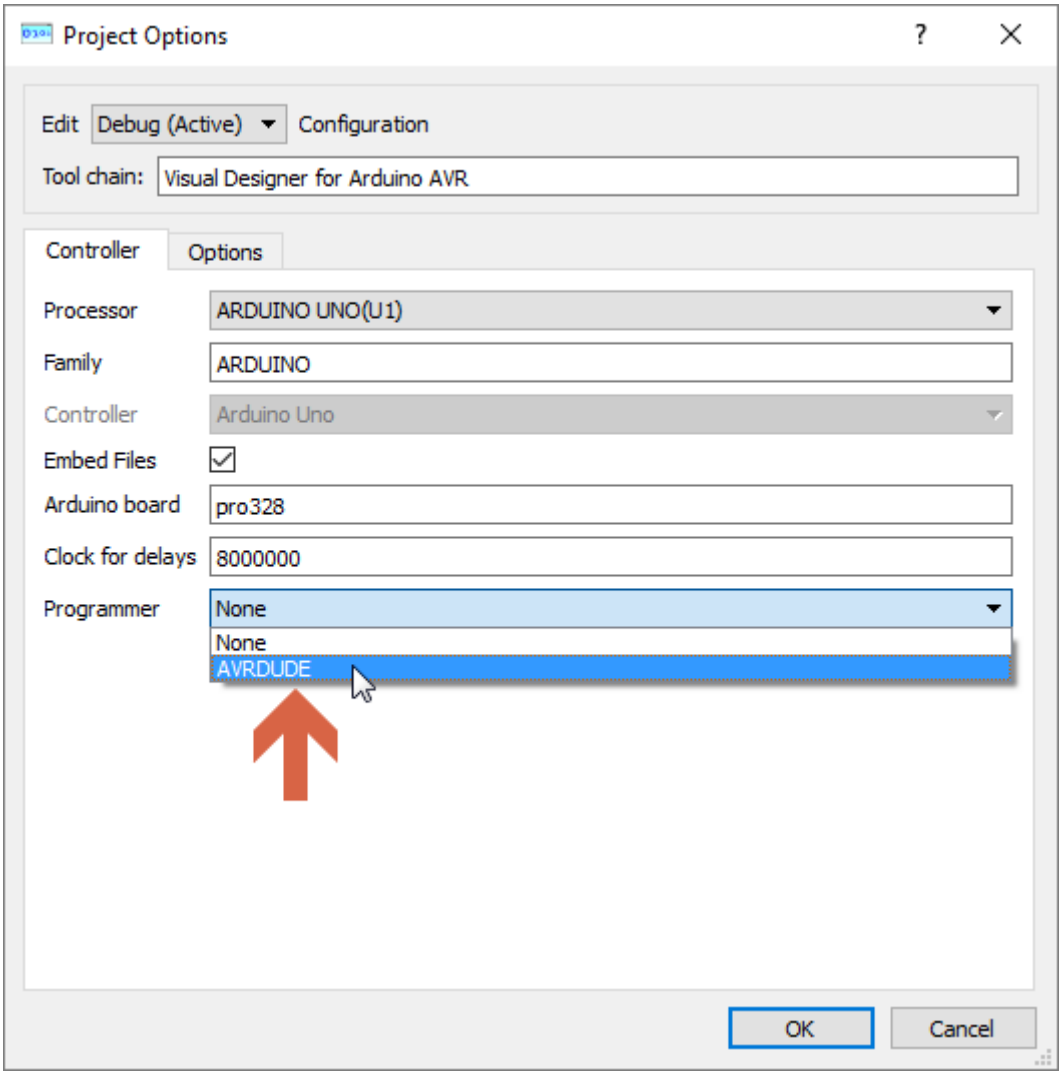
If you prefer, you can change the value in the field to 16MHz. This will be correct for the real hardware but may impact simulation performance.



Change the clock for correct hardware compilation at the expense of degraded simulation performance.

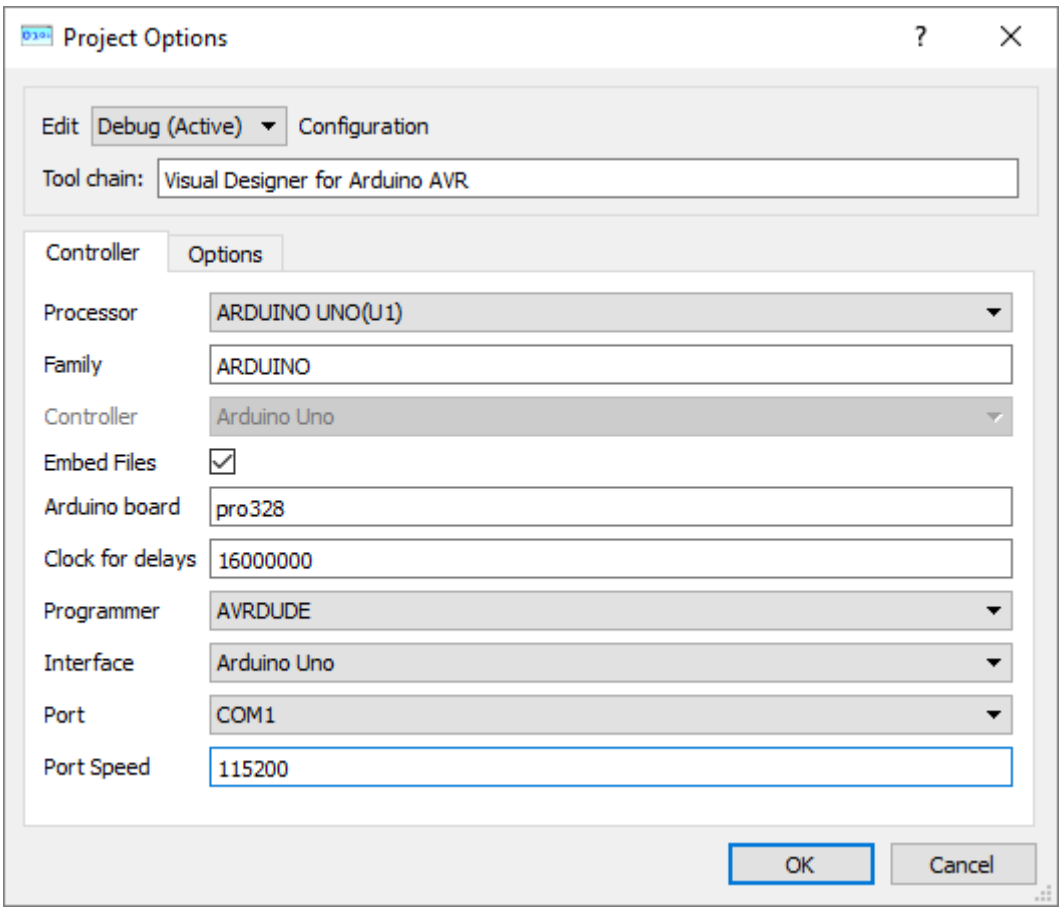
Programmer Option

The programmer field should be set to the AVRDUDE programmer. If this is not available/installed then you can easily get it by installing the Arduino IDE from their website.



Interface Option

The interface field should then be set according to the real hardware (e.g Arduino Uno for the Uno board).



COM Port Option

The COM port field is likely to cause the most confusion. This is determined by your PC when you connect the Arduino hardware to the computer and it is vital that you provide the correct port number in this field. The easiest way to make sure you have the correct number is to

a) Open Device Manager.

b) Plug in the Hardware

c) Make a note of the COMx number that appears and select it in the Port field.

See Also:

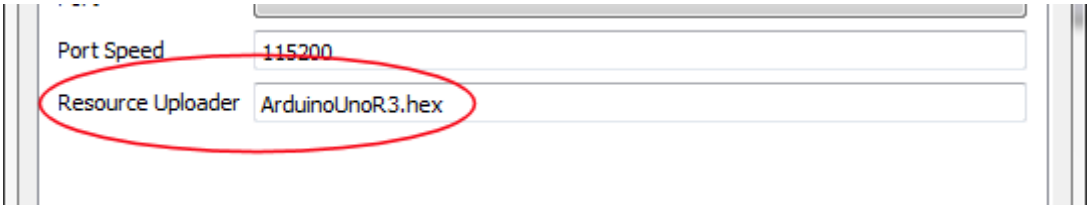
- Troubleshooting : COM Port.

Port Speed Option

The Port Speed or baud rate option is defined in the Arduino datasheets (e.g. 115200 for the Arduino Uno) and should not be changed unless you are sure that you know what you are doing.

Resource Uploader

Finally, the Resource Uploader file is used by Labcenter to automatically transfer Visual Designer resources such as WAV files and Bitmaps to the SD card. You should not modify this field.



Troubleshooting Guide

Program Button is Disabled

Problem

This happens when the programmer is not installed or set.



Solution

Plug the USB cable in to the board and in to the PC. Make sure that the board has power (LEDs will light up on the board). Visual Designer does not automatically detect that a board has been plugged in so the Upload icon may still be greyed out. Click on the Project Settings icon:



We need to let Proteus know which programmer we wish to use. In this case it is the AVRDUDE If there is nothing listed you will need to install Arduino IDE from: https://www.arduino.cc/en/Main/Software

Programming takes a long time and then reports multiple errors

> You can stop a build or program at any time by clicking on the Stop Build icon on the toolbar

There can be several different causes for this...

Cause 1

Problem

The wrong COM port has been selected. Go back to the Project settings and select the correct COM port.

VSM Studio Output

```
Uploading...

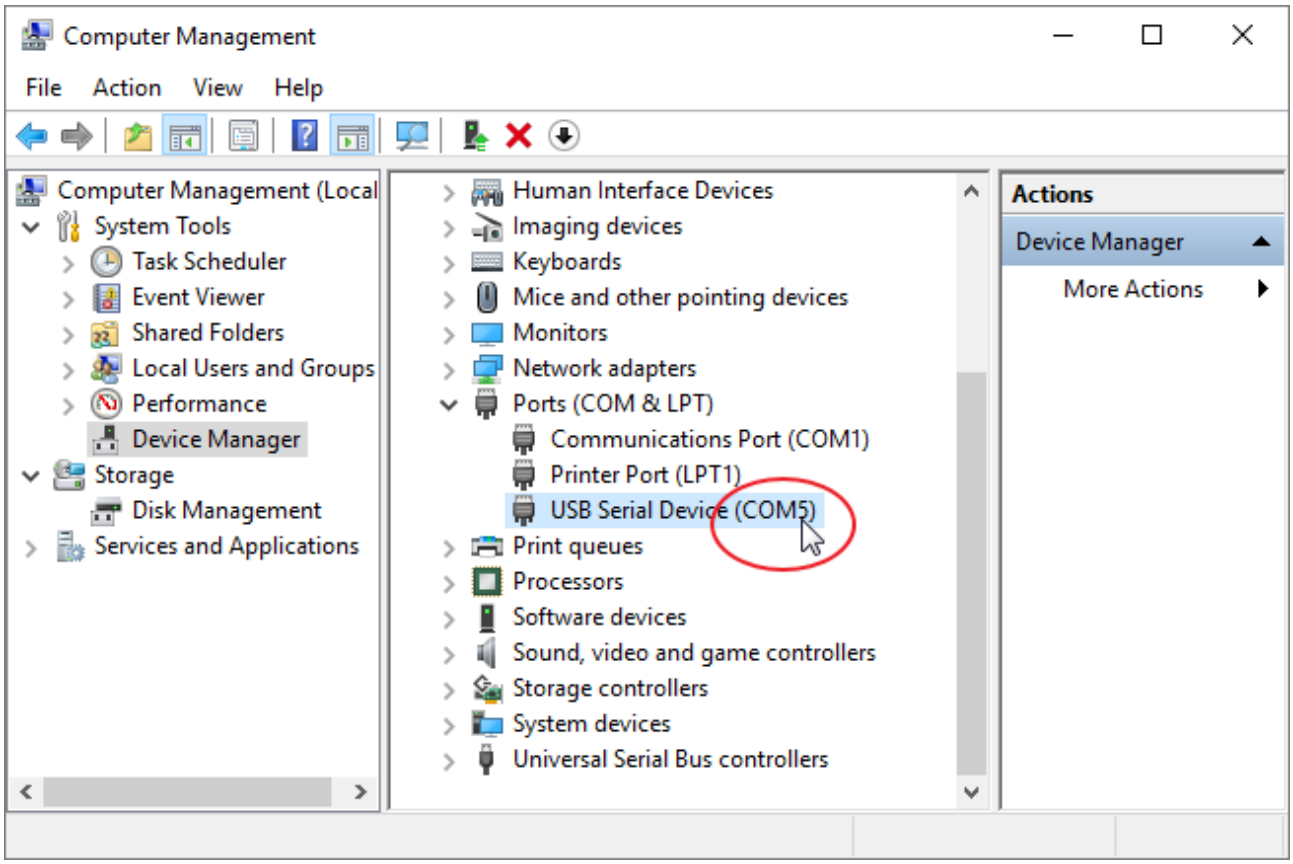
avrdude.exe: stk500_recv(): programmer is not responding
avrdude.exe: stk500_getsync() attempt 1 of 10: not in sync: resp=0x00
avrdude.exe: stk500_recv(): programmer is not responding
avrdude.exe: stk500_getsync() attempt 2 of 10: not in sync: resp=0x00
avrdude.exe: stk500_recv(): programmer is not responding
avrdude.exe: stk500_getsync() attempt 3 of 10: not in sync: resp=0x00
avrdude.exe: stk500_recv(): programmer is not responding
avrdude.exe: stk500_getsync() attempt 4 of 10: not in sync: resp=0x00
avrdude.exe: stk500_recv(): programmer is not responding
avrdude.exe: stk500_getsync() attempt 5 of 10: not in sync: resp=0x00
avrdude.exe: stk500_recv(): programmer is not responding
avrdude.exe: stk500_getsync() attempt 6 of 10: not in sync: resp=0x00
avrdude.exe: stk500_recv(): programmer is not responding
avrdude.exe: stk500_getsync() attempt 7 of 10: not in sync: resp=0x00
avrdude.exe: stk500_recv(): programmer is not responding
avrdude.exe: stk500_getsync() attempt 8 of 10: not in sync: resp=0x00
avrdude.exe: stk500_recv(): programmer is not responding
avrdude.exe: stk500_getsync() attempt 9 of 10: not in sync: resp=0x00
avrdude.exe: stk500_recv(): programmer is not responding
avrdude.exe: stk500_getsync() attempt 10 of 10: not in sync: resp=0x00

avrdude.exe done.  Thank you.
Error code 1

Uploading failed
```

Solution

- Making sure the board is plugged in
- Open the Windows Device Manager
- Select 'Windows Device Manager > Ports (COM & LPT1) > USB Serial Device (COMx)'.
- Use what ever COM port number is stated there.



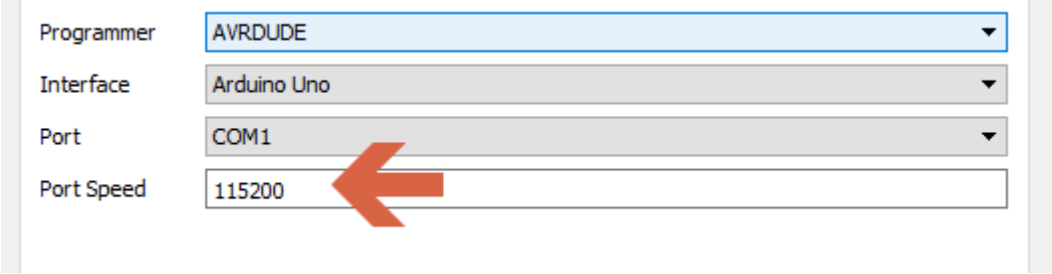
Cause 2

Problem

Incorrect Port Speed.

Solution

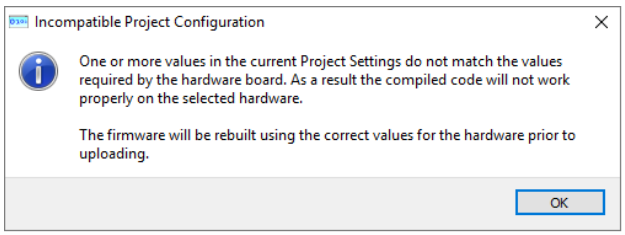
Have a look on the data sheet to find out what the Port Speed should be and in the project Settings form, change it to be the correct number (Generally 115200).

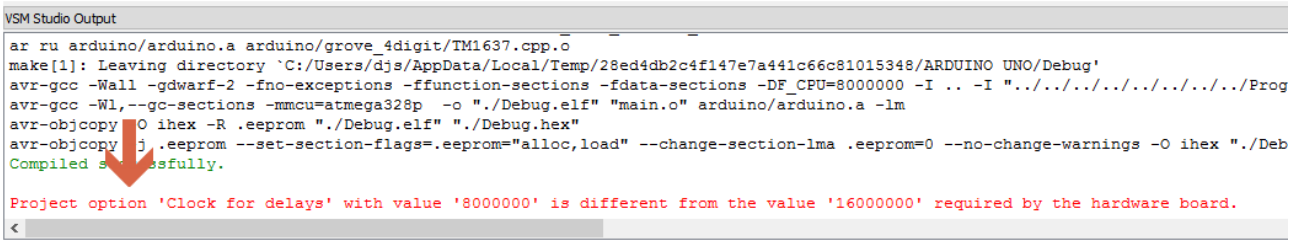


Clock for Delays Error

Problem

You get the following error messges:





This will generally be because the clock speed / 'Clock for Delays' value on the board for this particular variant is different than that of the program. In the error message in the output window, it will show the 'set' value in the in the Project Settings dialog, and the 'expected' value required by the board. The software will compile the code with the 'expected' and upload the program. But it will not change it in the settings, so this error will always appear every time the Upload button is pressed.

Solution.,

This is not necessarily a problem. Indeed, for good simulation performance on modest computers it is almost essential that the clock speed for simulation is different from the clock speed of the real hardware. However, if you do want to change the field to be consistent you can do so by going to to the Project Settings dialog and set the Clock For Delays to be the 'Expected' value or read the datasheet for that variant to find out what the clock speed should be..



ELF File not found

Problem

There is no compiled firmware to upload

Solution

Compile the firmware via the build button and then try again.



CONFIGURING RASPBERRY PI

Introduction

Unlike the Arduino, the Raspberry Pi needs to be configured to work with Proteus before it can be programmed. This is normally a one time configuration but can be quite involved. There are three main stages:

1) Prepare the SD Card.

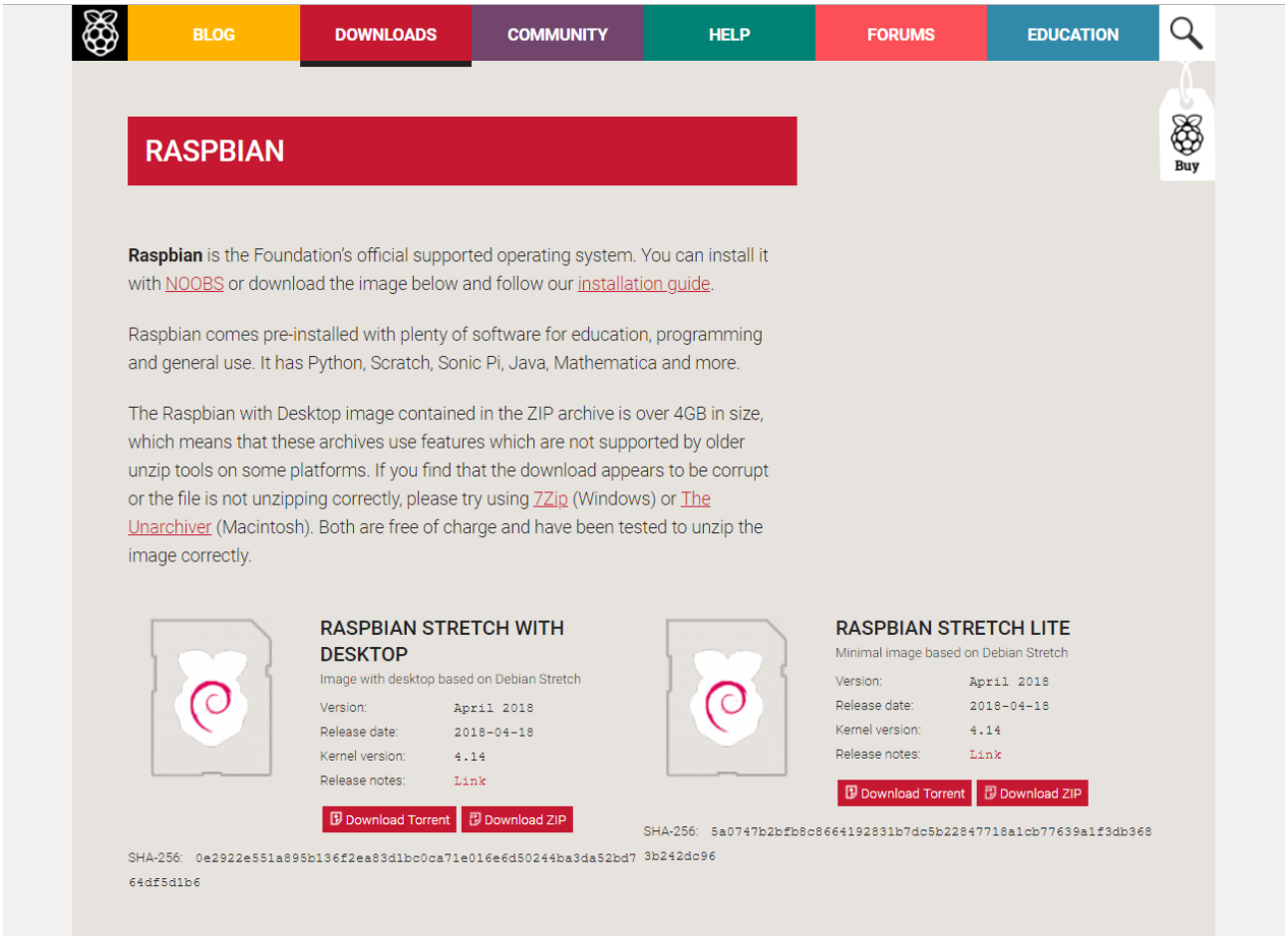
2) Find the Raspberry Pi.

3) Configure the Raspberry Pi for use with Proteus.

Preparing the SD Card

Before you start with your Raspberry Pi you will need a copy of Raspbian installed on an SD card. If you don't know how to do this there is a link below which will walk you through it.

https://www.raspberrypi.org/downloads/raspbian/

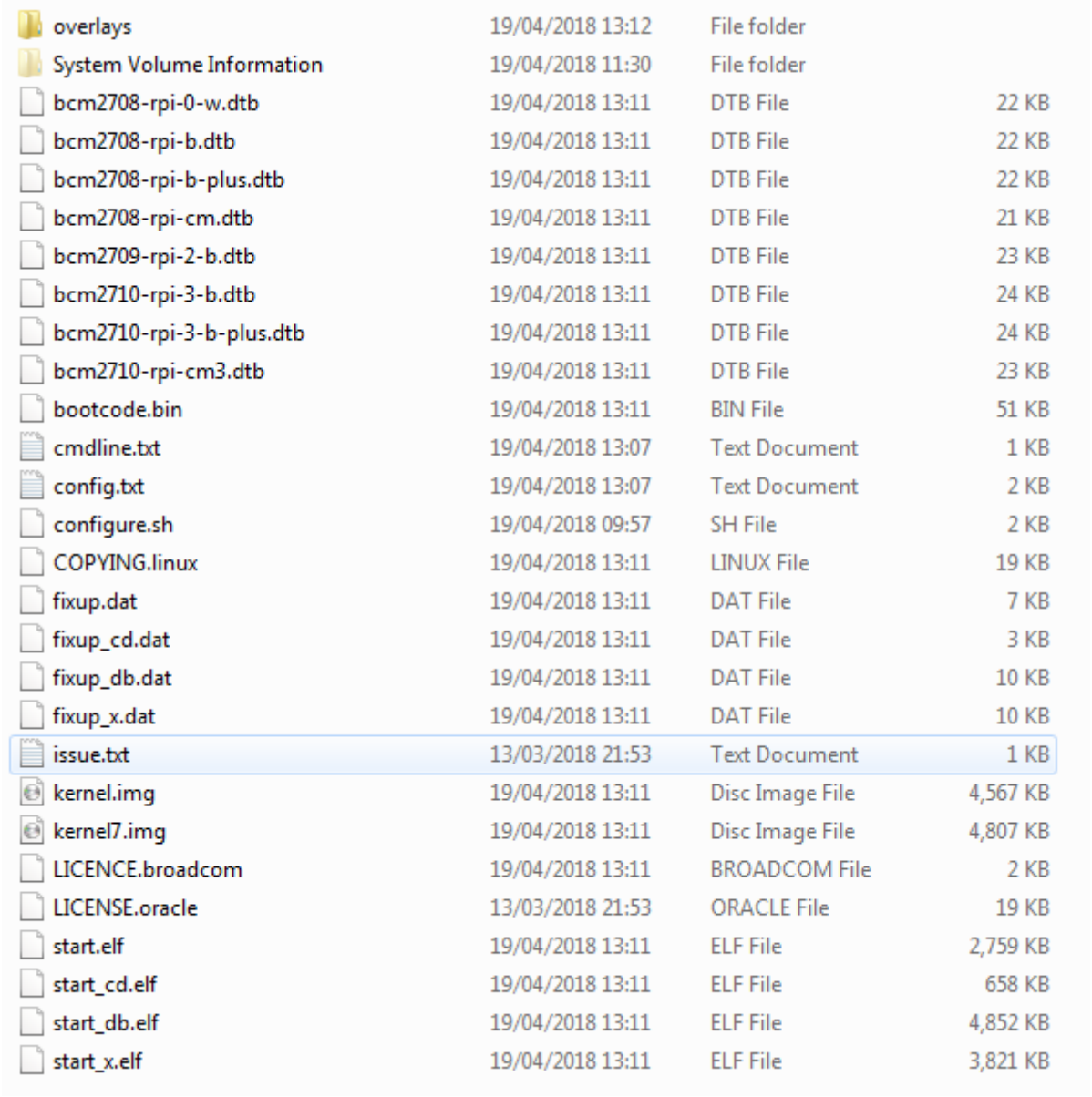


Download Raspbian onto an SD Card.

Proteus will work with both Raspbian and Raspbian Light depending on your preference however Raspbian Light will boot up much faster.

Copying Files

Now that you have Raspbian on an SD card you'll need to open it on your main computer. Don't plug the SD card into the Raspberry Pi yet. Instead, remove it and re-insert it into your computer. Once you open the SD card in your computer you should see a list of files like this.



overlays	19/04/2018 13:12	File folder	
System Volume Information	19/04/2018 11:30	File folder	
bcm2708-rpi-0-w.dtb	19/04/2018 13:11	DTB File	22 KB
bcm2708-rpi-b.dtb	19/04/2018 13:11	DTB File	22 KB
bcm2708-rpi-b-plus.dtb	19/04/2018 13:11	DTB File	22 KB
bcm2708-rpi-cm.dtb	19/04/2018 13:11	DTB File	21 KB
bcm2709-rpi-2-b.dtb	19/04/2018 13:11	DTB File	23 KB
bcm2710-rpi-3-b.dtb	19/04/2018 13:11	DTB File	24 KB
bcm2710-rpi-3-b-plus.dtb	19/04/2018 13:11	DTB File	24 KB
bcm2710-rpi-cm3.dtb	19/04/2018 13:11	DTB File	23 KB
bootcode.bin	19/04/2018 13:11	BIN File	51 KB
cmdline.txt	19/04/2018 13:07	Text Document	1 KB
config.txt	19/04/2018 13:07	Text Document	2 KB
configure.sh	19/04/2018 09:57	SH File	2 KB
COPYING.linux	19/04/2018 13:11	LINUX File	19 KB
fixup.dat	19/04/2018 13:11	DAT File	7 KB
fixup_cd.dat	19/04/2018 13:11	DAT File	3 KB
fixup_db.dat	19/04/2018 13:11	DAT File	10 KB
fixup_x.dat	19/04/2018 13:11	DAT File	10 KB
issue.txt	13/03/2018 21:53	Text Document	1 KB
kernel.img	19/04/2018 13:11	Disc Image File	4,567 KB
kernel7.img	19/04/2018 13:11	Disc Image File	4,807 KB
LICENCE.broadcom	19/04/2018 13:11	BROADCOM File	2 KB
LICENSE.oracle	13/03/2018 21:53	ORACLE File	19 KB
start.elf	19/04/2018 13:11	ELF File	2,759 KB
start_cd.elf	19/04/2018 13:11	ELF File	658 KB
start_db.elf	19/04/2018 13:11	ELF File	4,852 KB
start_x.elf	19/04/2018 13:11	ELF File	3,821 KB

*SD Card Files on your computer (*Not* the Raspberry Pi)*

Now collect all the files from the drivers directory of your Proteus installation:

..\VSM Studio\drivers\RaspberryPi\Configure

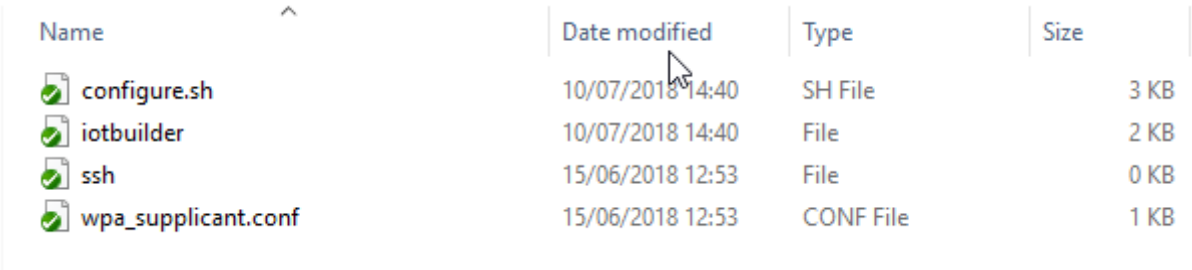
And insert them into the SD card. If you're not using ethernet you will need to edit wpa_supplicant.conf which will allow you to connect to the network on bootup. Simply input the network name and password and the Raspberry Pi will connect on bootup.

Also at this stage edit the configure.sh file and provide a unique hostname for the Raspberry Pi.

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=«your_ISO-3166-1_two-letter_country_code»

network={
    ssid="«your_SSID»"
    psk="«your_PSK»"
    key_mgmt=WPA-PSK
}
```

Enter wireless access details in the wpa_supplicant file.



Name	Date modified	Type	Size
configure.sh	10/07/2018 14:40	SH File	3 KB
iotbuilder	10/07/2018 14:40	File	2 KB
ssh	15/06/2018 12:53	File	0 KB
wpa_supplicant.conf	15/06/2018 12:53	CONF File	1 KB

Files to be copied onto the SD Card from your computer.

Now safely eject the card from the computer.

Finding the Raspberry Pi

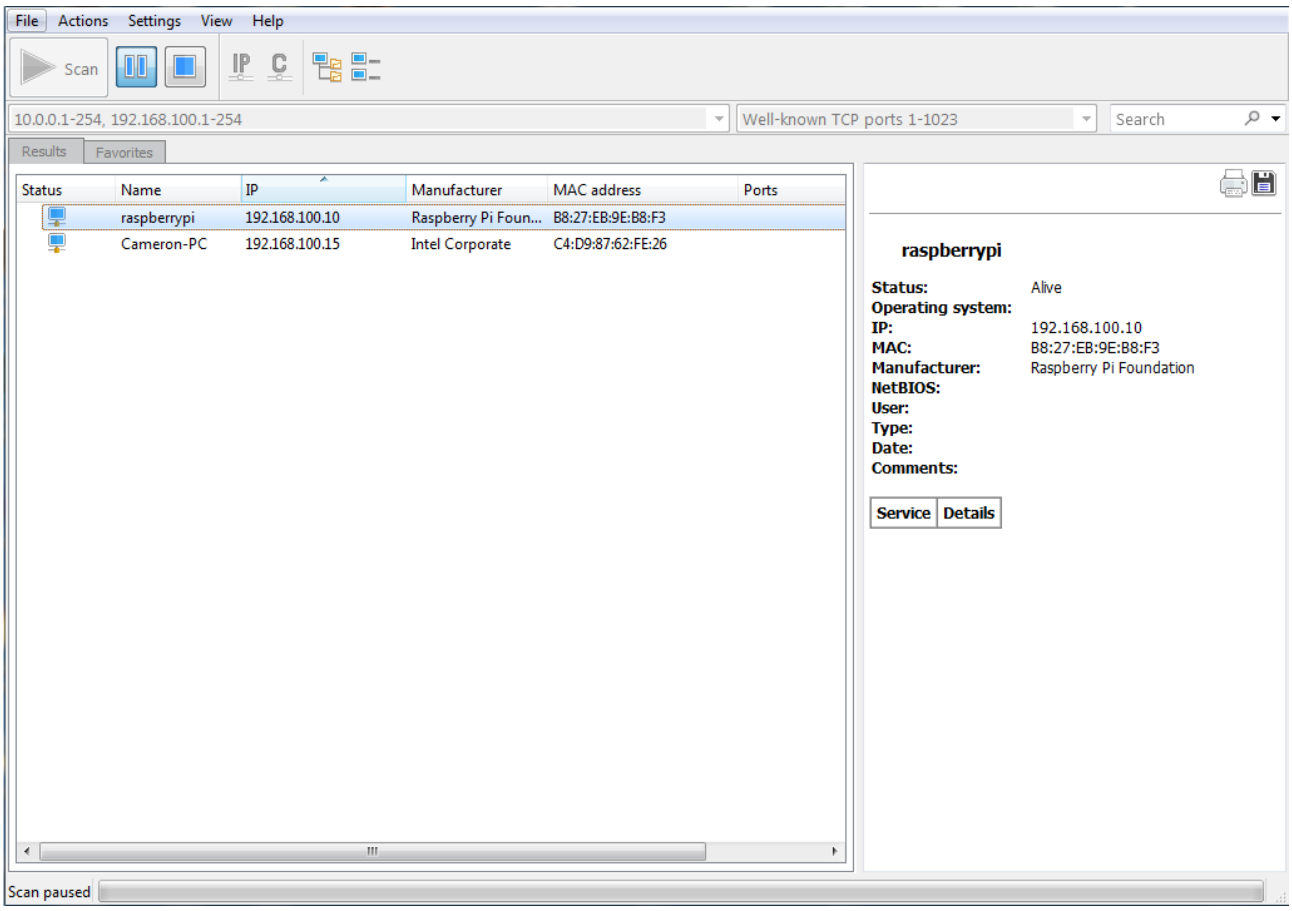
You will need to find the IP address of your raspberry Pi. There are numerous ways to do so, including the following.

First, installing a free port scanner. This saves time and means you don't have to plug in your Raspberry Pi to a screen and keyboard. So visit the Advanced Port Scanner website.

https://www.advanced-port-scanner.com/

Now follow their instructions and install Advanced Port Scanner. The Download button is on their main page. Note this software is free.

Once installed power on the Raspberry Pi.Raspian light will boot faster than normal Raspian as there is no GUI to load. Then use Advanced Port Scanner to look for your Raspberry Pi.

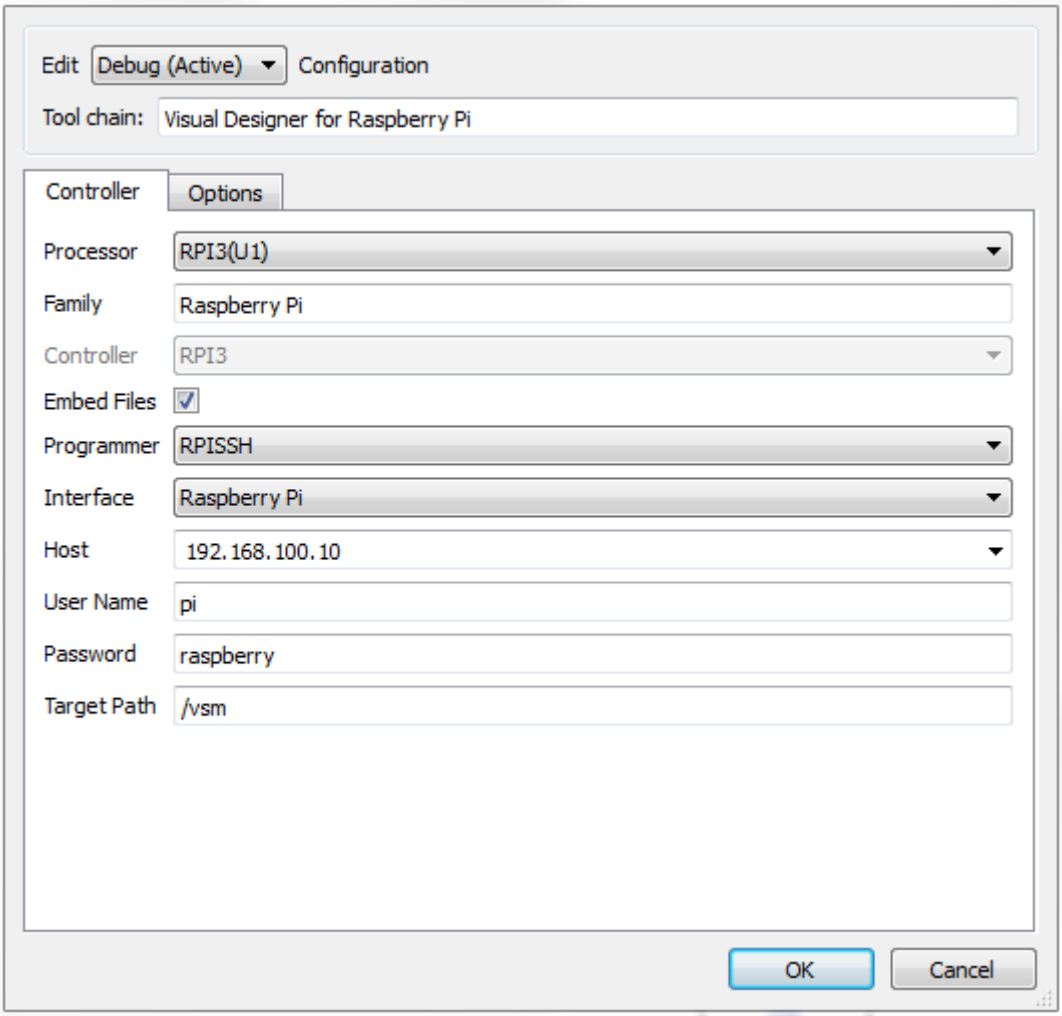


You will most likely have lots of devices on your network. You will be looking for a device with name raspberrypi and manufacturer Raspberry Pi Foundation. Find the IP which in this case is 192.168.100.10. Do not turn off the raspberry Pi.

Configuration of the Raspberry Pi

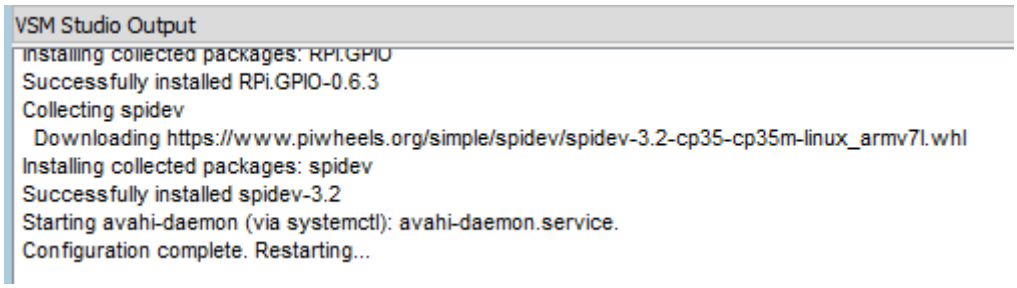
Method 1: Inside Proteus

Create a blank Raspberry Pi Project in proteus and open project settings. Change Programmer from None to RPISSH and enter the IP address into the Host section.



Now Upload the blank project to your Pi. A dialogue box will pop up asking for confirmation to complete the Raspberry Pi configuration, press OK to continue (Picture may be required). This section will take a while as it downloads necessary libraries and sets it's self up

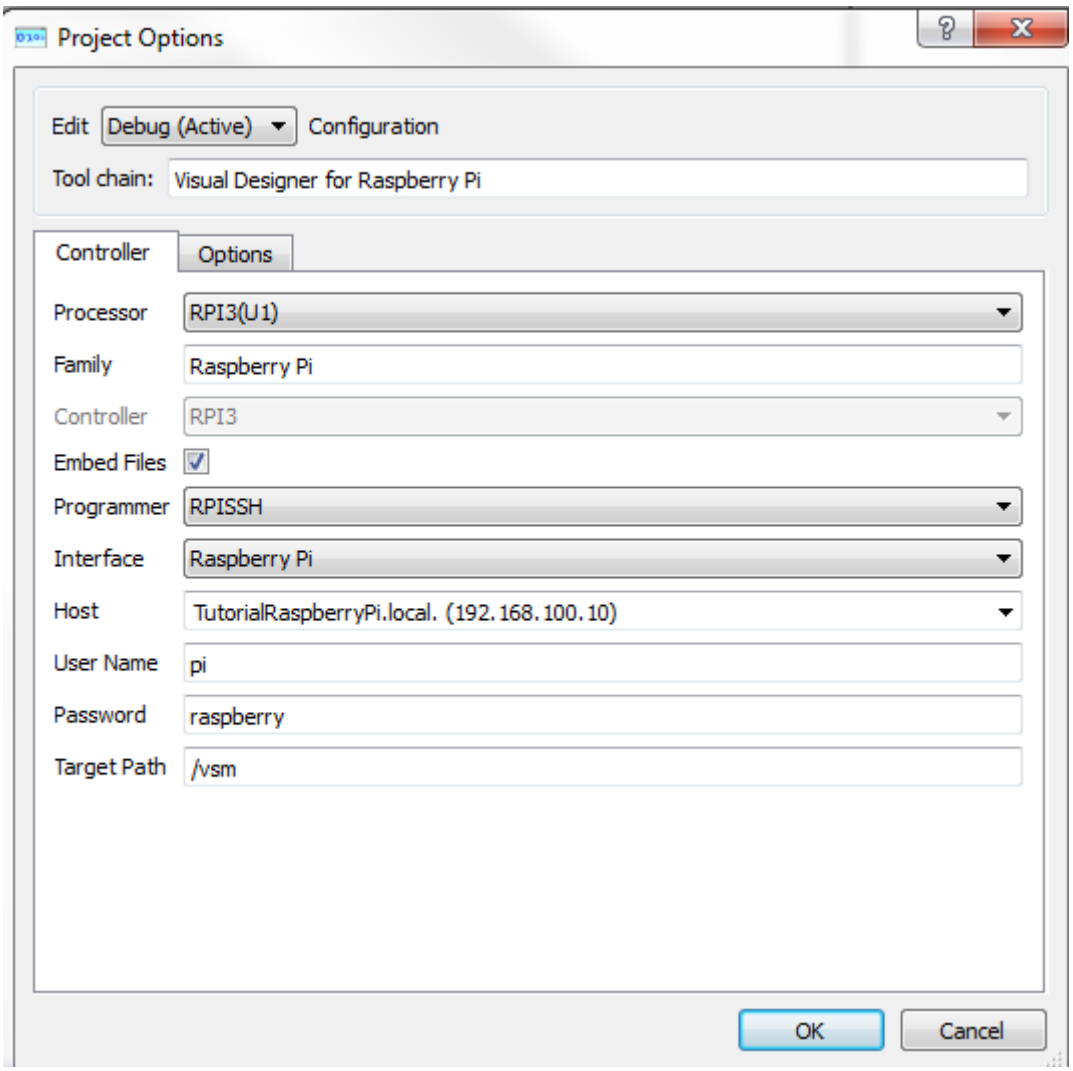
Once complete, the Raspberry Pi will reboot and connection will be lost giving Error Code: -1



Wait a moment for the Raspberry Pi to reboot and the process is complete.

Your raspberry pi will now appear in Proteus every time you boot it up and advanced port scanner is no longer needed.

Now if you go into program settings the raspberry pi will be there with whatever name you provided in the configure.sh file.



Method 2: PuTTY

You should be able to work with PuTTY to talk to the Raspberry Pi without plugging in a keyboard, mouse and screen. If you are using Raspbian and are plugging in a keyboard and mouse you may not need this step.

This link will take you to PuTTYs main site and will talk you through the download steps.

https://www.raspberrypi.org/documentation/remote-access/ssh/windows.md

Connect PuTTY to the Raspberry Pi and write :

sudo /boot/configure.sh

This may take a while to install everything but once it has finished you will need to reboot the Raspberry Pi.

Method 3 : Keyboard and Mouse

If you are instead using keyboard and screen open the command line which is called Terminal in Raspbian. Now write:

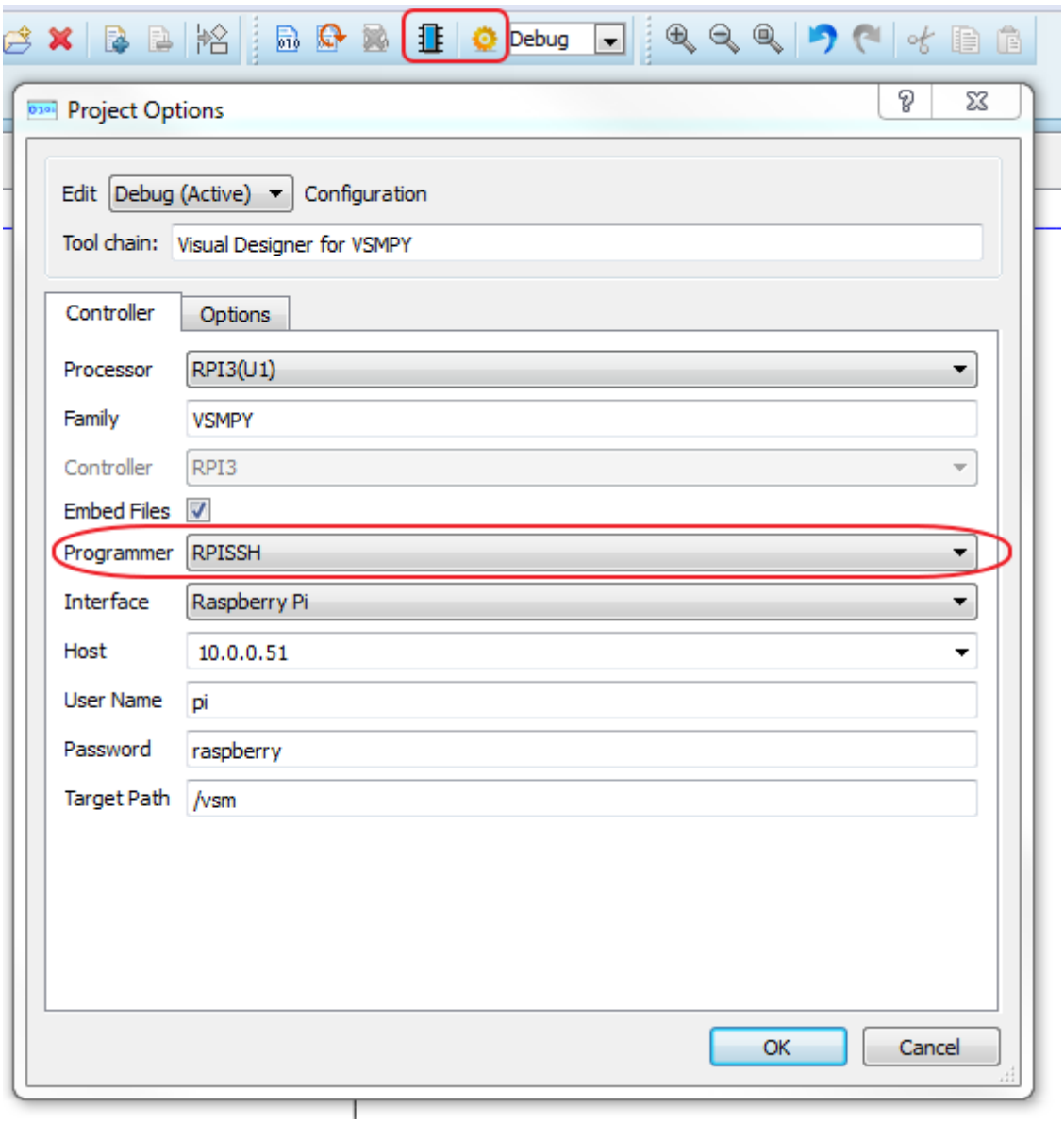
sudo /boot/configure.sh

This may take a while to install everything but once it has finished you will need to reboot the Raspberry Pi.

Programming the Raspberry Pi

⚠ This is the process for programming your project firmware once you have completed the one time configuration of the device. It will not work if you have not first configured the Raspberry Pi

Once you have completed your visual designer project and want to program the real hardware simply open Program Options. Change Programmer from NONE to RPISSH and it will automatically find your Raspberry Pi if it is turned on.



Setting the Raspberry Pi programmer inside VSM Studio

Now hitting the Program button will upload your project to the Raspberry Pi and it will start running immediately.



Program button to transfer your firmware to the hardware.

Note that if you have changed the Username or Password within the Raspberry Pi you will also need to change it in the Program Settings too.

TUTORIALS

Introduction

It's remarkably simple to design and create electronic projects with Visual Designer. These short tutorials each cover a separate project and include various different techniques and methods in the project design.

Arduino Tutorials

- Tutorial 1 : Flashing an LED (Grove).

  Covers project creation, adding peripherals. basic flowchart programming and Proteus simulation.

- Tutorial 2 :Nightlight (Grove)

  Includes grove module configuration, decision making flowcharts, interactive simulation and basic debugging.

- Tutorial 3: Data Storage (AdaFruit Shield)

  Includes resource handling, adding shields, taking measurements and programming physical hardware.

- Tutorial 4 : Motor Control (Adafruit Shield)

  Includes programming with loops, adding and calling sub-routines and using the clipboard..

- Tutorial 5: LED (Schematic Design)

  Includes schematic entry for the electronics, using interrupts and CPU methods in Visual Designer.

Raspberry Pi Tutorials

- Tutorial 1 : Flashing an LED (raw Pi).

  Covers project creation, adding peripherals. basic flowchart programming and Proteus simulation.

- Tutorial 2 : Nightlight (w/ Grove).

  Includes grove module configuration, decision making flowcharts, interactive simulation and basic debugging.

- Tutorial 3 : Motor Control (w/ rPi Hat)

  Includes programming with loops, adding and calling sub-routines and using the clipboard..

TUTORIALS

Introduction

It's remarkably simple to design and create electronic projects with Visual Designer. These short tutorials each cover a separate project and include various different techniques and methods in the project design.

Arduino Tutorials

- Tutorial 1 : Flashing an LED (Grove).

  Covers project creation, adding peripherals. basic flowchart programming and Proteus simulation.

- Tutorial 2 :Nightlight (Grove)

  Includes grove module configuration, decision making flowcharts, interactive simulation and basic debugging.

- Tutorial 3: Data Storage (AdaFruit Shield)

  Includes resource handling, adding shields, taking measurements and programming physical hardware.

- Tutorial 4 : Motor Control (Adafruit Shield)

  Includes programming with loops, adding and calling sub-routines and using the clipboard..

- Tutorial 5: LED (Schematic Design)

  Includes schematic entry for the electronics, using interrupts and CPU methods in Visual Designer.

Raspberry Pi Tutorials

- Tutorial 1 : Flashing an LED (raw Pi).

  Covers project creation, adding peripherals. basic flowchart programming and Proteus simulation.

- Tutorial 2 : Nightlight (w/ Grove).

  Includes grove module configuration, decision making flowcharts, interactive simulation and basic debugging.

- Tutorial 3 : Motor Control (w/ rPi Hat)

  Includes programming with loops, adding and calling sub-routines and using the clipboard..

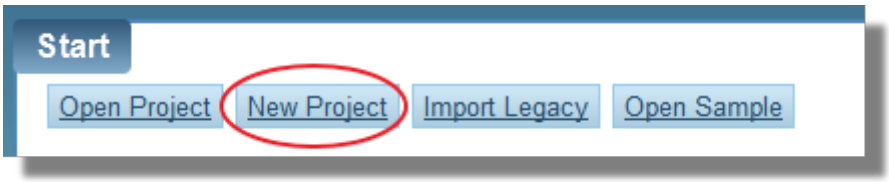
Tutorial : Flashing LED's

Introduction

This tutorial starts when you open Proteus and finishes with the flashing of an LED in a Proteus simulation. The goal here is to cover the basics with the most simple of projects.

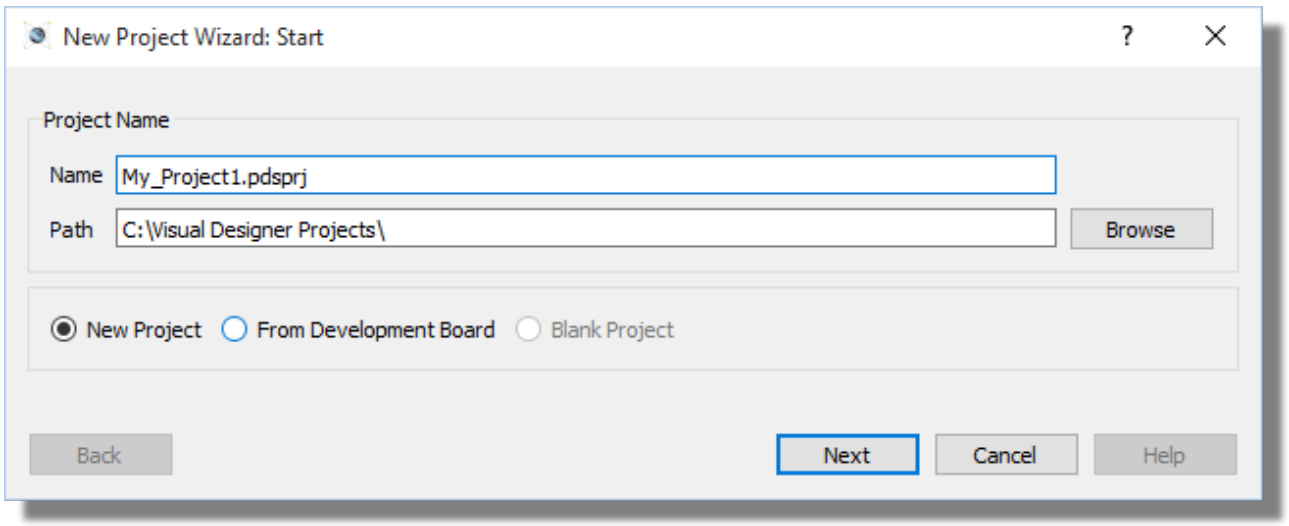
New Project Wizard

Visual Designer is integrated into the Proteus Design Suite and so we start our Visual Designer Projects from the Proteus Home Screen. The easiest and best way to create a Visual Designer Project is, unsurprisingly, through the New Project Wizard.

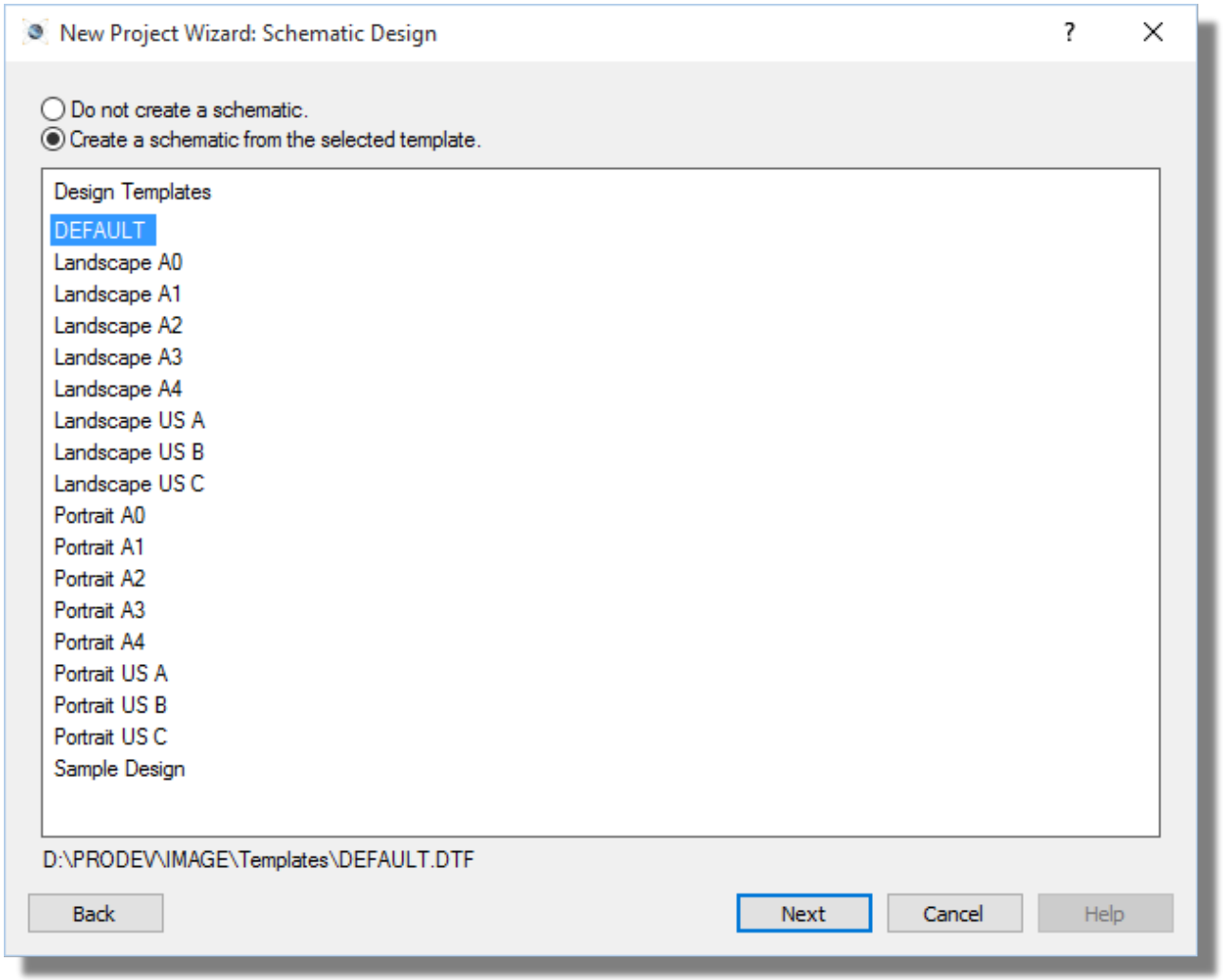


New project from the Home page of Proteus

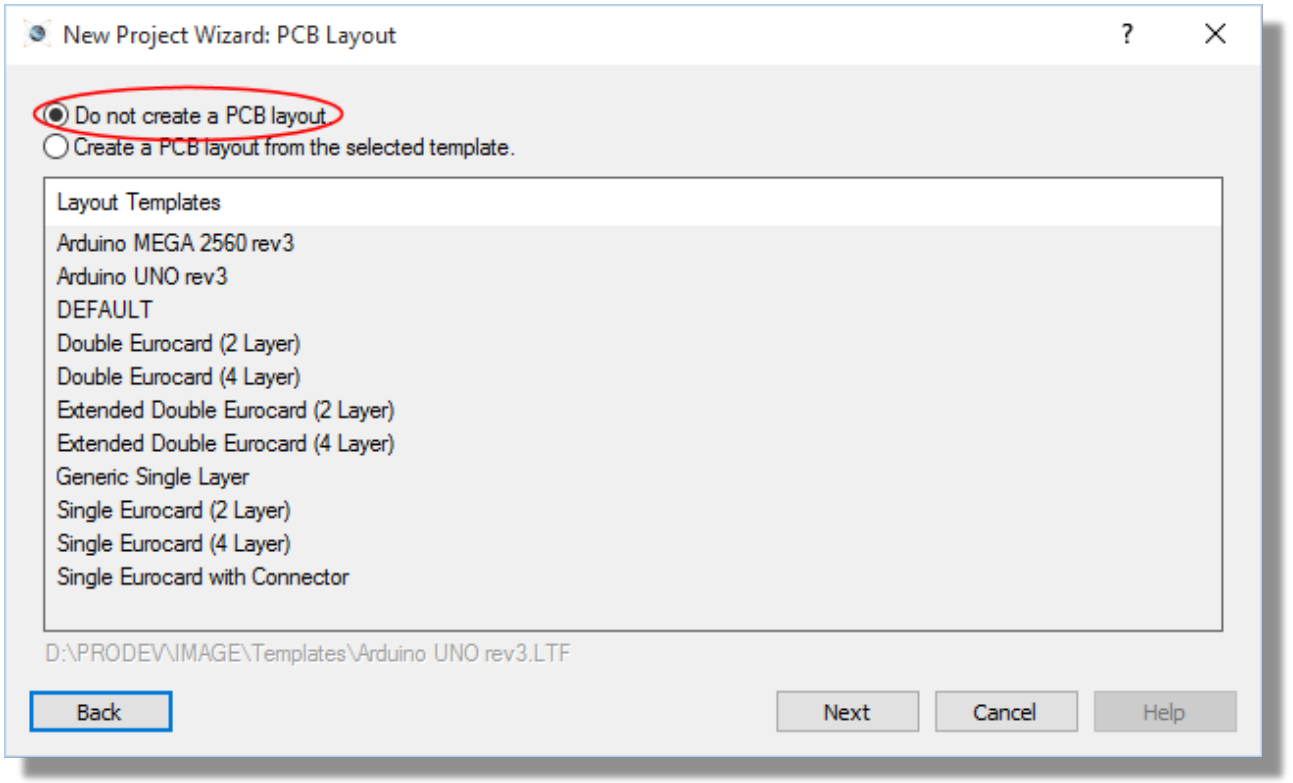
The first page allows you to select the name and destination of your project.



The second screen of the wizard allows us to select which size of schematic sheet we want. The default is fine for all but the largest of projects and is certainly OK for the flashing of an LED.

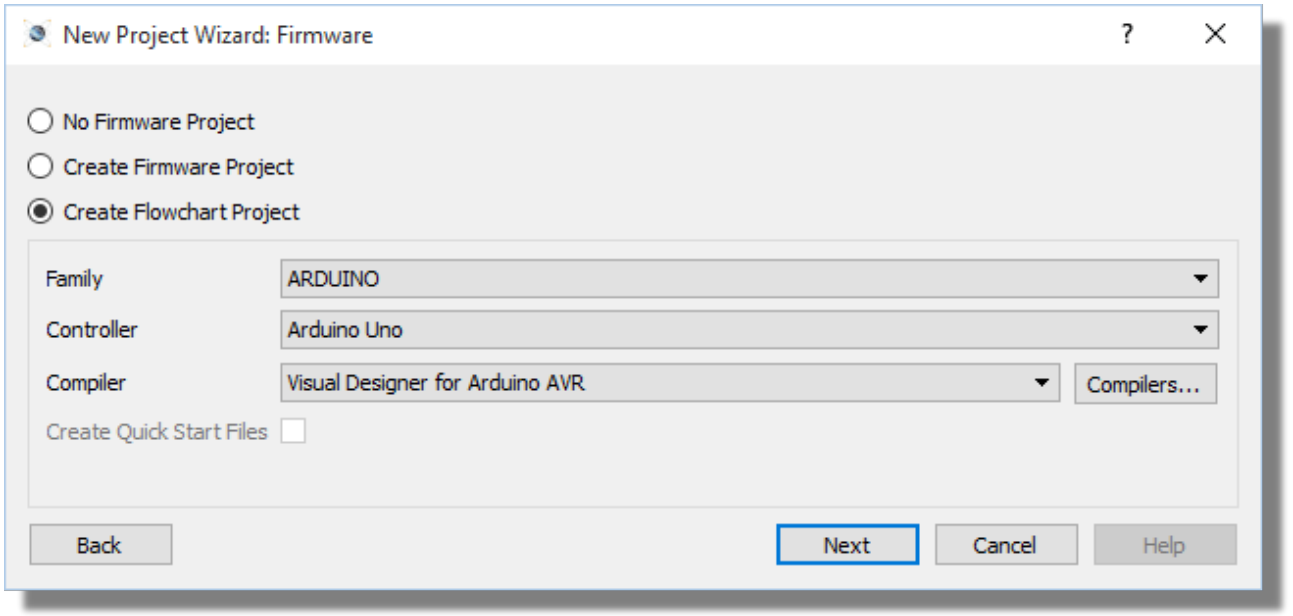


If your license includes Proteus PCB Design the next screen will offer you an option to create an accompanying PCB with the project. It's not necessary for our purposes and beyond the scope of this document so we'll leave the option disabled.



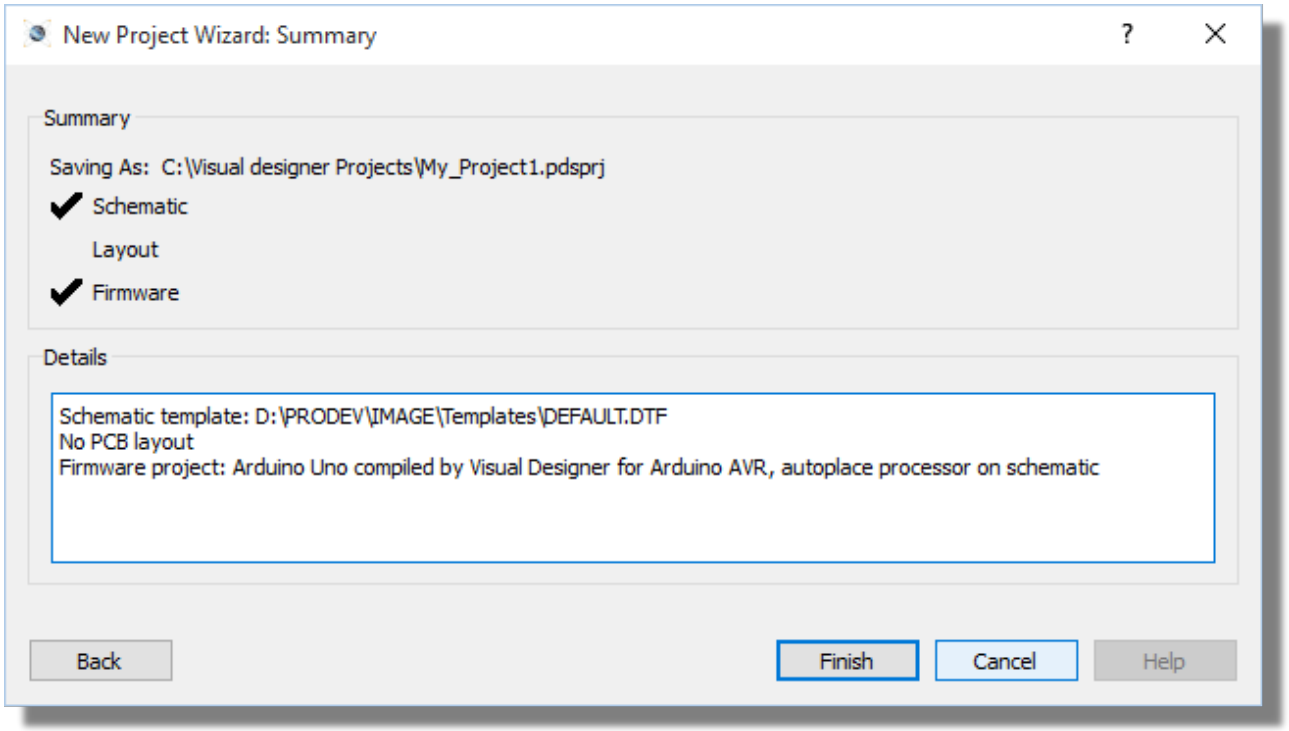
You may not see this screen at all if your license key doesn't include PCB Design.

Next up is the firmware screen and it is here that we really define our Visual Designer project. First, we change the radio buttons at the top to the option for Flowchart Project and then we select the Arduino family and either the Arduino Uno or the Arduino Mega from the controller combo box.

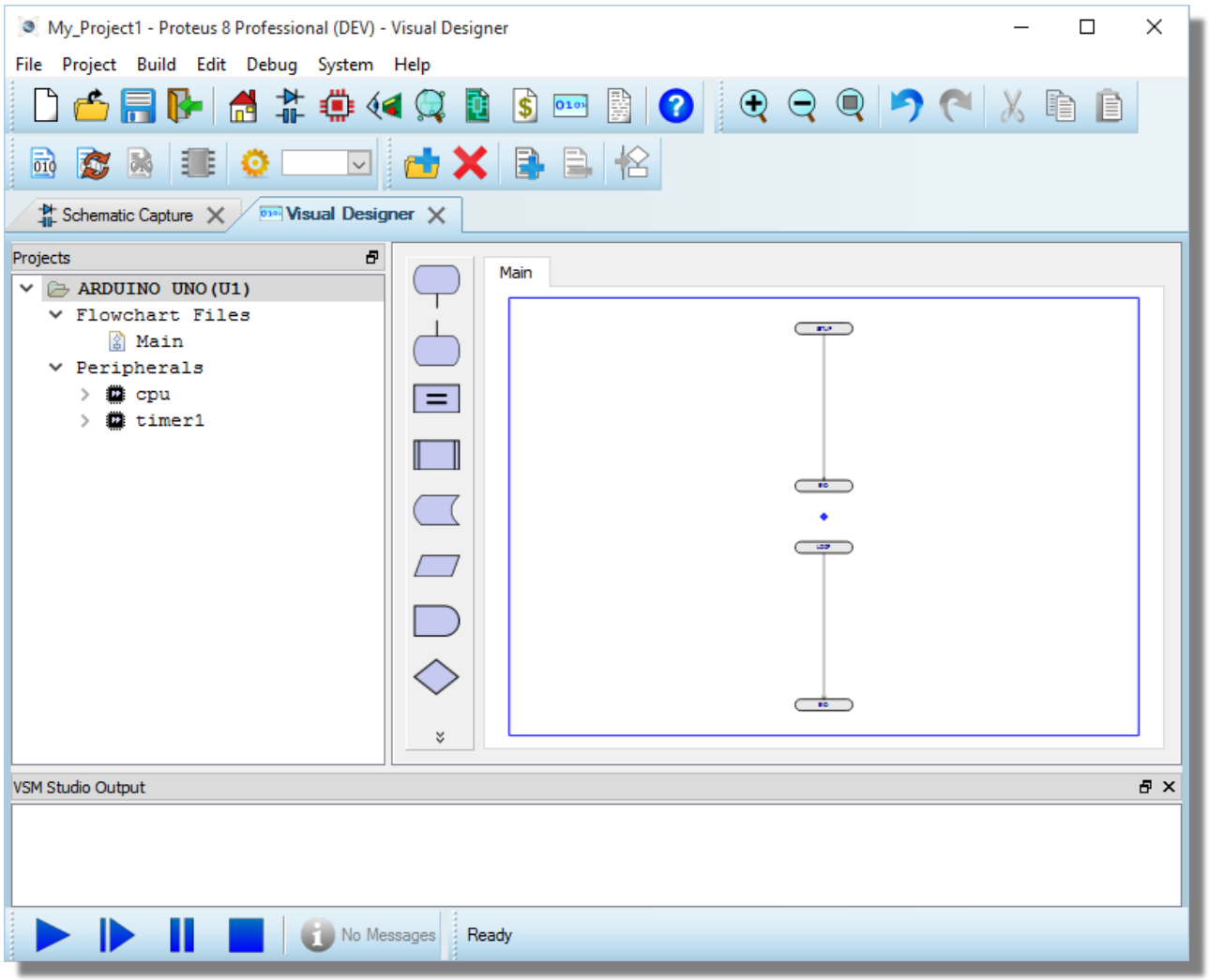


The firmware selection refers to a Proteus VSM simulation project rather than a Visual Designer flowchart project.

Finally, you will be presented with a summary page of your configuration.



After you exit the dialogue form the project will be created and you will see both a skeleton schematic with the Arduino processor placed and a skeleton chart project with the familiar Setup and Loop routines on the Editing Window.



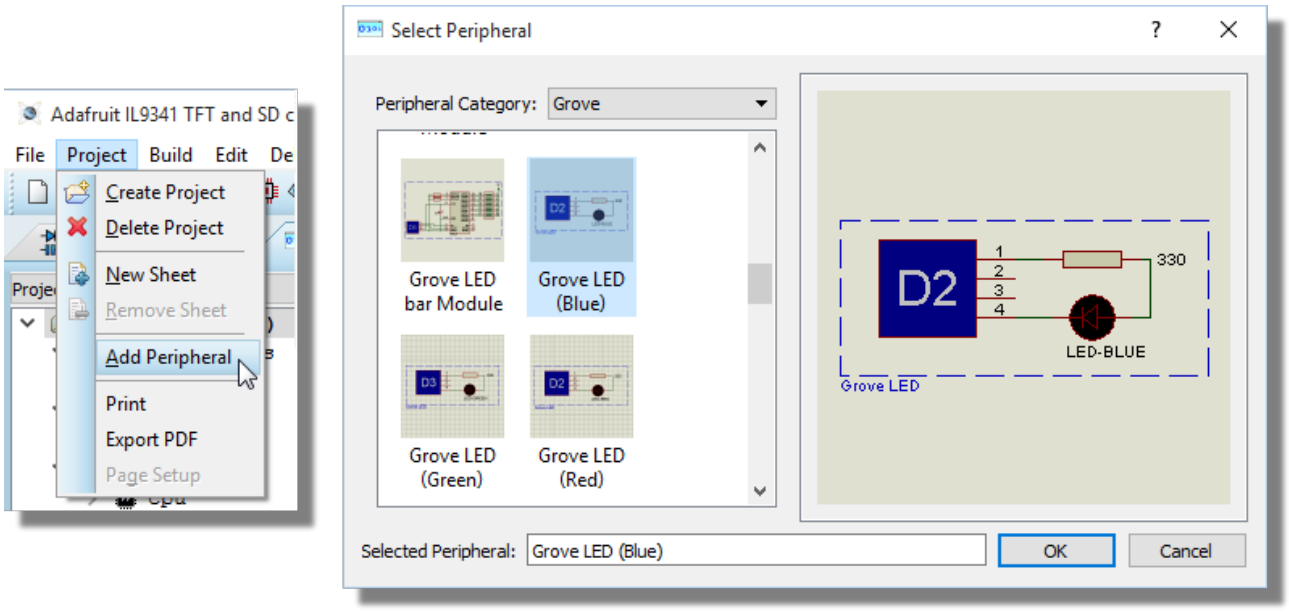
> The standard Arduino Sketch includes two functions, namely Setup and Loop. The Setup function is intended for one time initialization and the Loop function provides a place for the main program loop. Visual Designer for Arduino uses the same paradigm and you must have at least one of these two constructs in your flowchart for it to compile. If you have deleted them, just add an event block and name it appropriately.

Now we are ready to start designing our project.

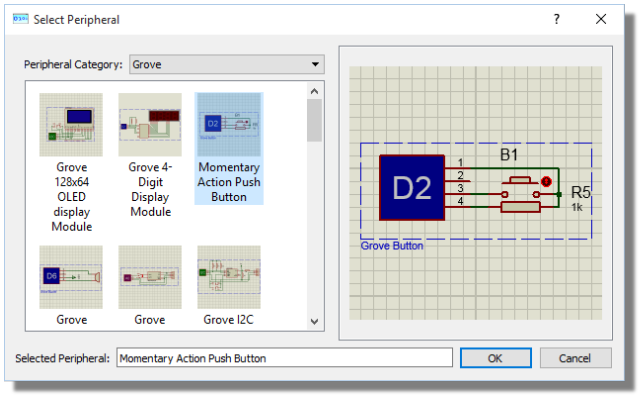
Adding a Peripheral

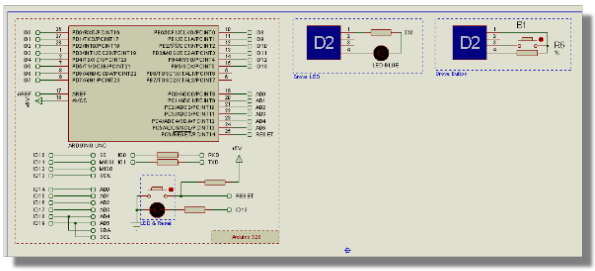
The first thing to do is to add the peripheral(s) for the project. These can be Arduino shields or just little Grove sensors. The Grove system consists of an Arduino shield with lots of 4-pin headers into which any number of sensors, buttons and LED's can be plugged. This makes it very flexible and ideal for experimenting.

We'll be adding a Grove LED. In Visual Designer, we right click on the Project Tree and select the Add Peripheral command. Next, switch to Grove and select one of the LED's.

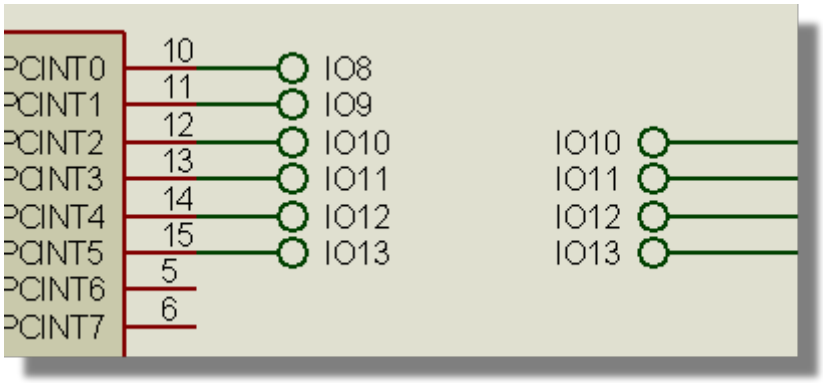


Repeat the process to add a Grove button. If you then switch to the schematic tab you will see that the 'virtual hardware' has been autoplaced for you.

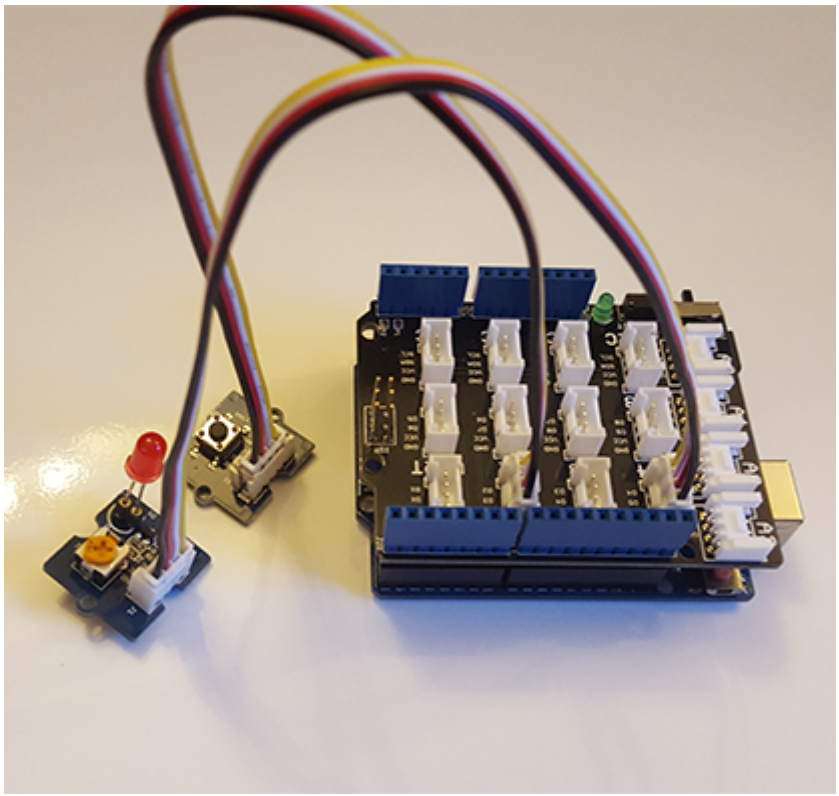




To avoid clutter on the schematic, connections are made on the schematic by giving terminals the same name. Any terminals with the same name can be thought of having an invisible wire between them

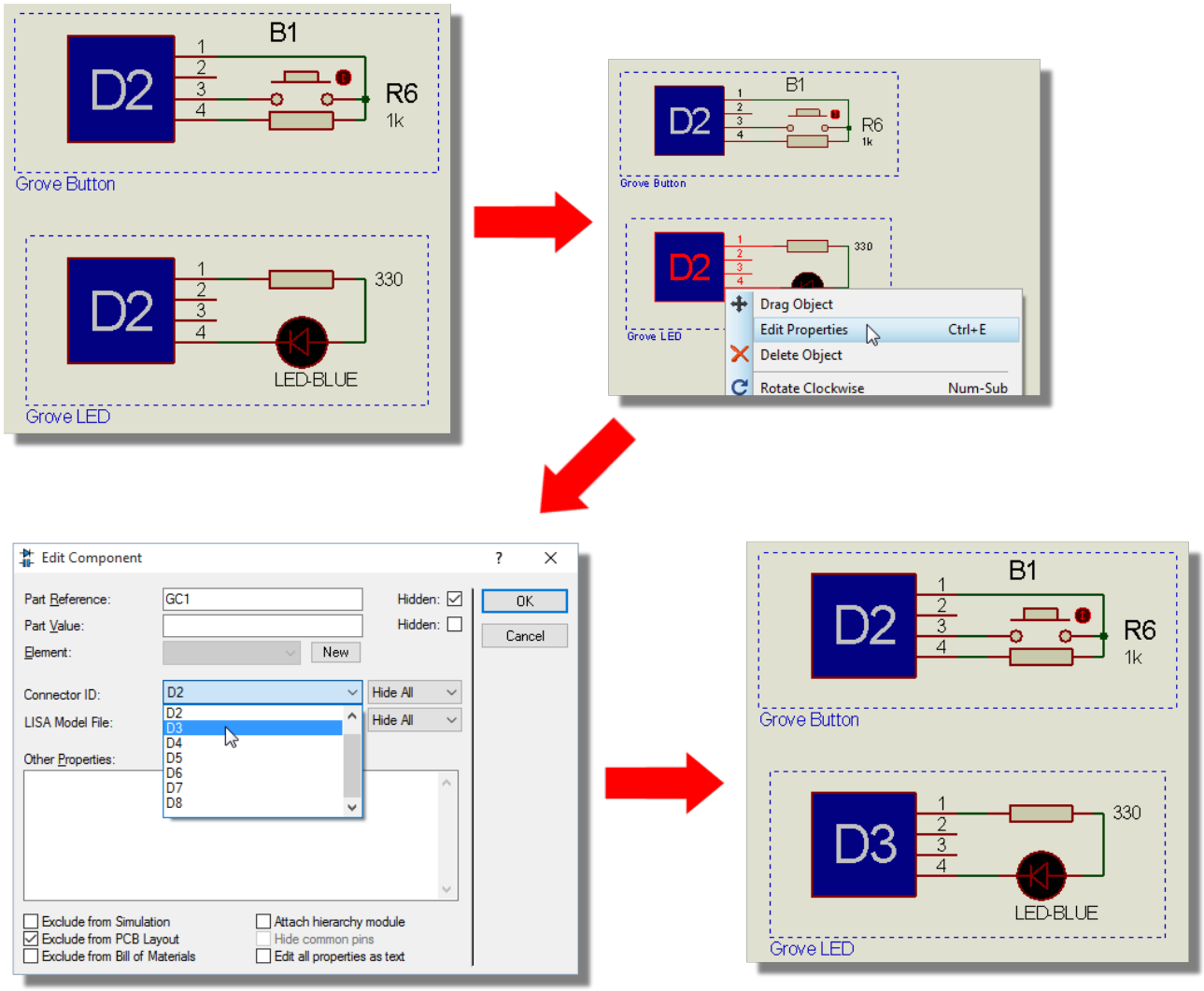


In real hardware, this equates to connecting up the Grove button and Grove LED to the shield and attaching the shield to the Arduino Uno.



Grove LED and Grove Button connected onto the Arduino UNO.

Note that you must plug the peripherals into the same connectors on the Grove shield as they are designated on the Proteus schematic. You must also ensure that you don't have two peripherals with the same connector ID on the schematic; if you do, you need to switch to the schematic and change the ConnectorID on one of the peripherals.

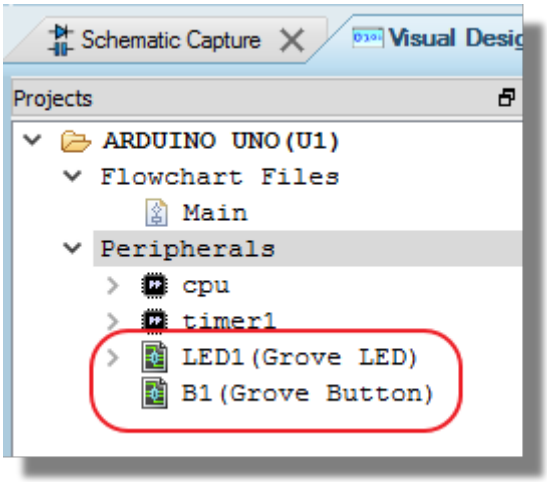


See Also:

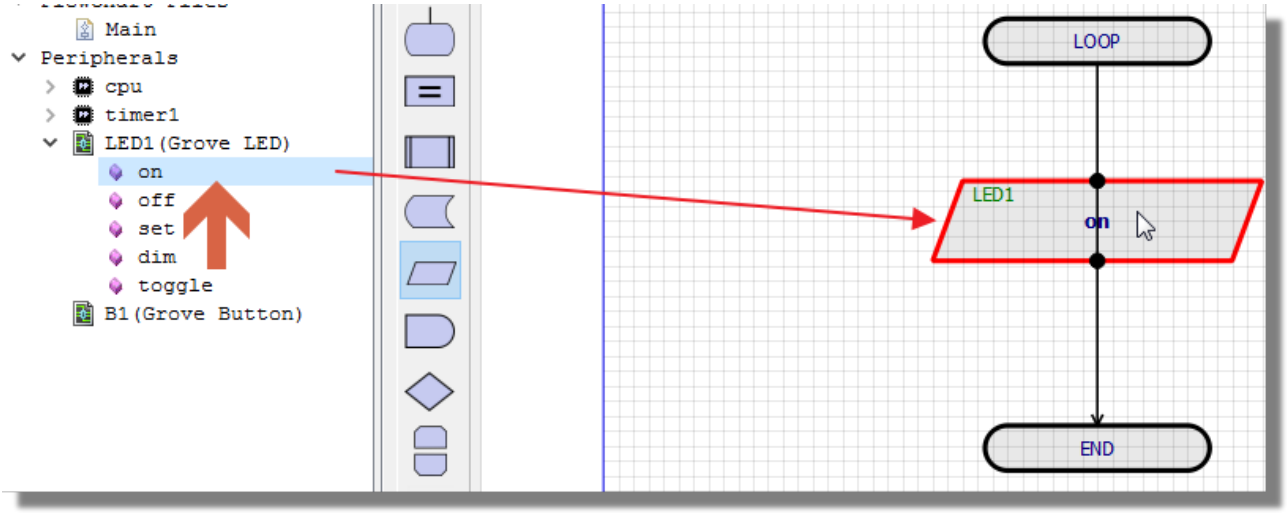
- Peripherals and Shields

Drag and Drop Flowchart Design

Back in Visual Designer, you'll notice that the Project Tree now has two entries under the peripherals section.



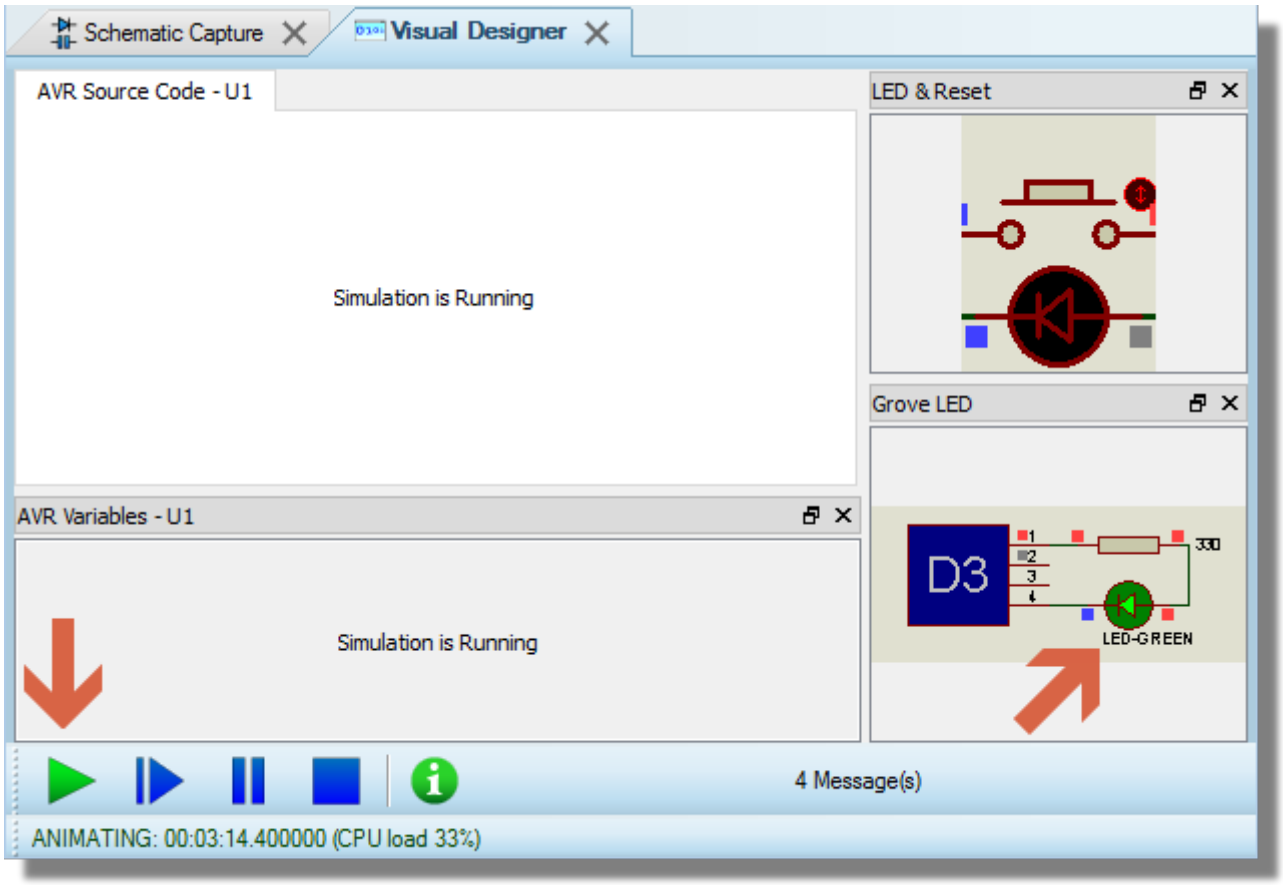
The beauty of Visual Designer is that you can expand the peripherals to find a list of methods. These methods are the primary way in which we interact with the hardware. For example, drag and drop the 'On' method of the LED into the loop routine.



See Also:

Attaching a block

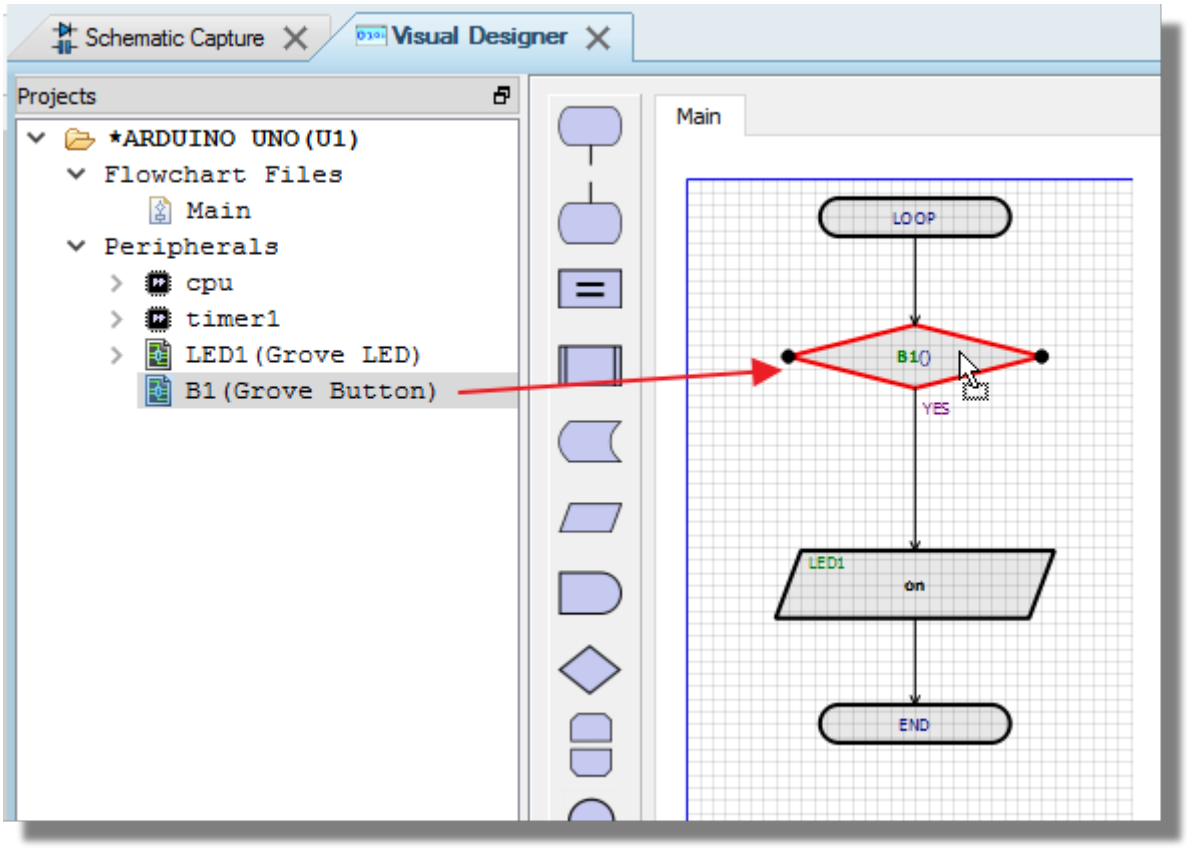
To test this simply press the play button. The program will compile, the simulator will start and the LED will turn on. You can see this either via the Active popup at the right of Visual Designer or by switching to the schematic tab.



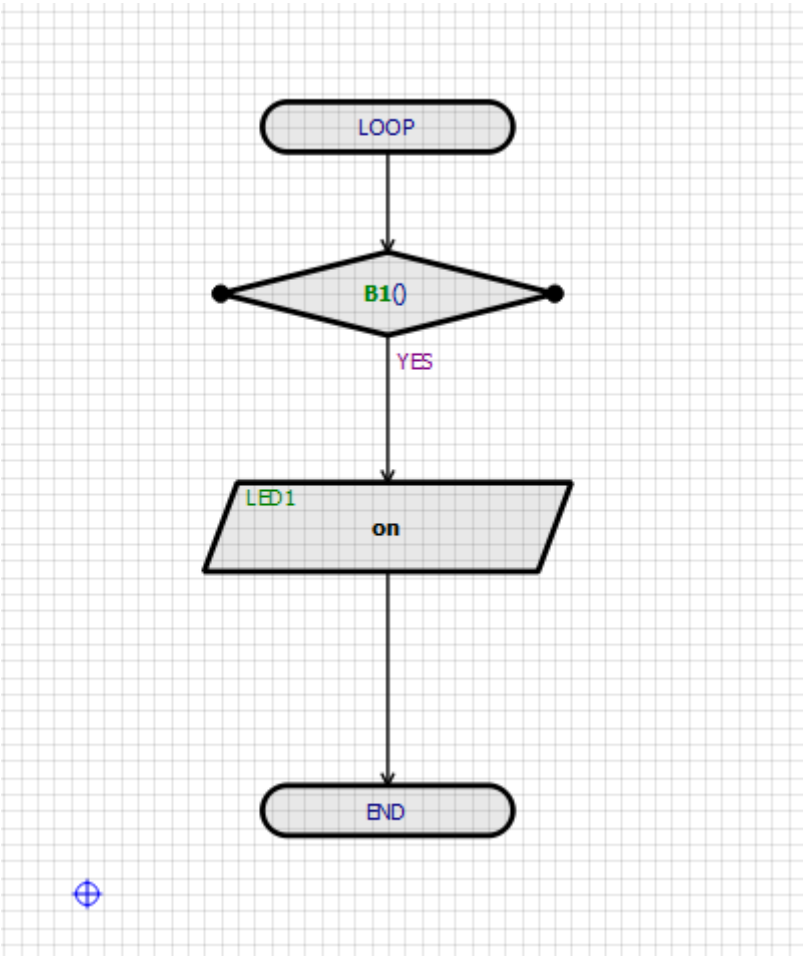
Press stop to exit the simulation and we'll extend our program to switch the LED on and off with a button press. You'll notice that the button has no methods in the project tree.



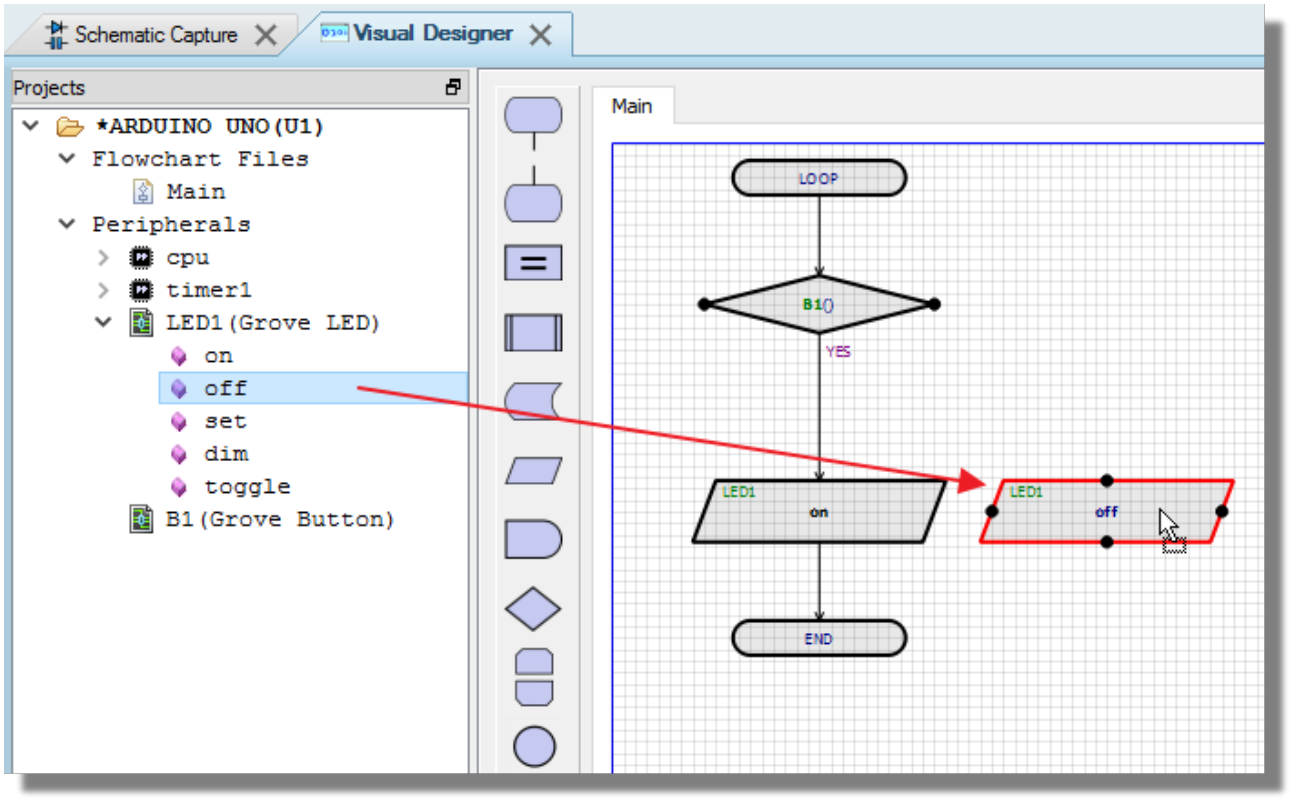
The only sensible information we need from the button is to know whether it has been pressed and for that we have what we call a 'sensor function'. A sensor function returns either TRUE or FALSE depending on whether the basic peripheral function is true or not. For example, the sensor function for a button will return TRUE when the button is pressed and FALSE if it isn't. Similarly, the sensor function for an LED will return TRUE when the LED is on and FALSE when the LED is off. Place the sensor function for the button by dragging and dropping from the button itself onto the loop function of the flowchart.



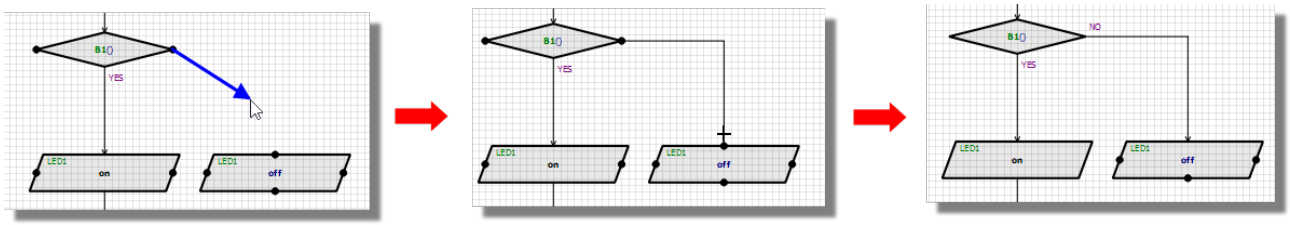
You will notice that it appears as a decision block. Sensor functions always return TRUE or FALSE and a decision block allows us to split our code into two conditional paths. Your loop routine should currently look something like the following.



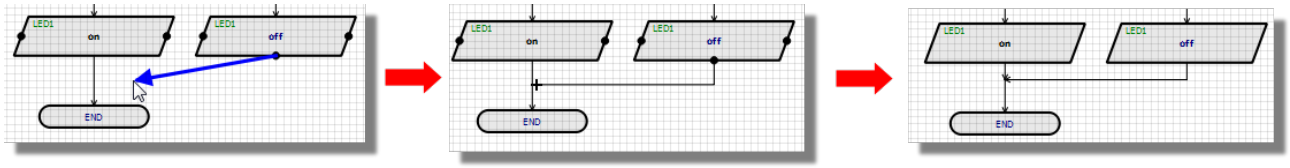
We are constantly testing the button to see if it is pressed and we are turning on the LED if it is. However, we still need to tell the program what to do if the button is not pressed. We want the LED to turn off so lets drag and drop that method into space alongside the ON operation.



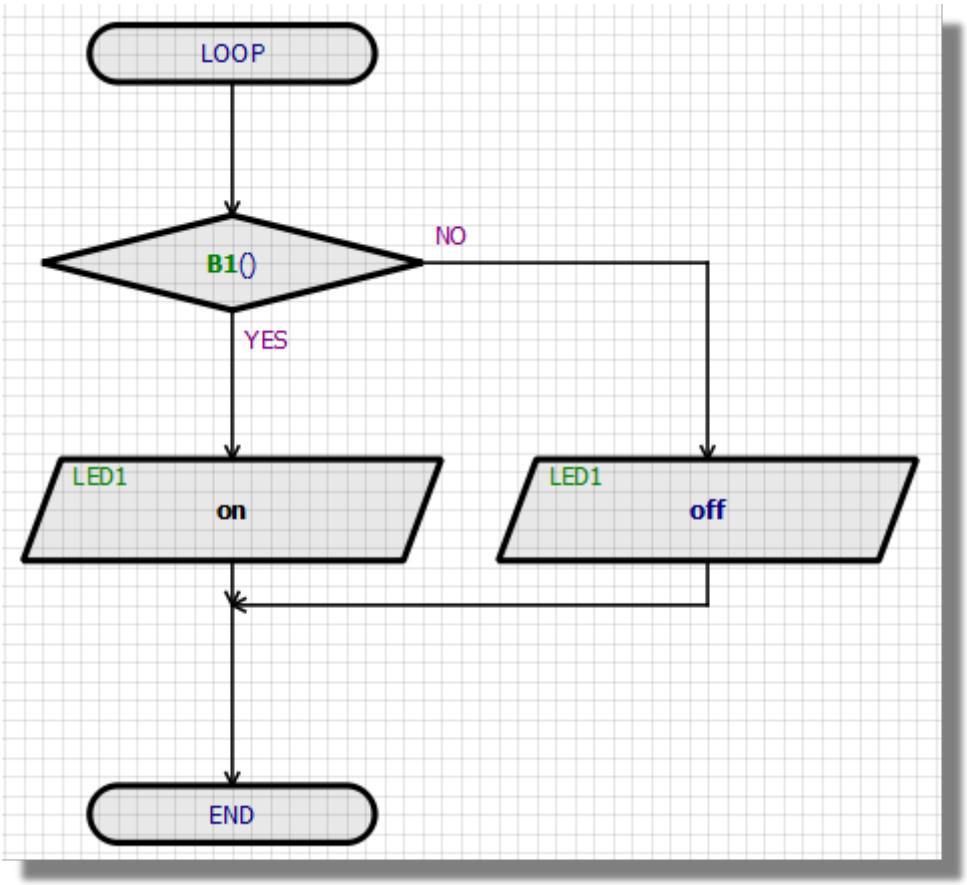
Now, we need to wire from the decision block to the top of the OFF command. To place a flowline click on an output node (the decision block), move the mouse to an available input node (top of the OFF routine) and click the mouse again.



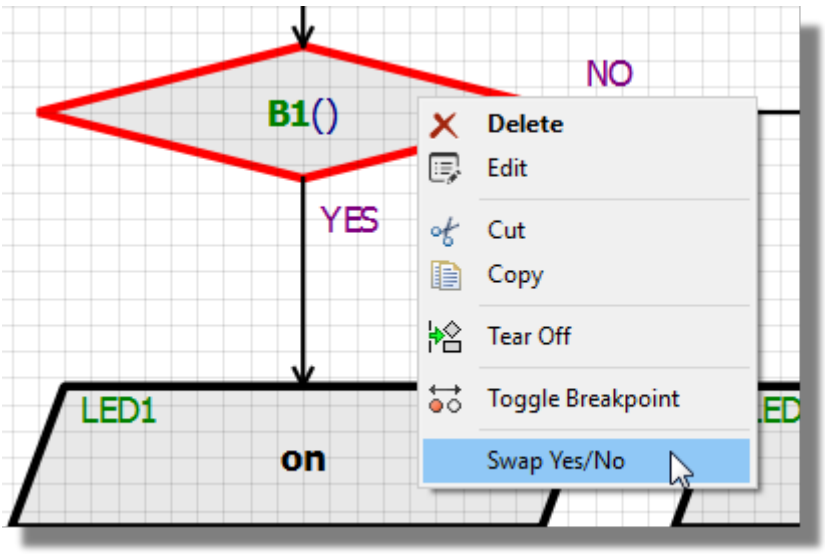
Similarly we need to connect the flow from the bottom of the OFF command back into the main loop.



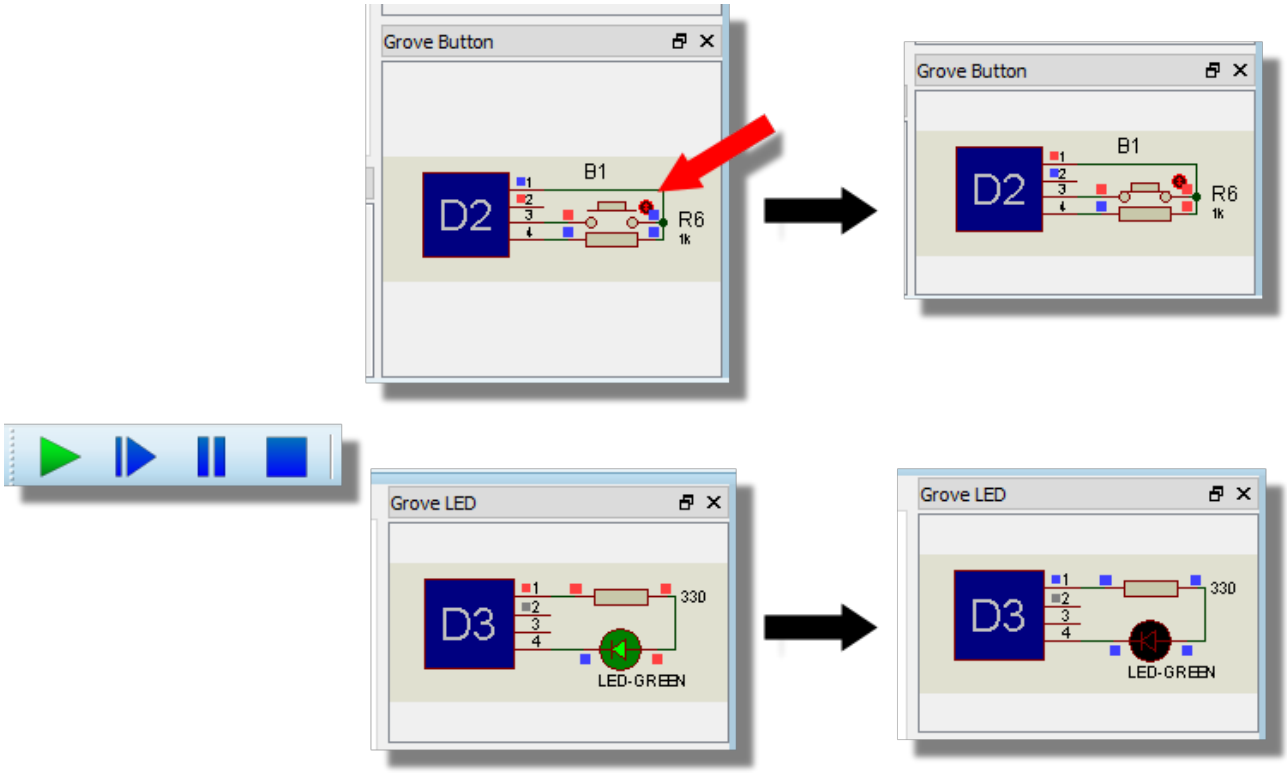
Your program should now look like the following



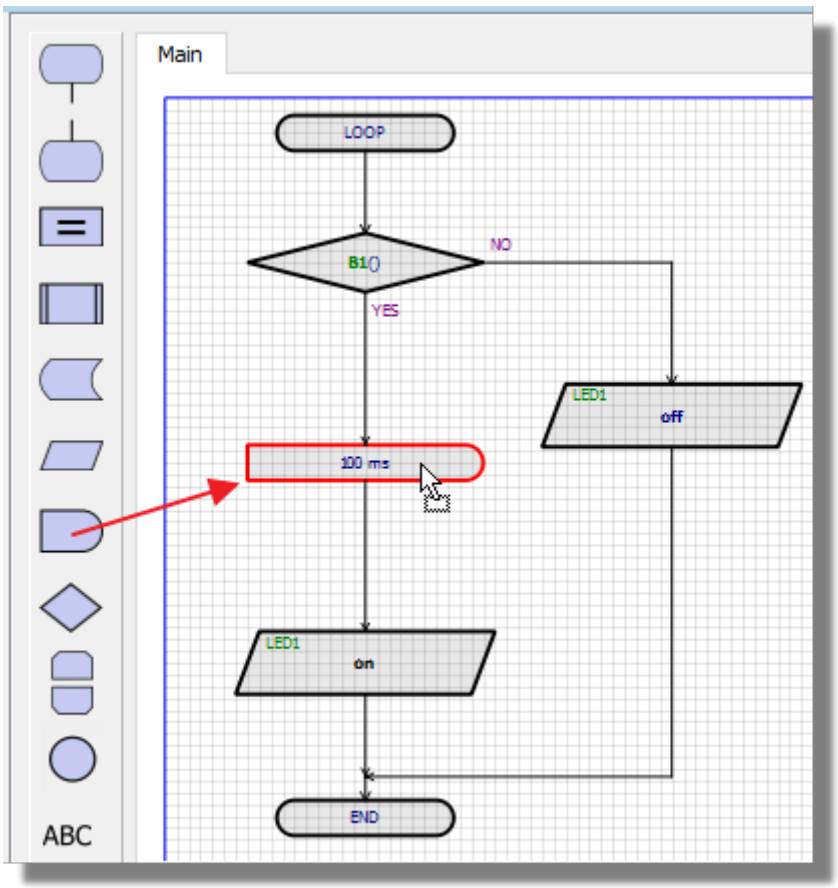
Note that the decision block has two labels indicating which code path is followed if the decision resolves to TRUE (YES) and which code path is followed if the decision resolves to FALSE (NO). If these are the wrong way around you can switch them by right clicking on the decision block and selecting the swap command from the context menu.



Having made this change, our program now turns the LED off when the button is down and on when the button is released. To compile and test, first press the play button and then use the mouse to press the button down.



You can experiment further here with changing the decision or swapping in the toggle method. You might also find it useful to add a delay.



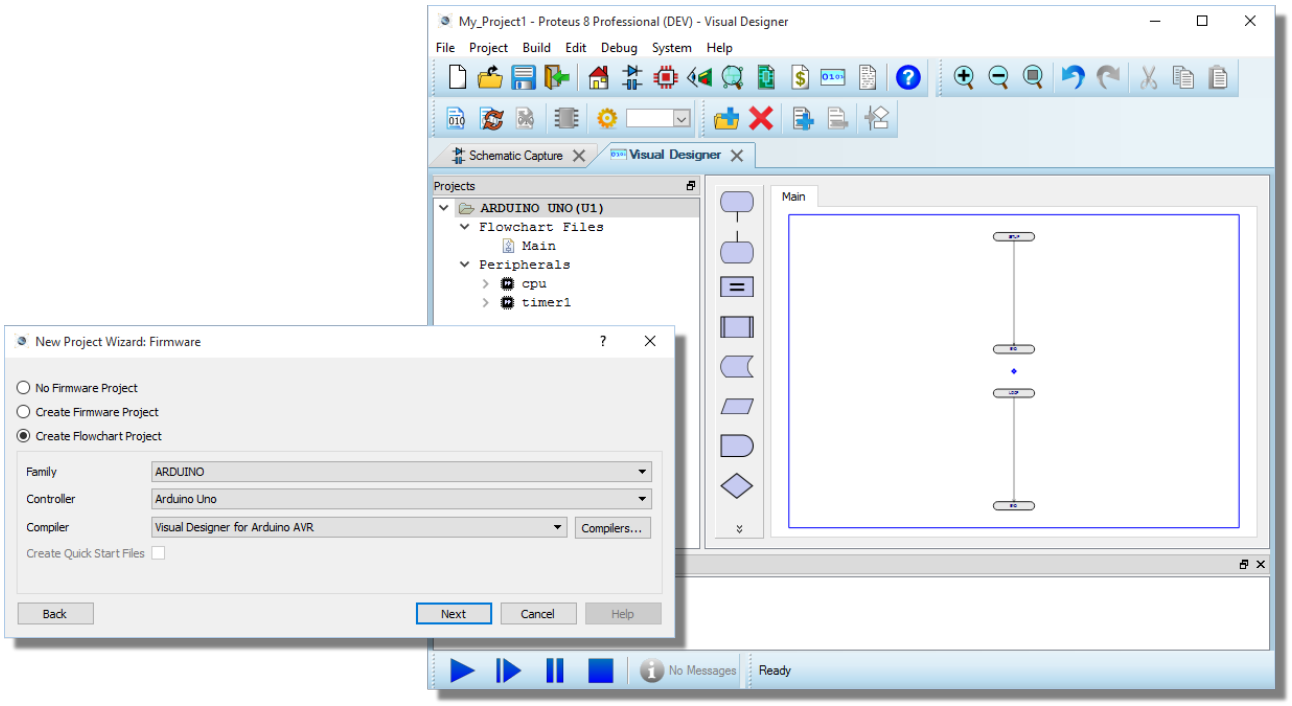
See Also:

- The next tutorial will introduce interactive simulation and basic debugging skills.

TUTORIAL 2: LUX SENSOR

Introduction

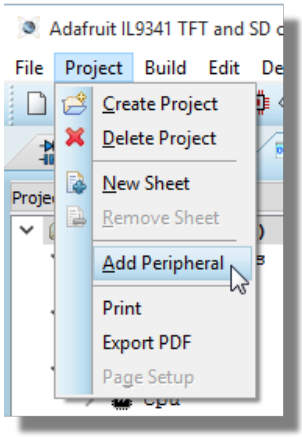
In this tutorial we are going to use a couple of Grove sensors and an LED to design a mini nightlight. Project setup is covered in the first tutorial so we'll assume that you have a blank Arduino Uno project in front of you.

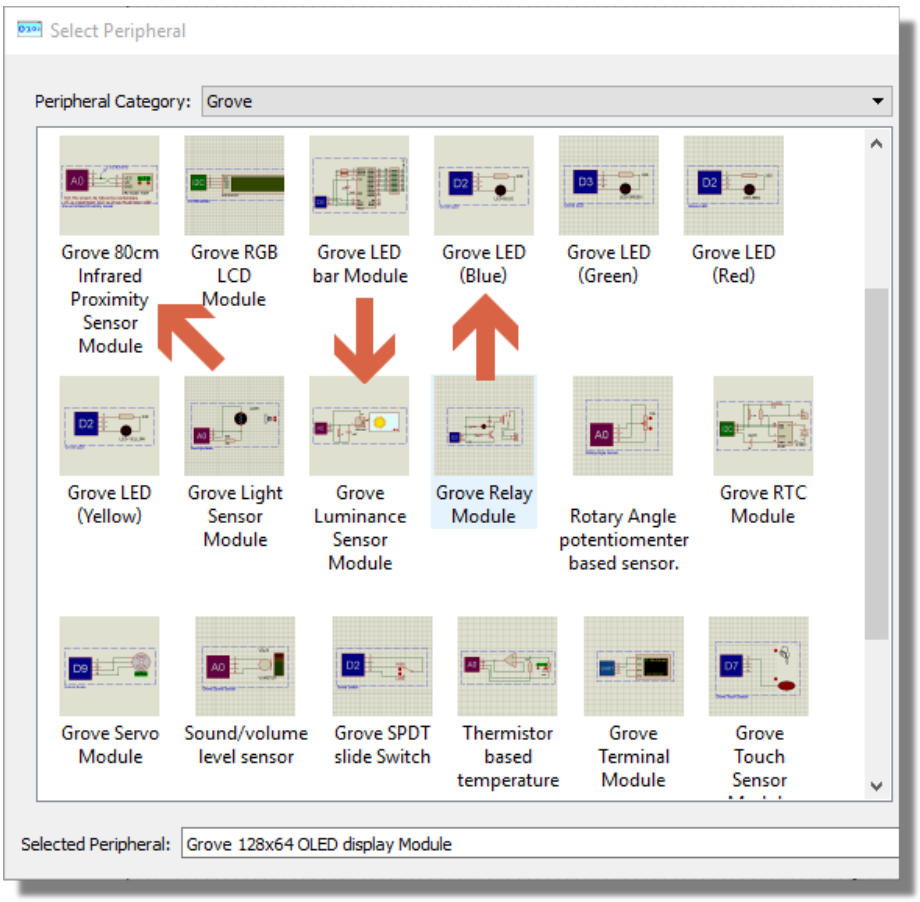


This tutorial is also covered by a short movie on the Labcenter Youtube Channel

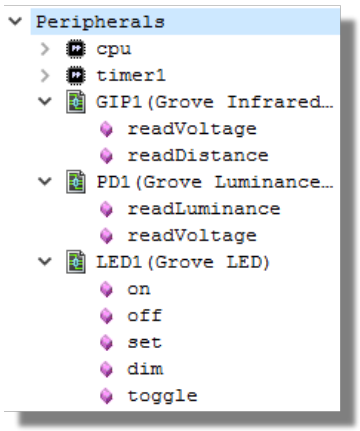
Adding Hardware

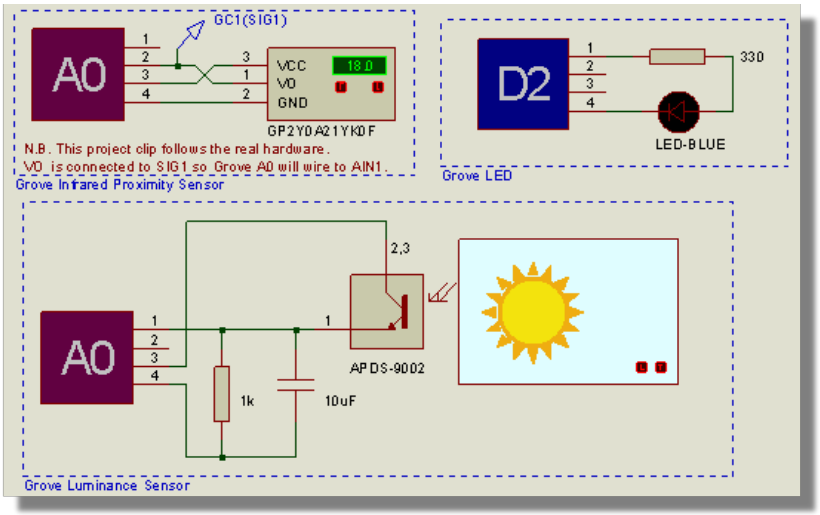
We'll use the Grove infra-red proximity sensor and the Grove luminance sensor along with a Grove LED. All of these can be picked via the Add Peripheral command on the Project Menu.





After they have been added to the project, you will see the methods for controlling the peripherals in the Project Tree and the 'virtual hardware' for the peripherals placed for you on the schematic.

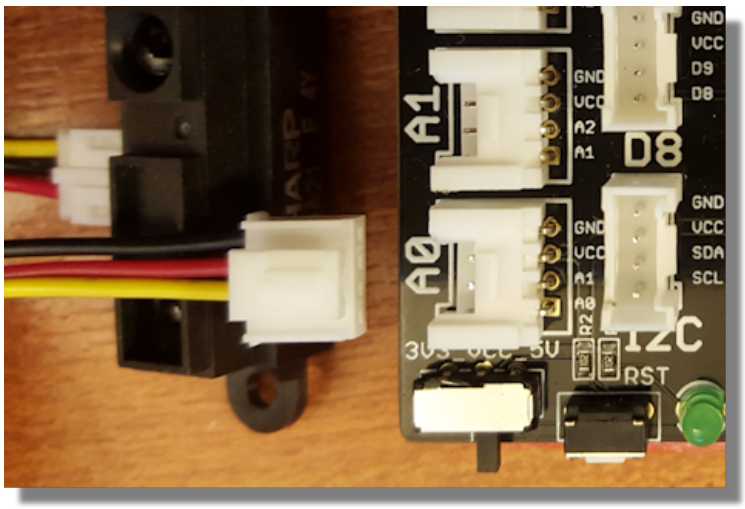




> ⚠ Note that the connectors are both labeled A0 and will need to be manually changed to be A0 & A2.

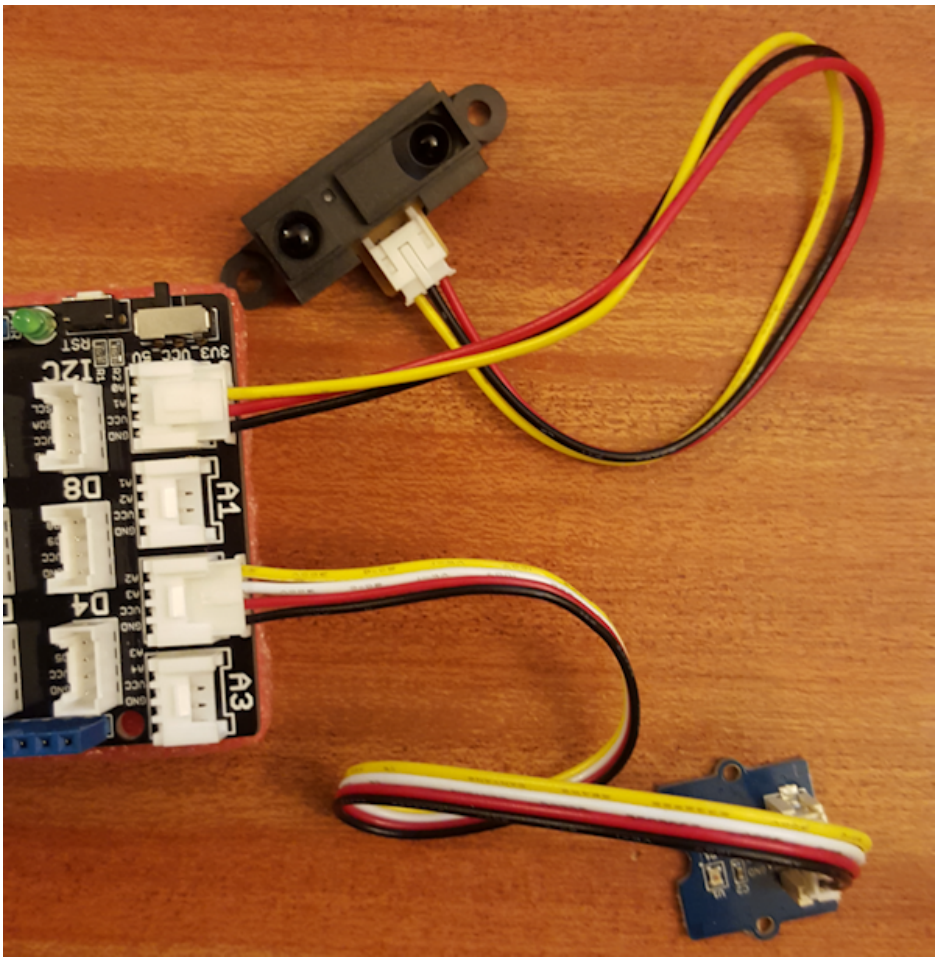
It is vitally important to configure the Grove connectors properly and this is often not as simple as it first appears. Let's take an example:

The real proximity sensor requires only three wires with the fourth left NC. Note therefore that, when plugged into socket A0 the sensor is connecting to analog pin A1.





This is fine, except if you then plug the luminance sensor into socket A1. The luminance sensor will use analog pin A1 on socket A1 and therefore clash. You should plug this sensor into socket A2.



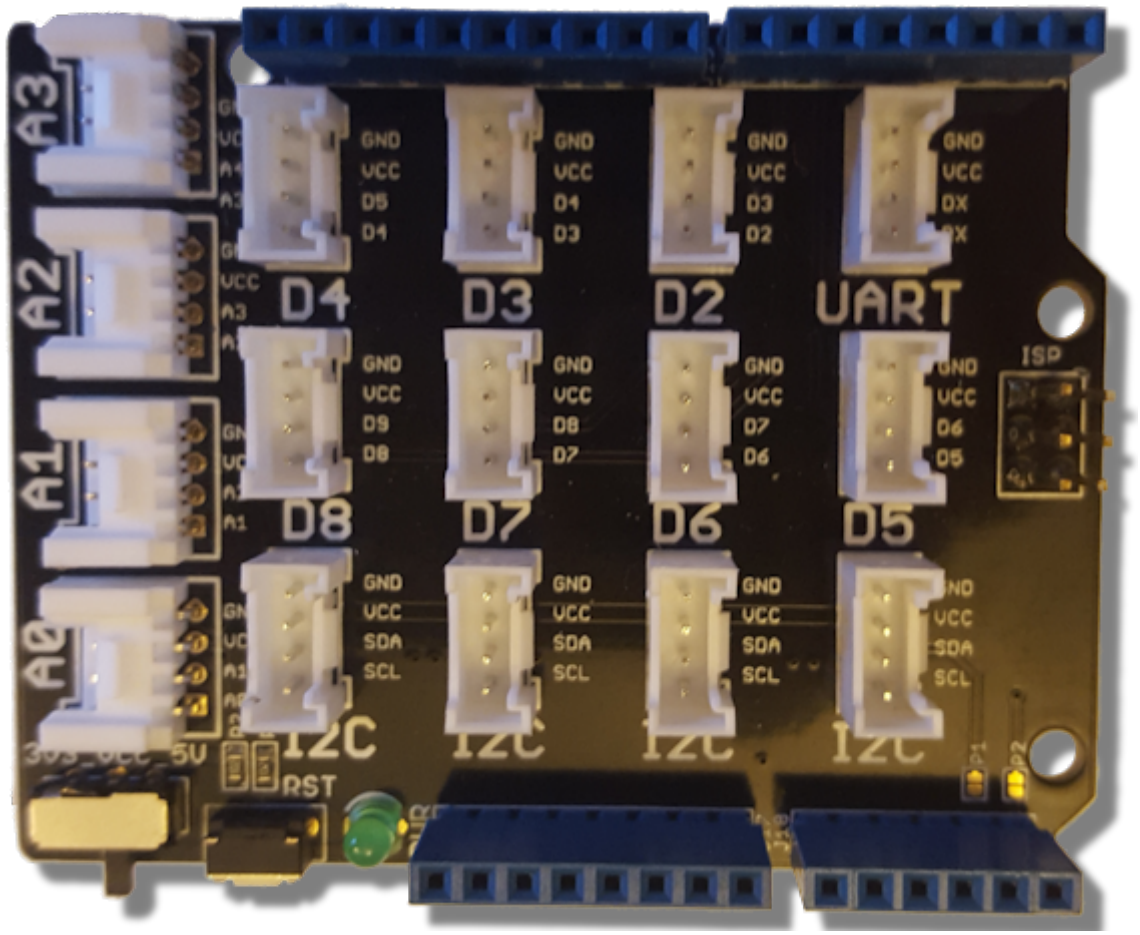
Golden Rule

Look closely at the Grove board to see the connections and double check against the sensors for conflicts.

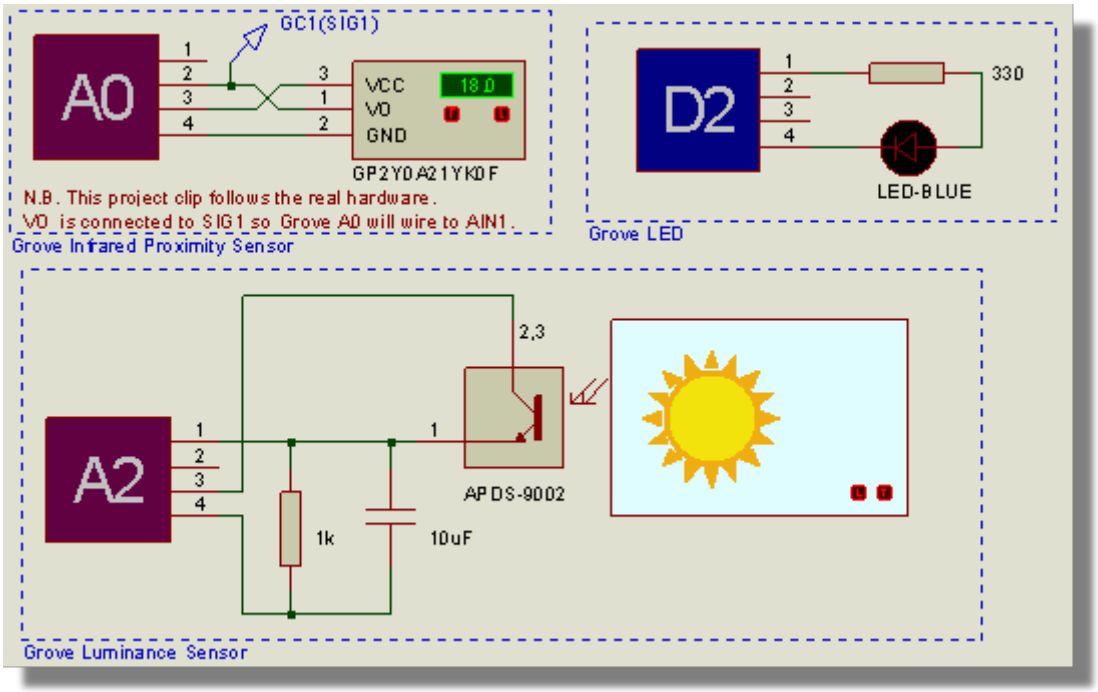
The sockets include connections through to two Arduino IO pins and there is a pin overlap. For example:

- Socket A0 can use pins A0 and A1
- Socket A1 can use pins A1 and A2
- Socket A2 can use pins A2 and A3.
- etc.

Digital sockets are exactly the same.



Our configuration for this project is therefore proximity sensor in socket A0 and luminance sensor in socket A2. The LED can be in any digital socket we like.



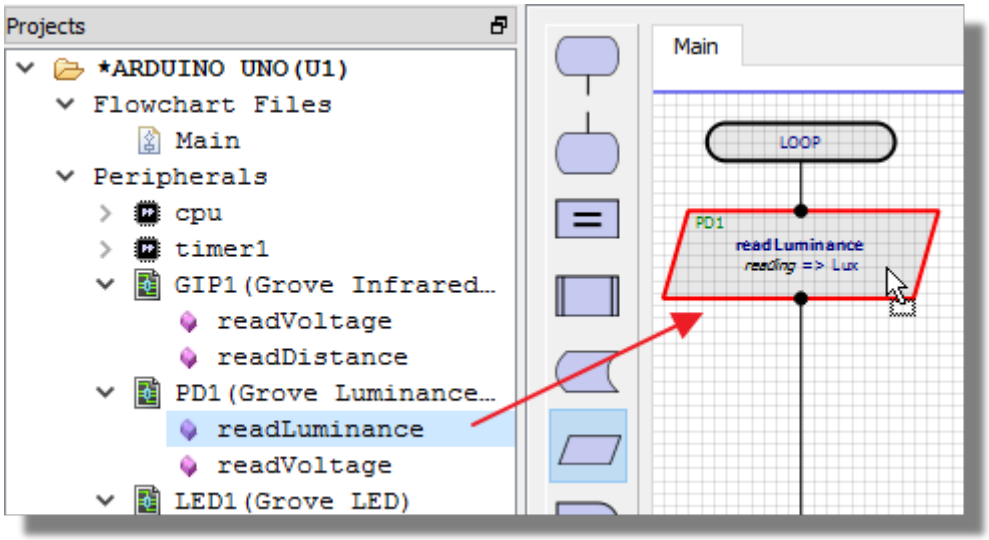
Designing the Program

In our nightlight mock up we are looking for our LED to light when the following are true:

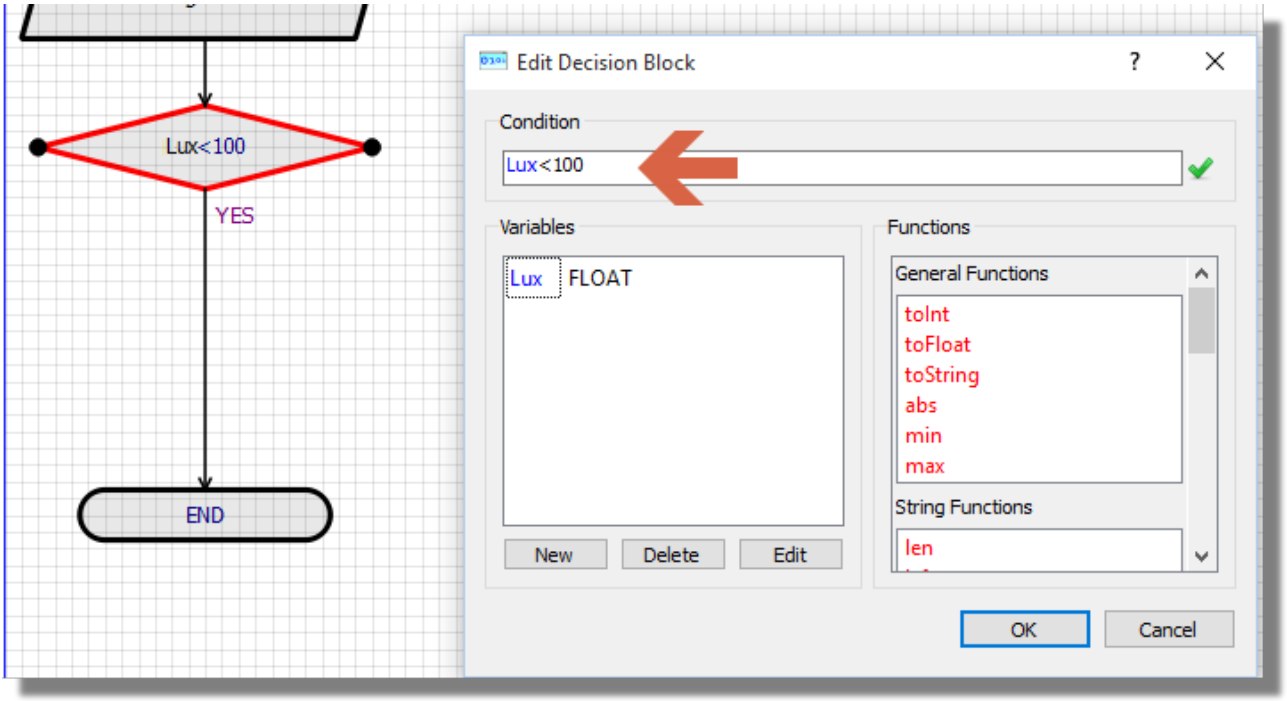
- It's is dark.

- Someone is nearby.

These are both decisions on the flowchart. Start by drag and drop the readLuminance() method onto the loop routine of the chart.



This sensor returns a lux value between 0 and 1000, with 0 being pitch black and 1000 max light. Let's then use our result in a decision block and set our initial test value to around 100.

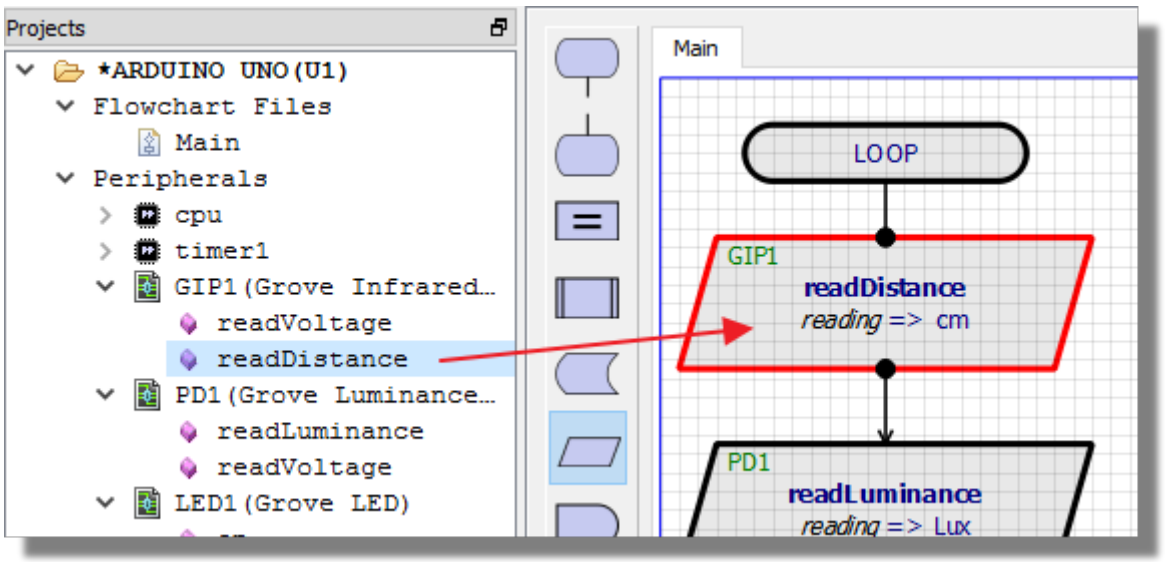


The datasheet for the APDS-9002 photo sensor used in this Grove module provides data on lux values.

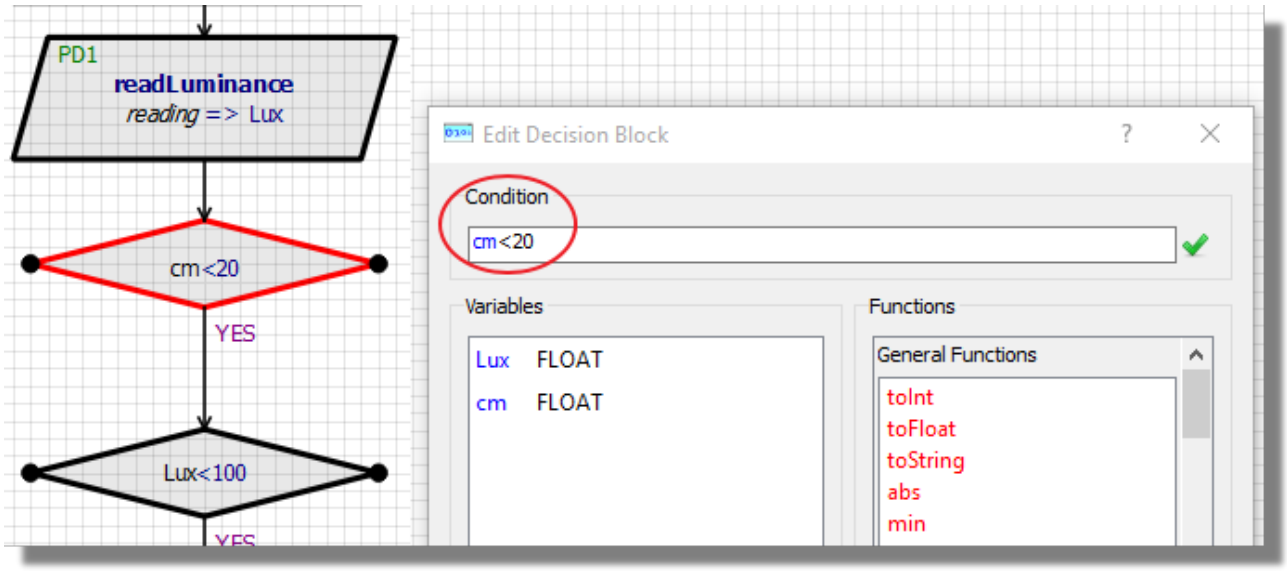
See Also:

http://wiki.seeedstudio.com/Grove-Luminance_Sensor/

For the proximity sensor, we want to read a distance so drag and drop into the top of the loop.



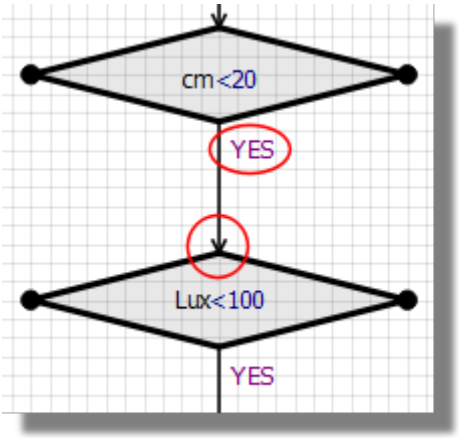
The proximity sensor can detect up to 80cm but we'll set our initial test to 20cm.



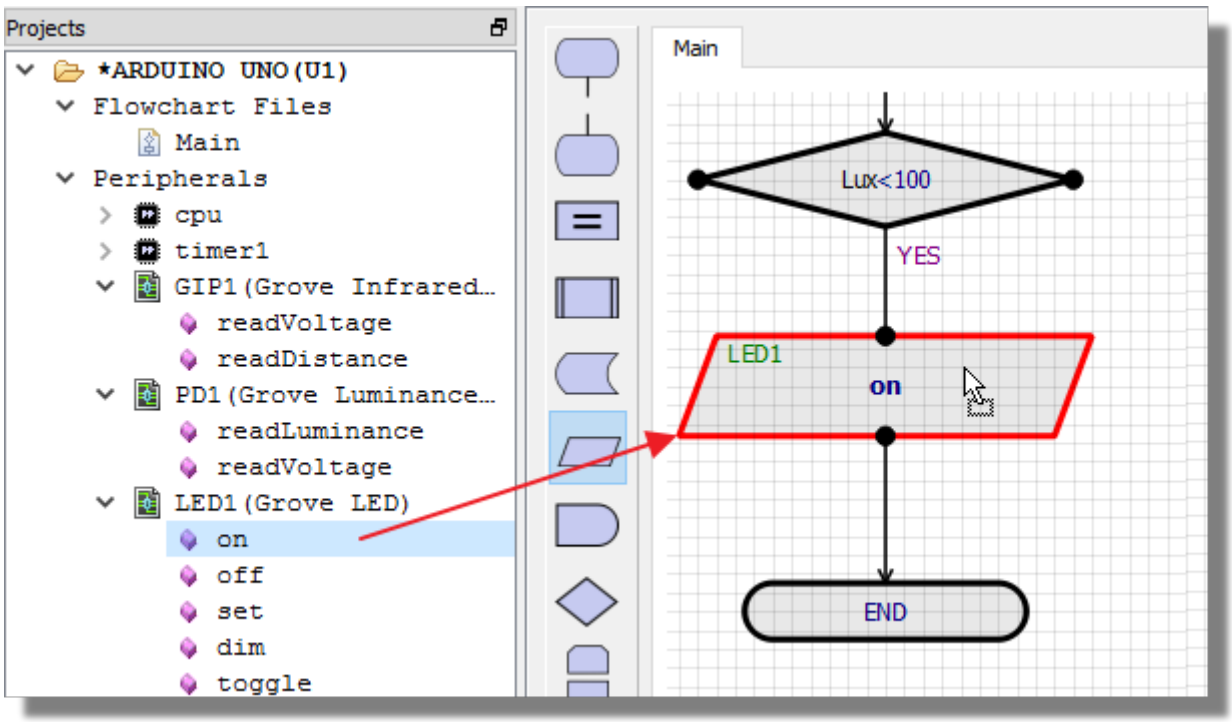
See Also:

http://wiki.seeedstudio.com/Grove-80cm_Infrared_Proximity_Sensor/

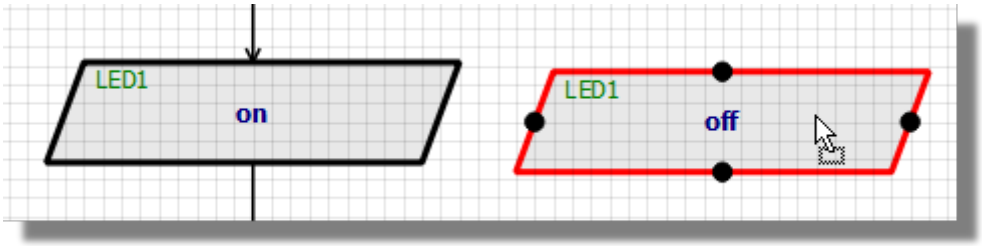
Since, we need both conditions to be true for the LED to turn on, make sure that the second decision is on the YES branch of the first decision.



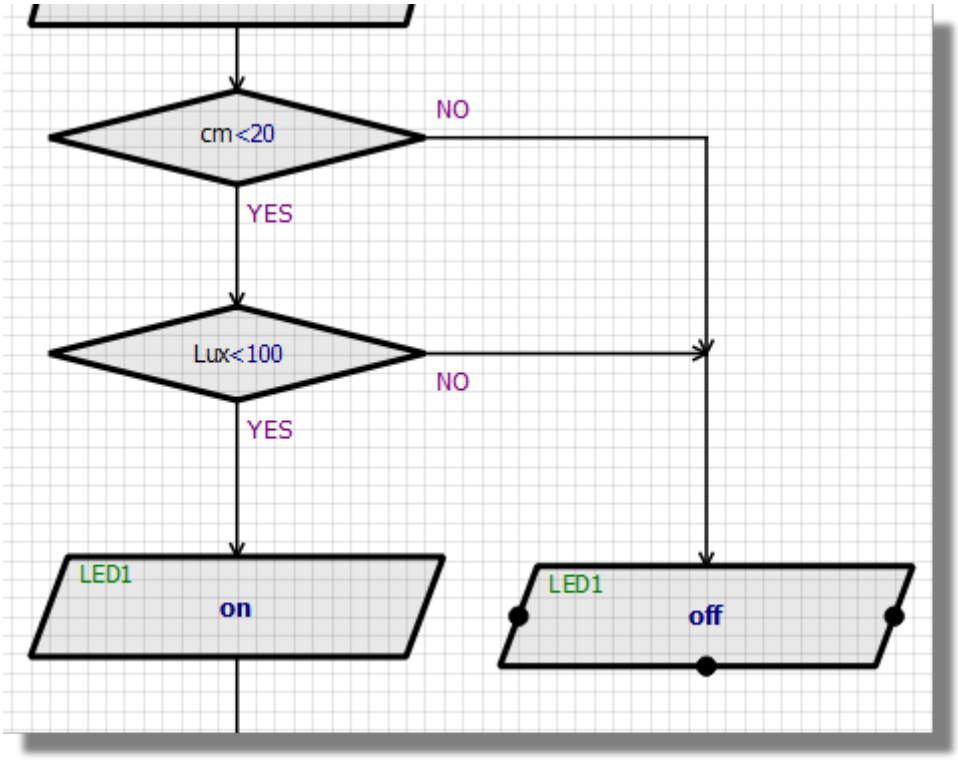
Next, drag and drop the led on method at the bottom of the YES branch.



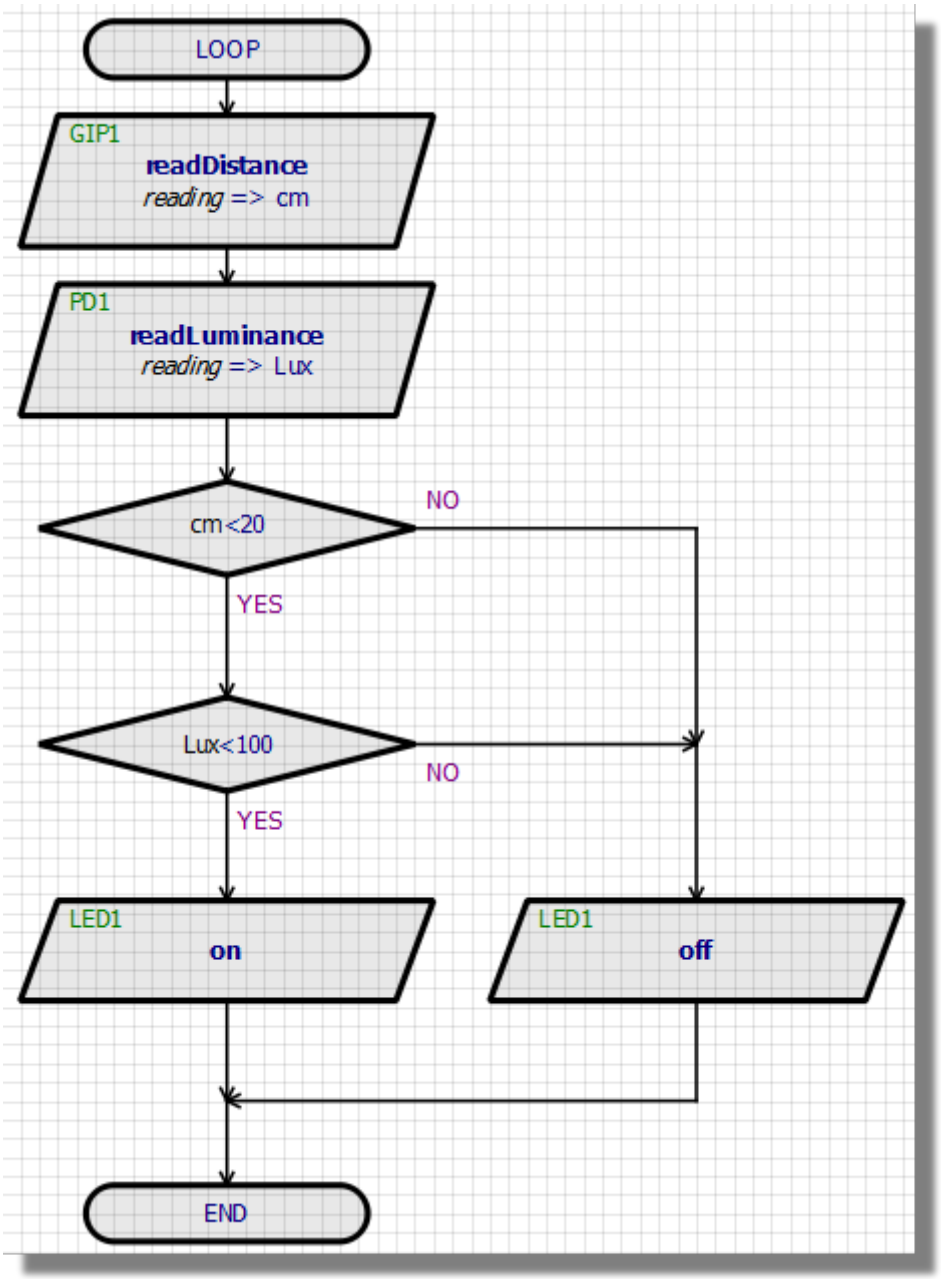
Finally, place the LED off method to the right of the main loop beside the LED on method.



Since we want the LED to turn off if either condition is false we can wire both NO branches from the decisions to this method.



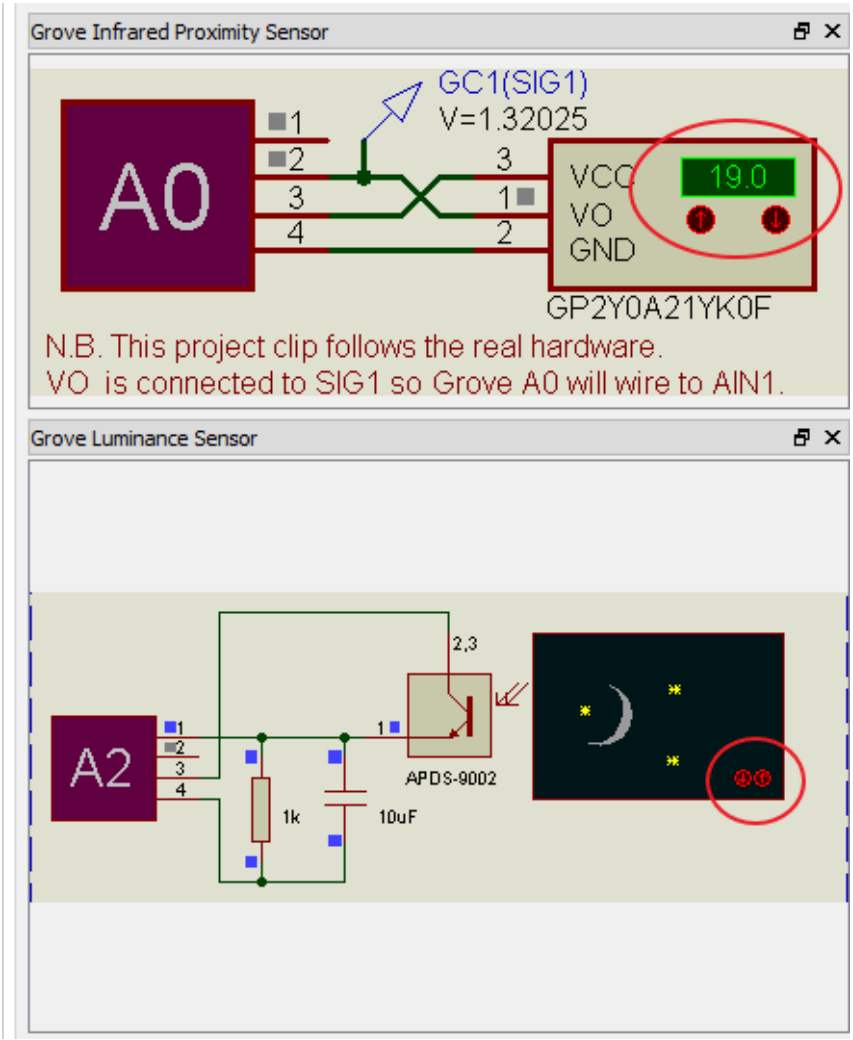
Finally, connect the bottom of the NO program branch back into the main program loop.



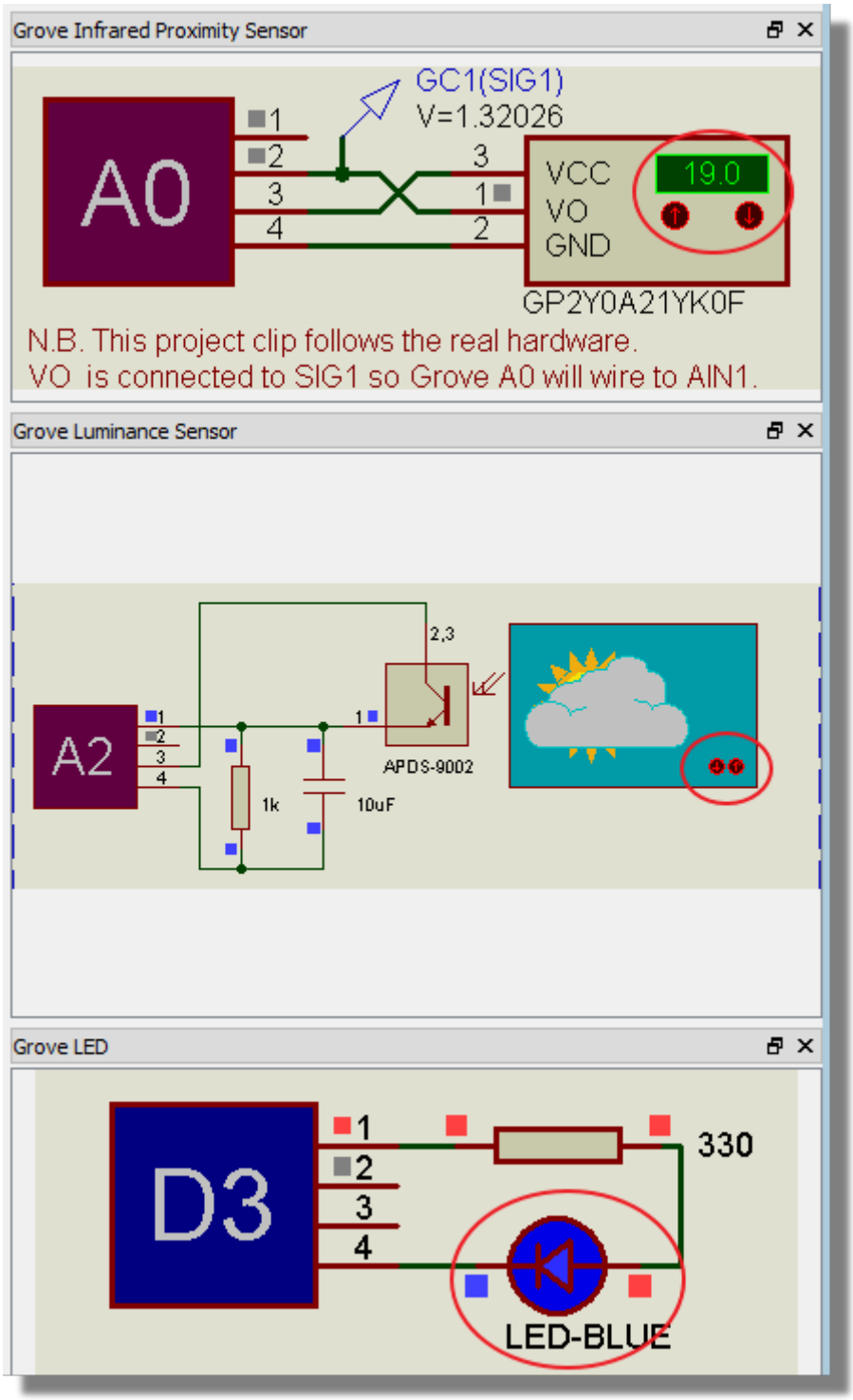
Simulating and Testing

To test our program, all we need to do is press the play button and then adjust the distance display on the proximity sensor and the lux display on the luminance sensor.





The LED should turn on when the distance value is less than 20 and it's a little cloudy.

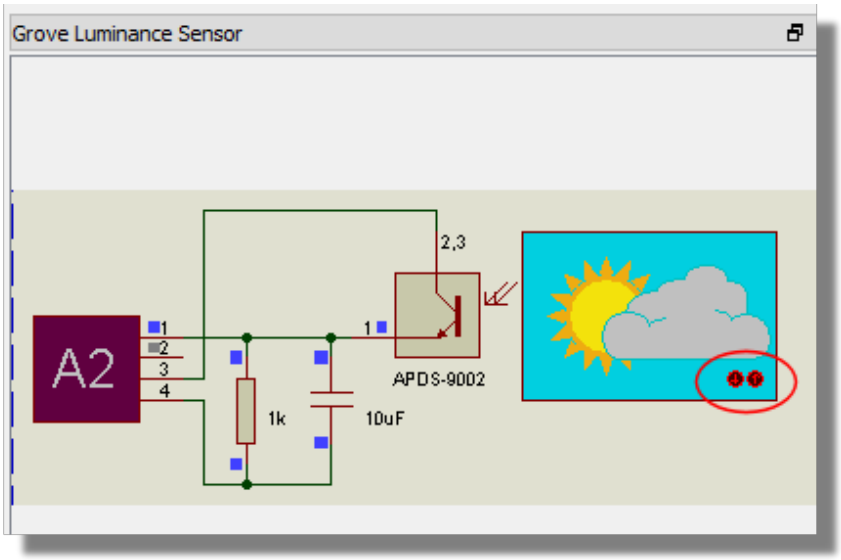


Pause the simulation and set a breakpoint on the LED off method.

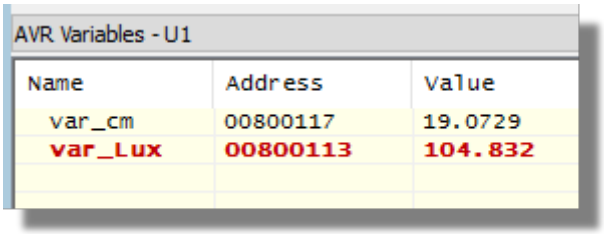


Press play again to run the simulation and change either the distance or the light until the breakpoint triggers

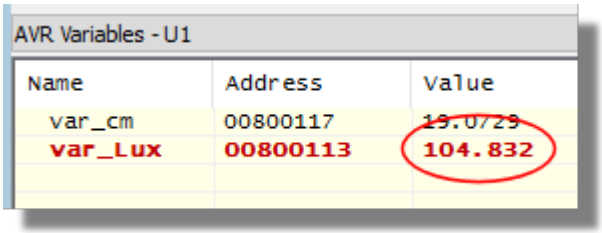


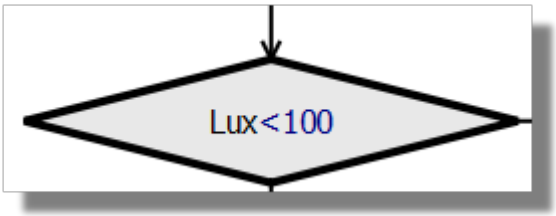


You'll notice that both our variables are available in the variables display.

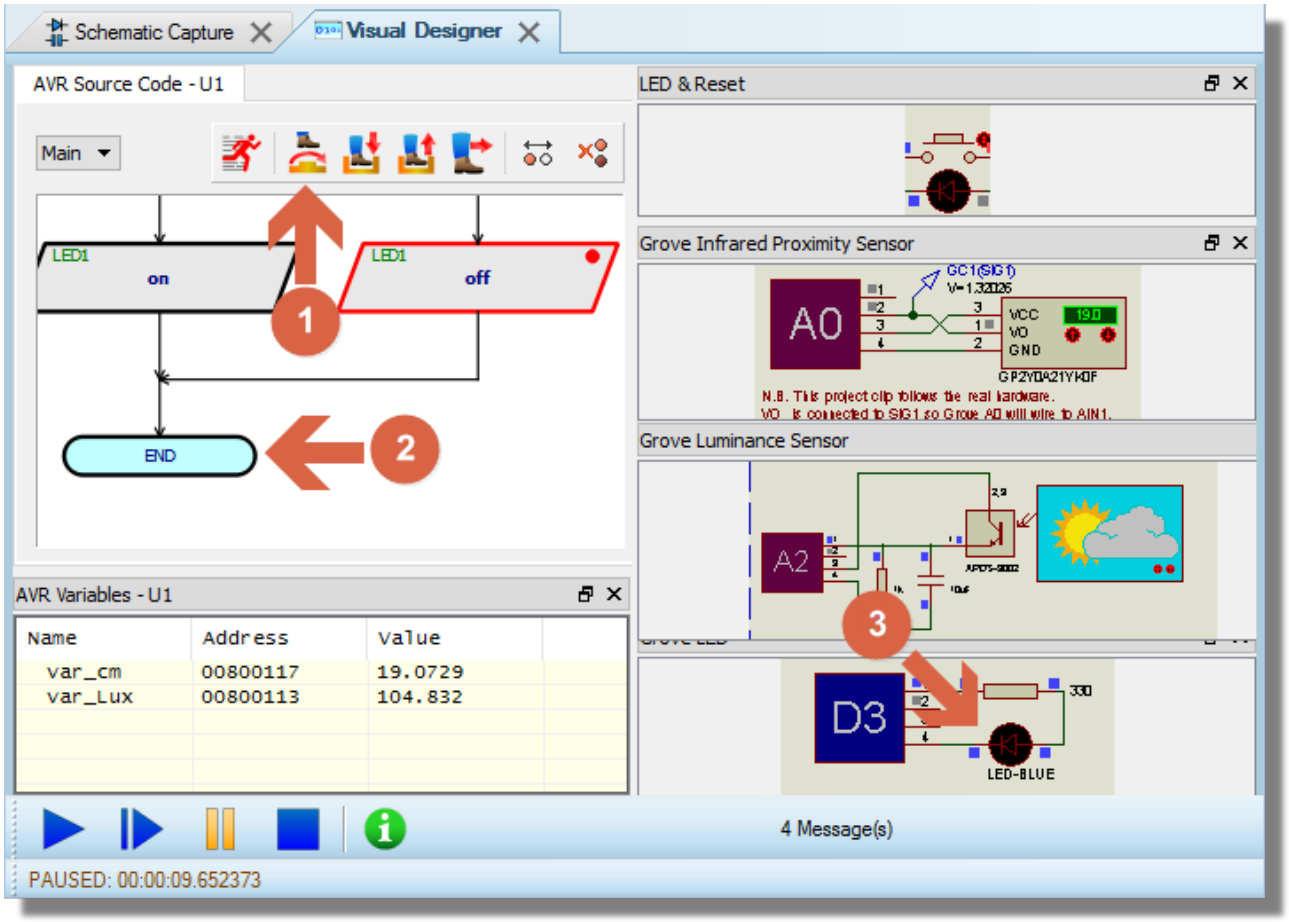


Comparing these values to the tested decisions will tell us which condition has failed.





As you single step via the icons at the top you'll notice the LED turn off.



1, Single step once. 2, Notice the END block is highlighted. 3. LED has turned off

See Also:

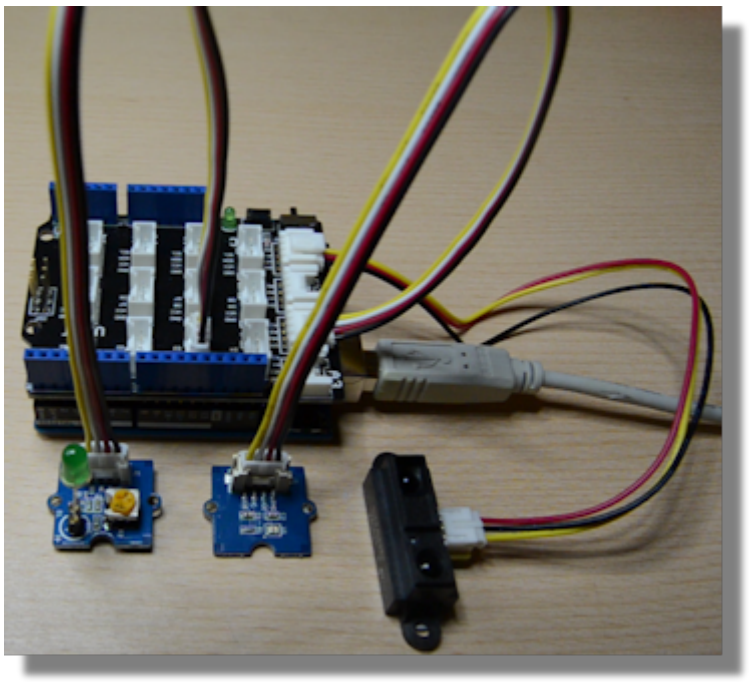
Debugging Skills

Programming

Since everything seems to working in good order, the remaining task is to plug in the Arduino and program the real hardware. The process here is:

1) Plug the Grove sensors into the Grove baseboard in the same connectors as on the schematic. See also the golden rule discussed earlier.

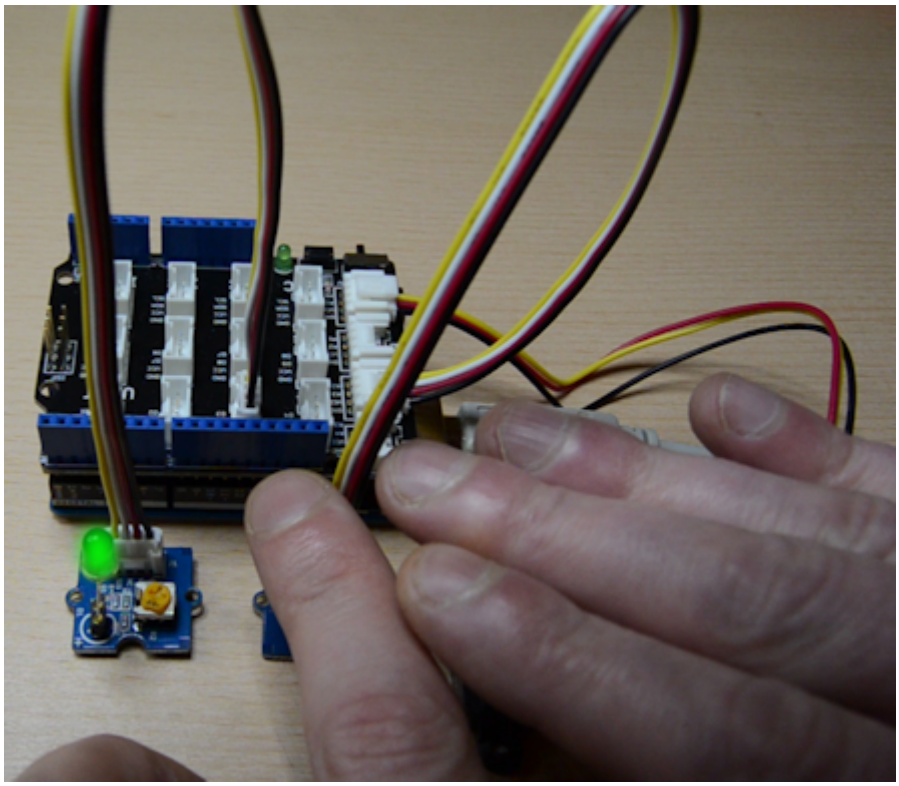
2) Connect the baseboard to the Arduino and connect the Arduino to the PC. Until you have something like so:



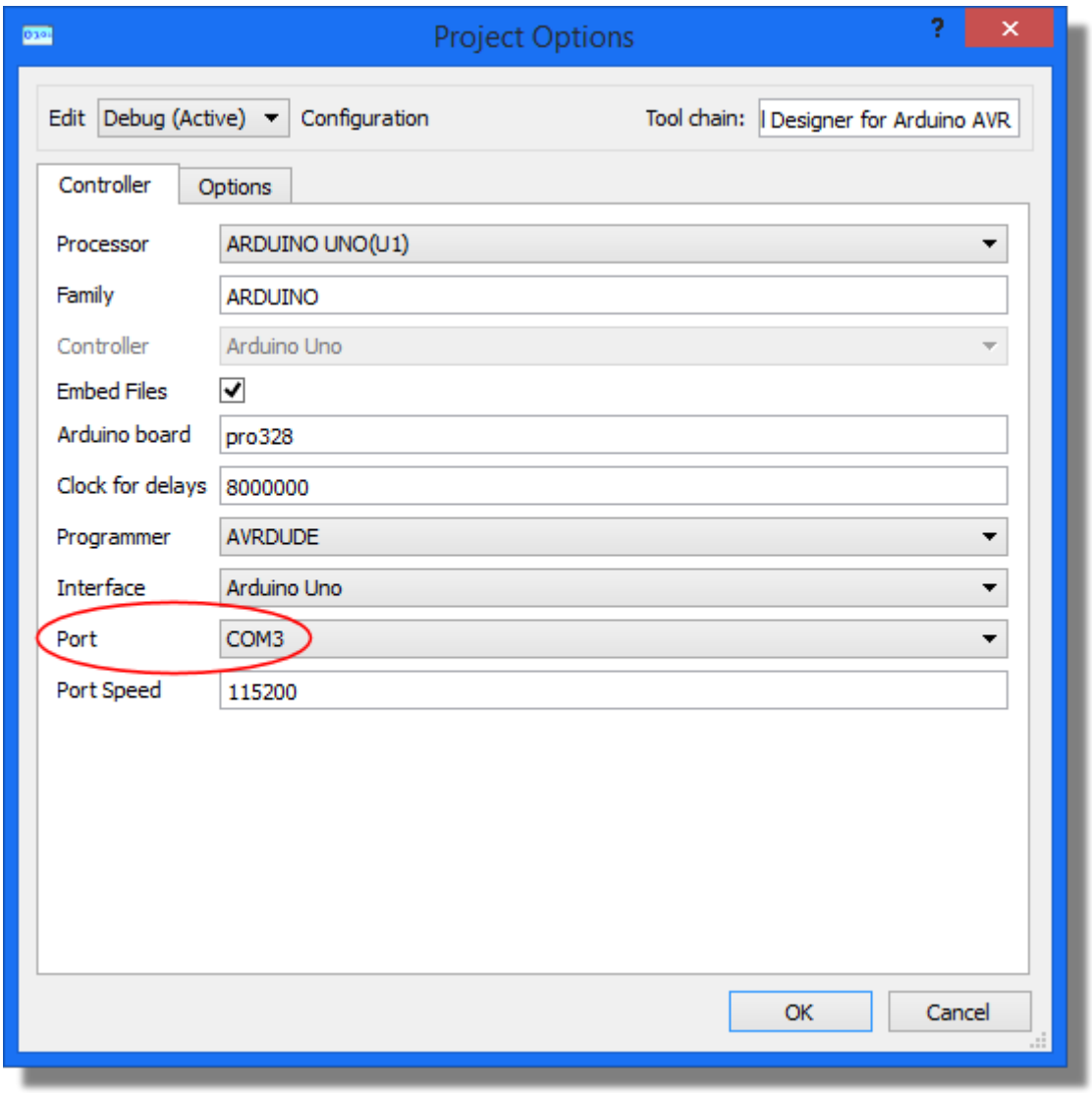
3) Click the program upload button in Visual Designer.



4) Test in hardware.



If the programming fails, click on the project settings icon (next to the icon shown above) to make sure the COM port and settings are correct.


Project Options
Edit
Debug (Active)
Configuration
Tool chain:
l Designer for Arduino AVR
Controller
Options
Processor
ARDUINO UNO(U1)
Family
ARDUINO
Controller
Arduino Uno
Embed Files
Arduino board
pro328
Clock for delays
8000000
Programmer
AVRDUDE
Interface
Arduino Uno
Port
COM3
Port Speed
115200
OK
Cancel

TUTORIAL 3: DATA STORAGE

Introduction

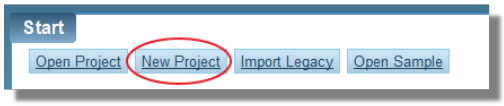
This tutorial introduces data resources and the storage model. We'll be creating a program to display a bitmap on an SD card onto the TFT display, although the same principles would hold true for audio (.WAV) files on the Wave Shield.

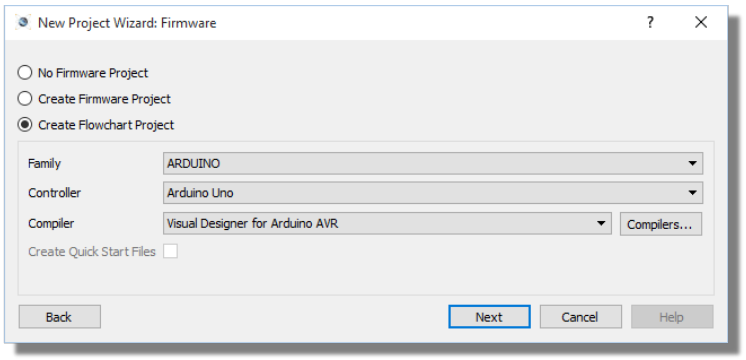
It is assumed as this stage that you have worked through the other tutorials and are familiar with general placement and connecting of flowblocks.

You can watch a short movie of this tutorial on the Labcenter Youtube Channel.

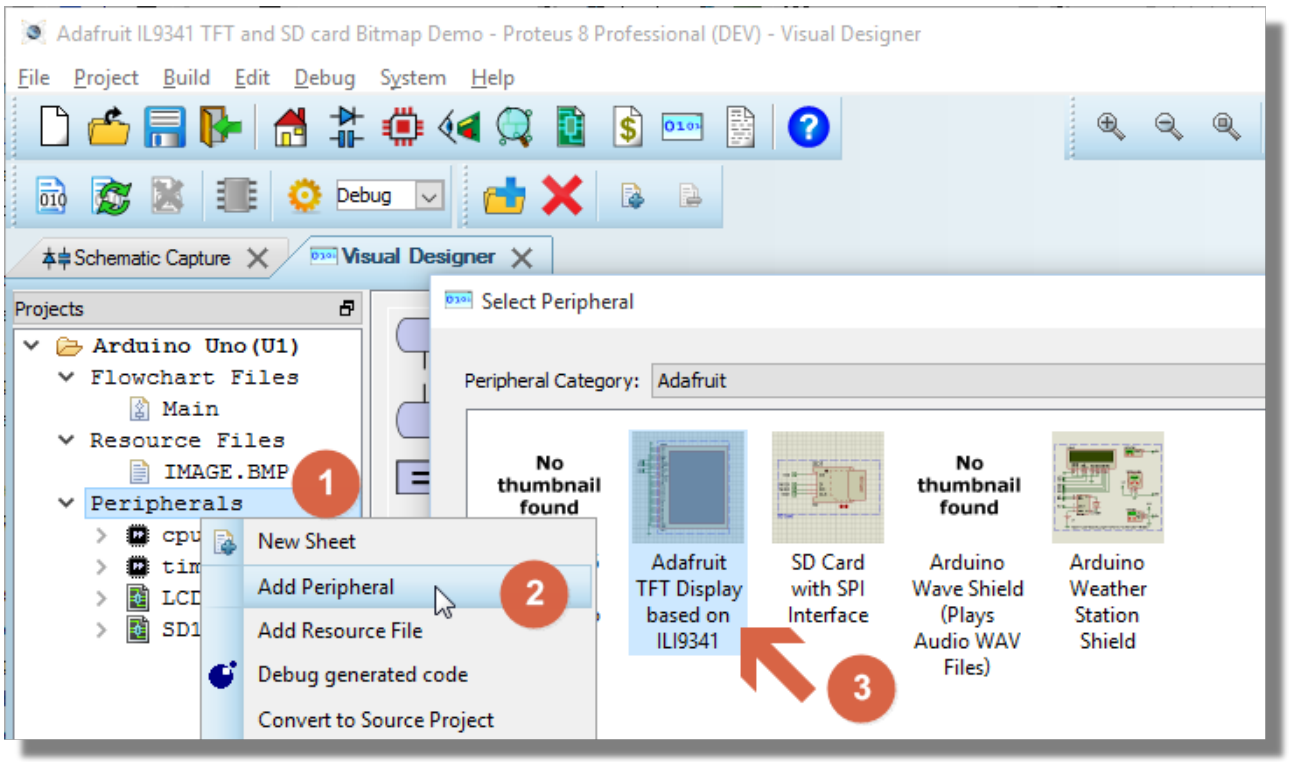
Setup

We'll set up this project in the usual way via the new project wizard and configure it as an Arduino Uno flowchart project.

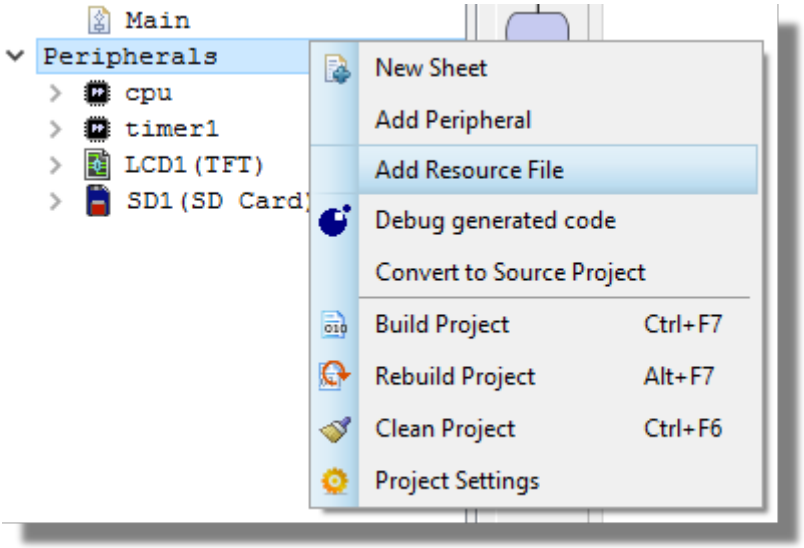




Next, we'll want to add our peripheral shield. We right click on the Project Tree and add the Adafruit TFT shield. Note that this is not a module for the Grove Shield but a completely different shield which contains both an SD card and a TFT display.



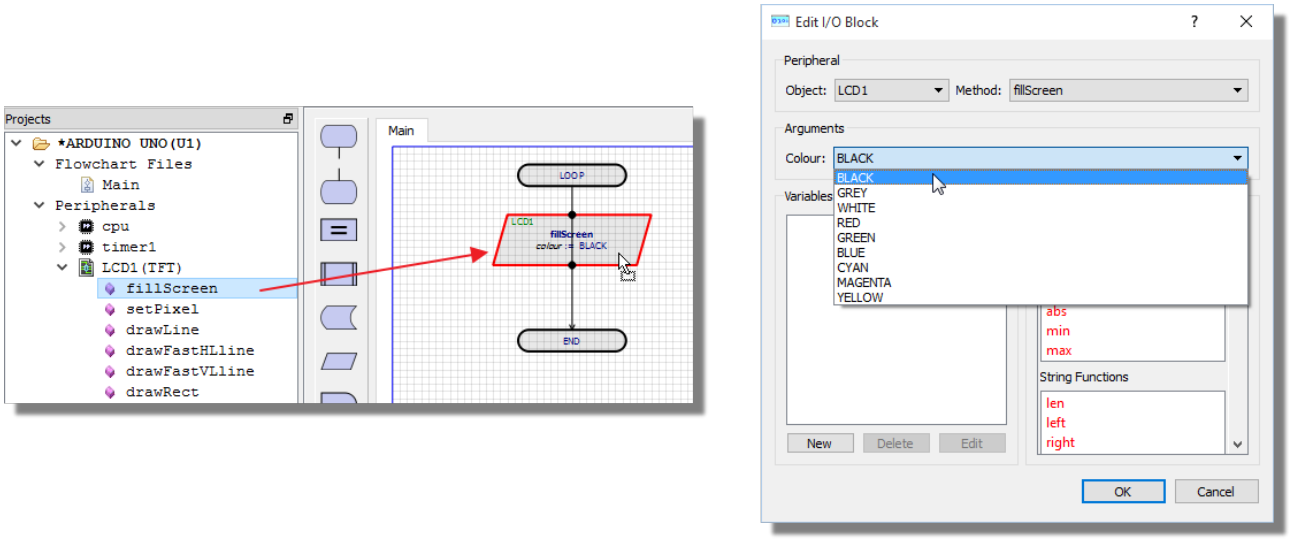
The last thing we need to do is add the resource. In our case this is the picture that we want to store in the SD card and display on the TFT display. We do this via the Add Resource command on the right click context menu.



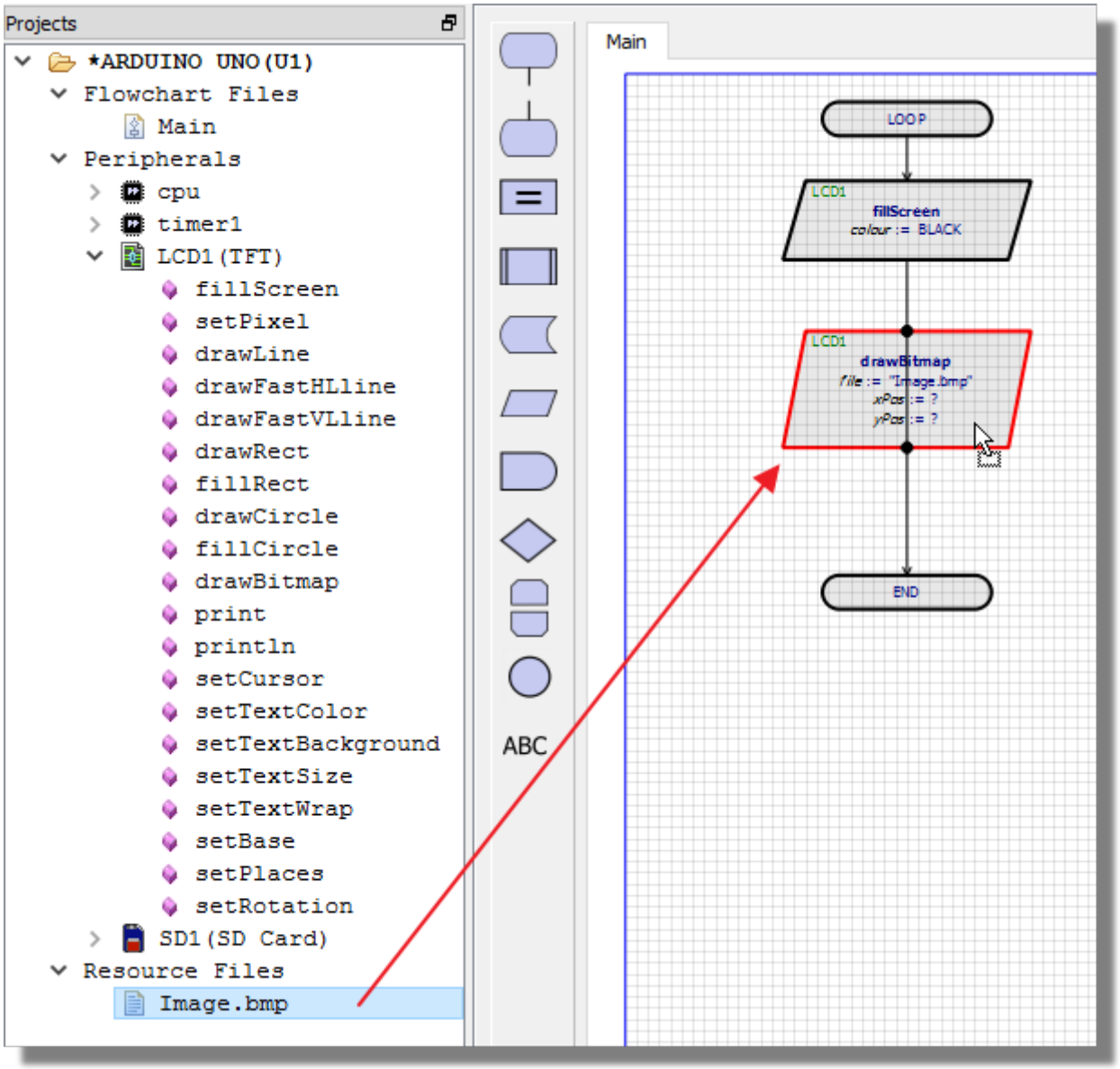
Be careful with the naming of resource files. The Arduino SD stack by default supports only 8.3 filenames in order to support FAT16 cards. This means that the namestem must be 8 characters or less and the extension must be 3 characters or less. For example, a picture called ProteusLogo.bmp will not load whereas renaming it to logo.bmp or image.bmp will work fine.

Designing the Program

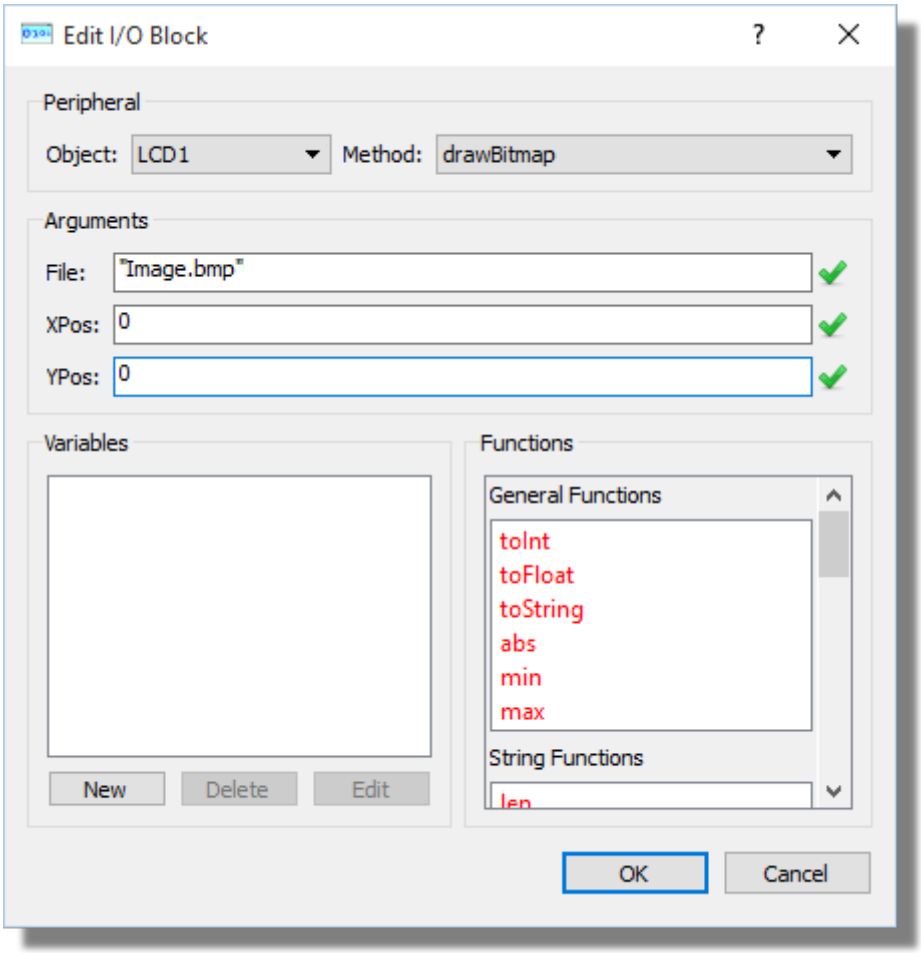
Now that we have all the pieces we need, the remaining job is to create the program. This could hardly be easier in Visual Designer. First, drag the fillSreen() method onto the top of the loop and set the fill colour to black.



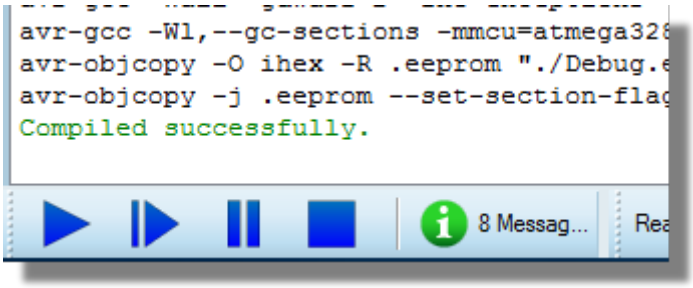
Next drag and drop the bitmap resource into the loop routine. Visual Designer knows that the only sensible target for the resource is the TFT display and it knows the method for rendering the resource is drawBitmap() and so it will automatically configure the flowblock for you.



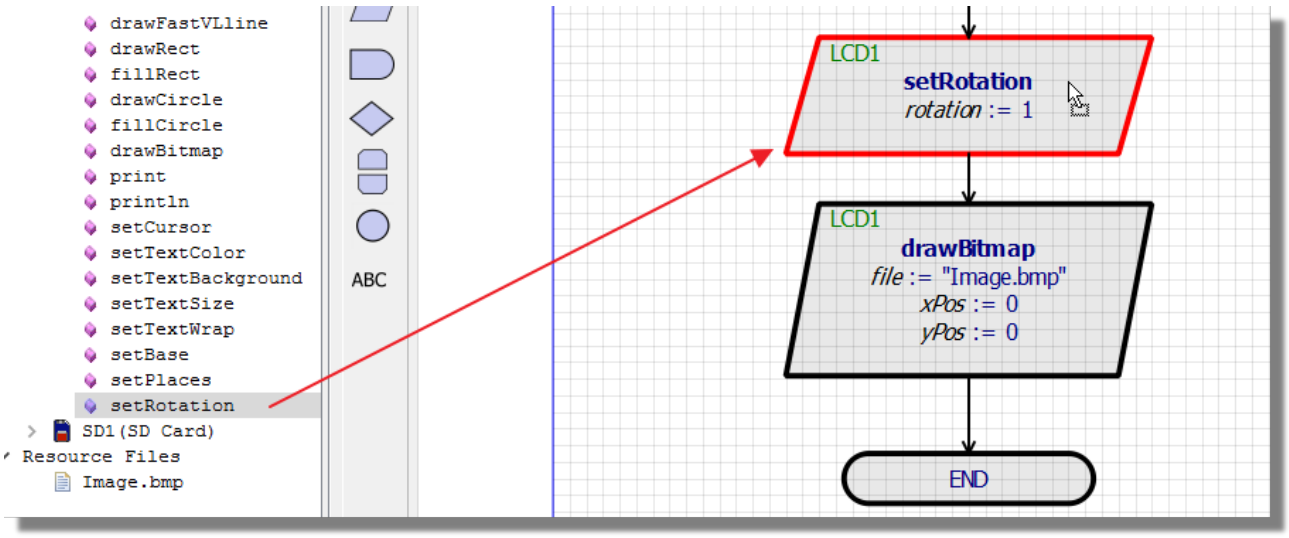
Now you need to edit the Image and set the xPos and yPos to be 0



Press the play button to compile and run the simulation.

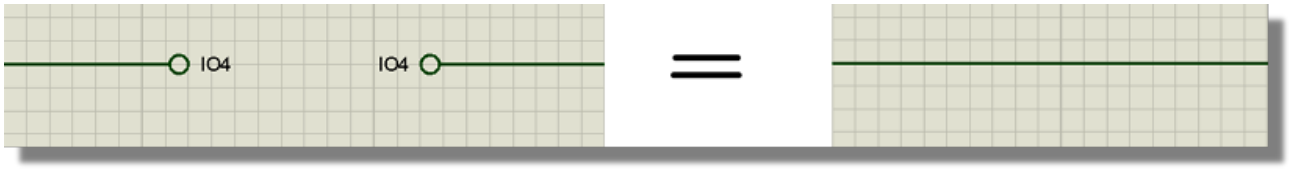


If necessary, you can configure the rotation in the setup routine via the same drag and drop method.



Taking Measurements

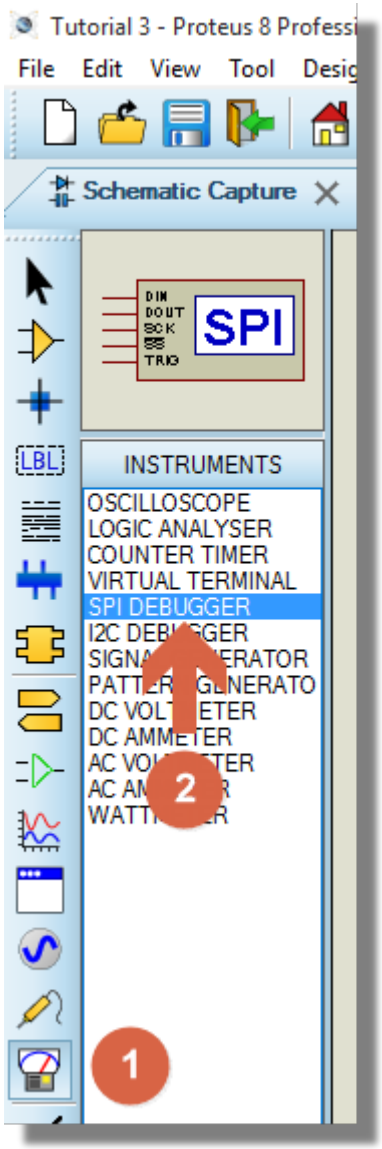
One of the nice things about Visual Designer is that it contains the Proteus schematic. This means that, if you want to, you can look a little deeper at what is happening in your design. For example, in our example the SD card is connected to the processor on the SPI bus and so we can place and wire a protocol analyzer to examine the SPI packets. In Proteus, all signals and waveforms travel through the wires, and two terminals with the same name have an 'invisible wire' between them.



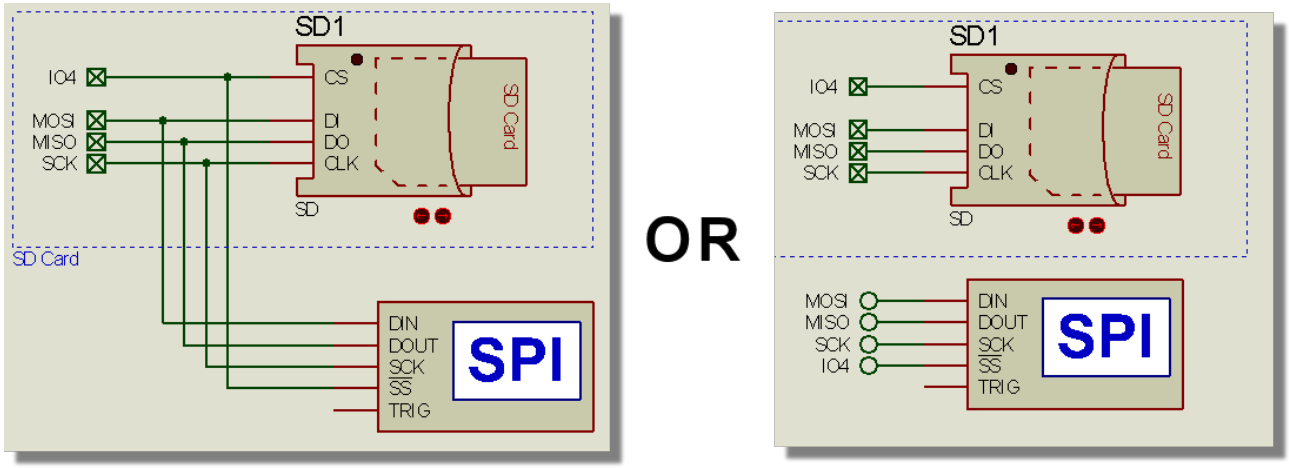
Two terminals with the same name are connected together as though they were wired.

Given this, the easiest way to connect the SPI terminal is to place one and wire it to terminals with the same names as the other SPI lines. The alternative is to connect up to existing wires.

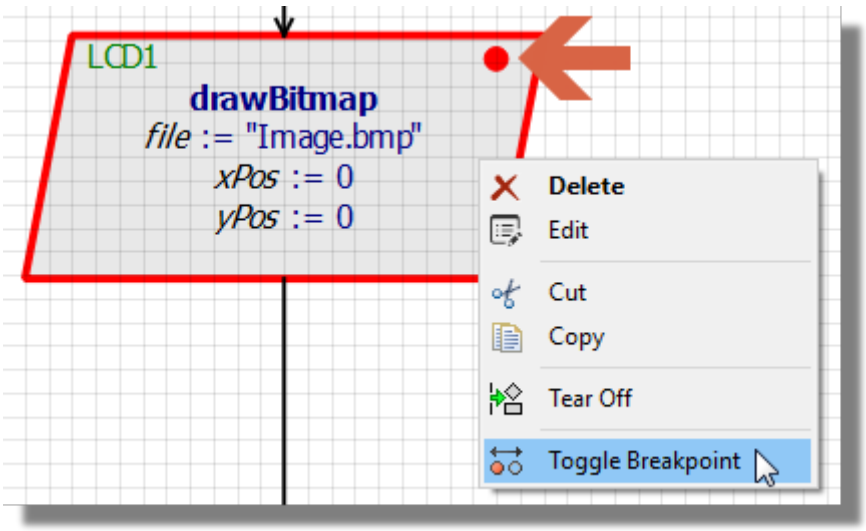
Placing a SPI Analyser:



Wiring SPI Analyser:



Next, set a breakpoint in the drawBitmap() command as this is where the majority of the SPI transmissions happen.



Now, when we run the simulation we'll see some initialisation chatter on the SPI bus until we hit our breakpoint. When we single step, we will see an awful lot of data being moved from the SD card corresponding to the bitmap being read by the processor. You can drill all the way down to bit level on any packet if need be.

The same basic process will work with any instrument (Oscilloscope, Logic Analyser, I2C Analyser etc.).

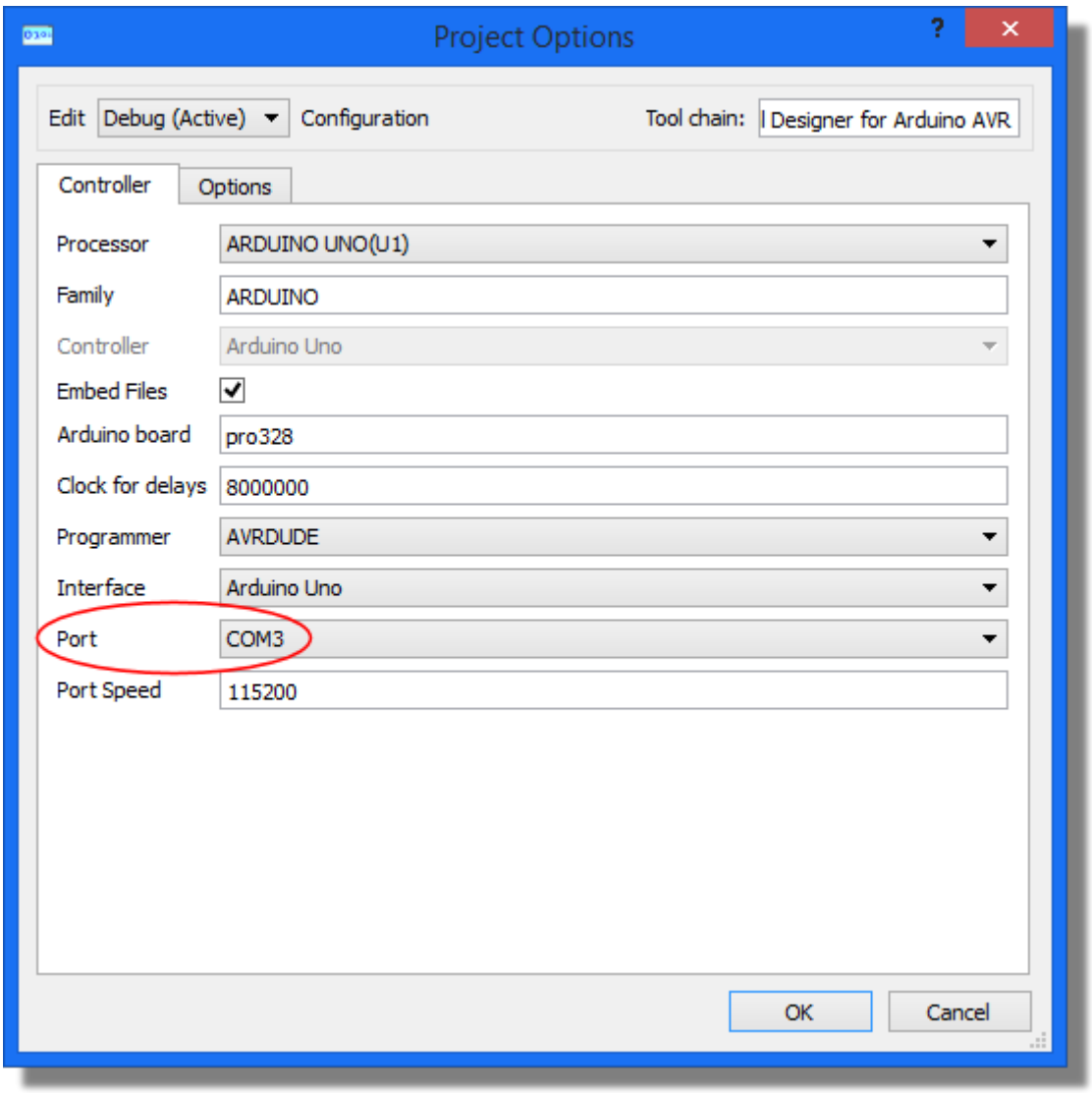
Attaching instrumentation is essential for analysis but will slow down simulation. It's worth deleting instruments when a debug session is complete.

Programming Hardware

When we are finished development we can easily program the physical hardware. First, connect the Adafruit TFT shield to the Arduino Uno and make sure a compatible SD card is inserted in the shield. Then, connect to the PC and then simply press the program button.



If the upload doesn't start ,use the settings dialogue to specify the COM port that you have connected to and try again.



You should see an upload complete message in the simulation log and your TFT should be displaying the picture.

This programming process is twofold. First, the resource is written to the SD card and then the firmware program is sent to the AVR microcontroller.

> They physical SD Card must be FAT16 or FAT32 and big enough to hold the resource file that will be programmed onto it.

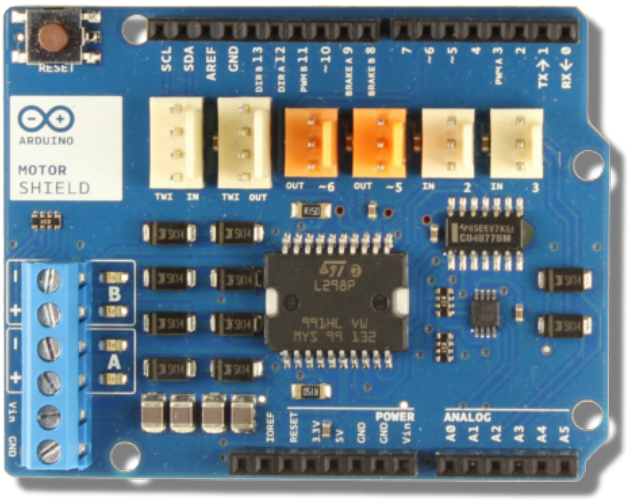
See Also:

The next tutorial will show how you can create your own electronics in the schematic (a virtual breadboard).

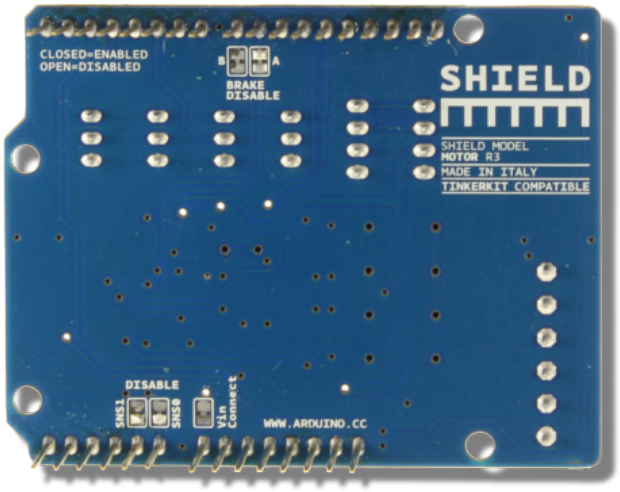
TUTORIAL 4 : MOTOR CONTROL

Introduction

This tutorial shows how you can easily control both DC motors and Stepper motors with Visual Designer. We'll do this using the Arduino Motor Shield V1 R3.



Front View



Rear View

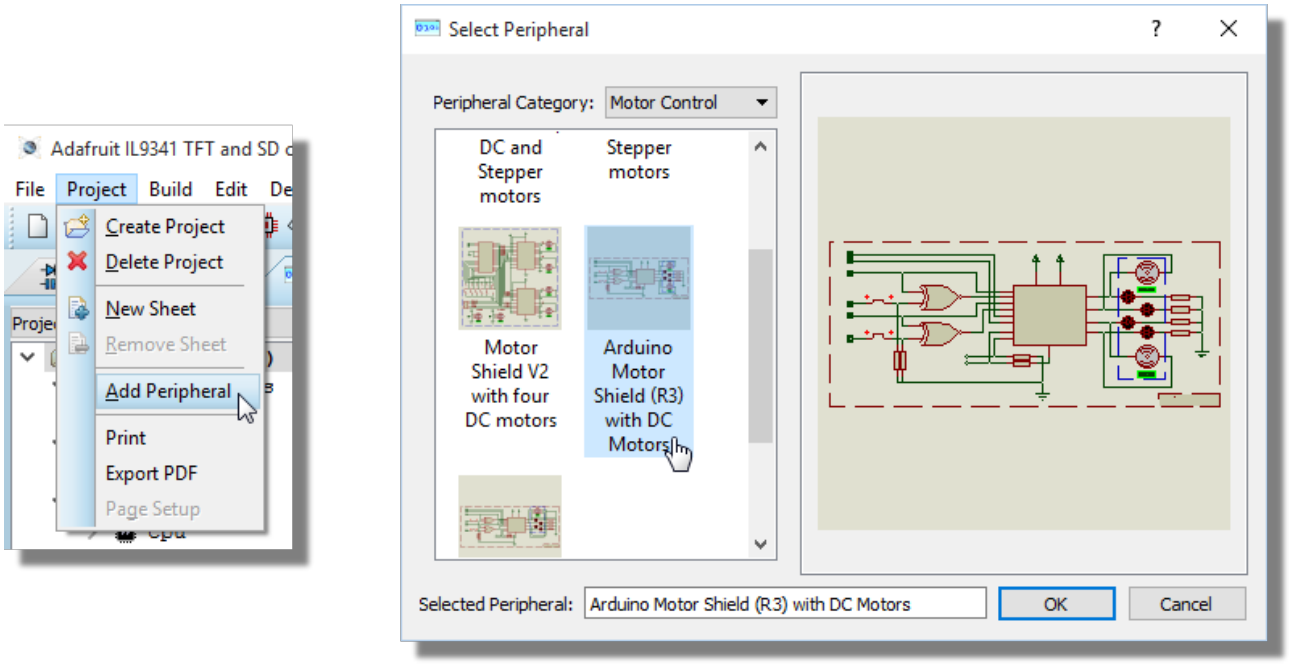
See Also:

https://www.arduino.cc/en/Main/ArduinoMotorShieldR3

> You can experiment with Servo motors using the Grove Servo peripheral and you can also use BLDC motors by drawing on the schematic.

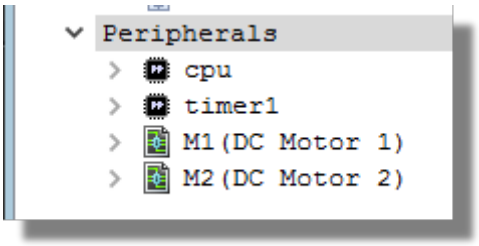
DC Motor Control

Start by creating a new Visual Designer Project and then Add the Arduino Shield with DC Motors in the normal way.

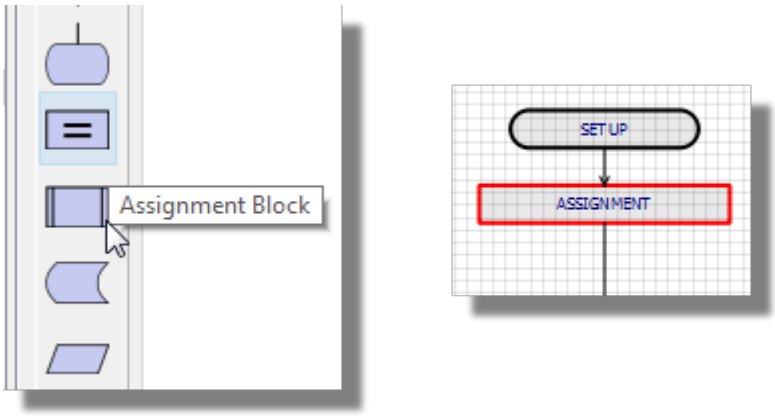


Note that under the Motor Control heading there are actually two shields. These are identical except that the first has the board configured and populated with two DC motors and the second is configured and populated with one stepper motor. In the real hardware of course you would need to configure and populate the shield manually according to which project you were programming the Arduino with.

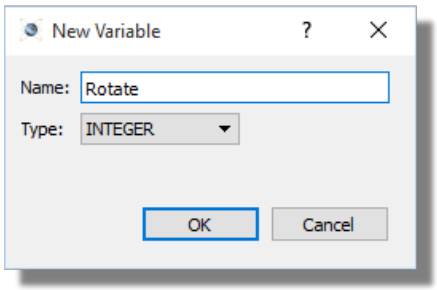
You should notice that in the Project Tree you now have two motors with associated methods.

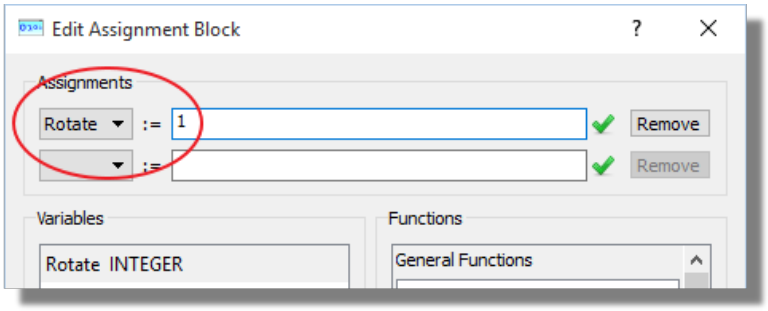


Let's write a small program to drive the motors in opposite directions. The first thing we want to do is initialise a variable for the speed of the motors so drag and drop an assignment block into the Setup routine (we only need to do this once).

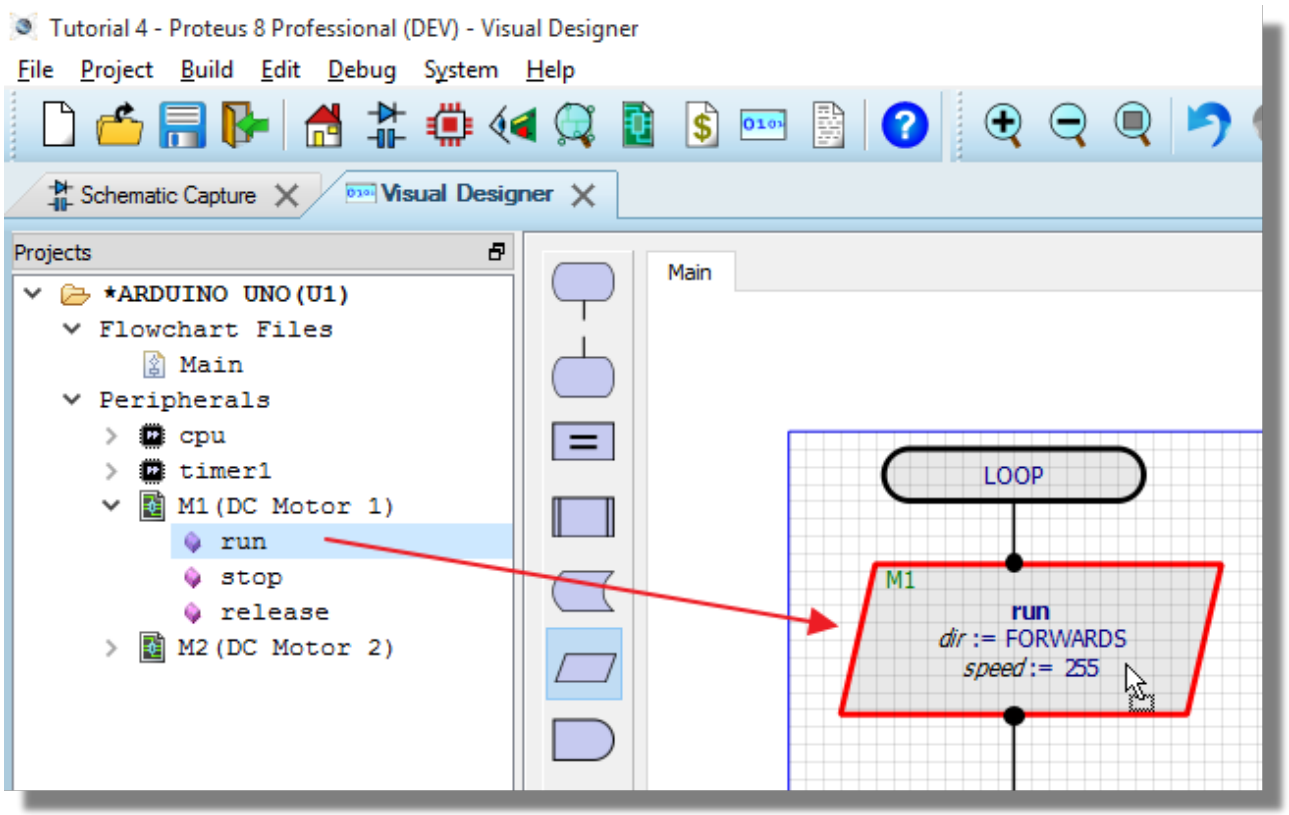


Edit the block, create a new variable and assign a value.

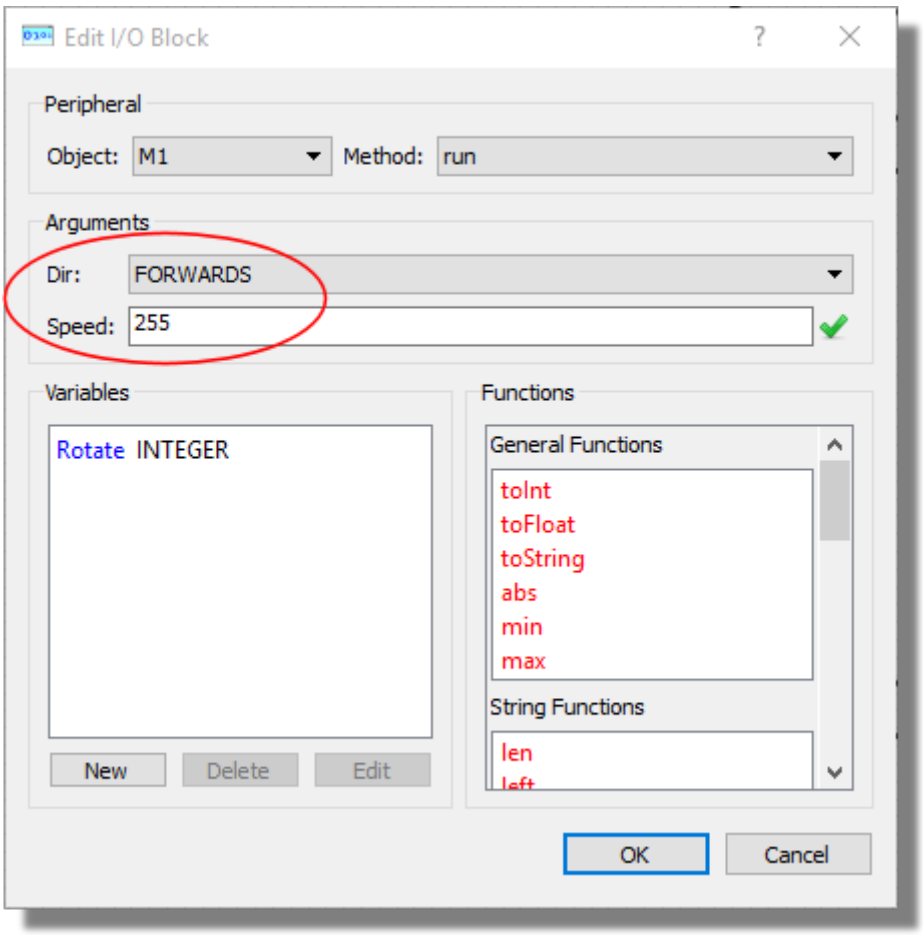




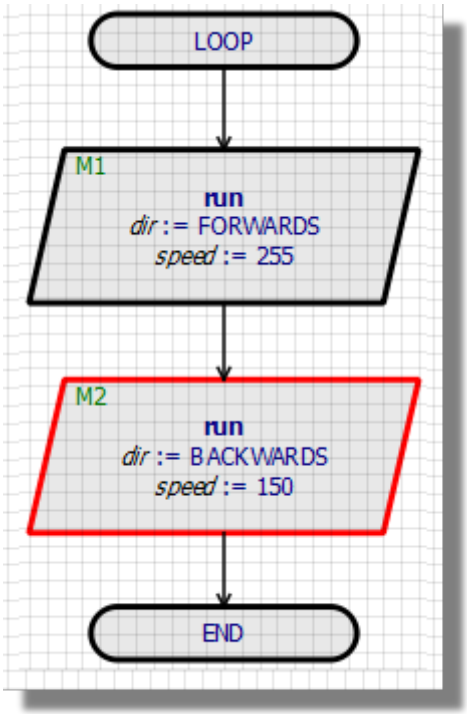
Now, let's drive the first motor forwards. Drag and drop the run methods into the loop routine.



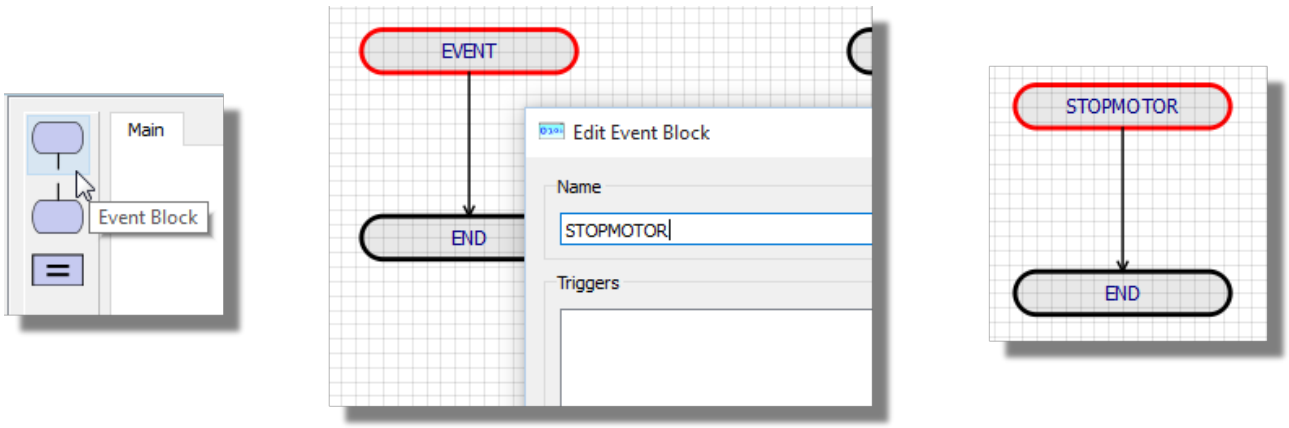
Edit the block, set the direction forwards and assign the speed to our variable.



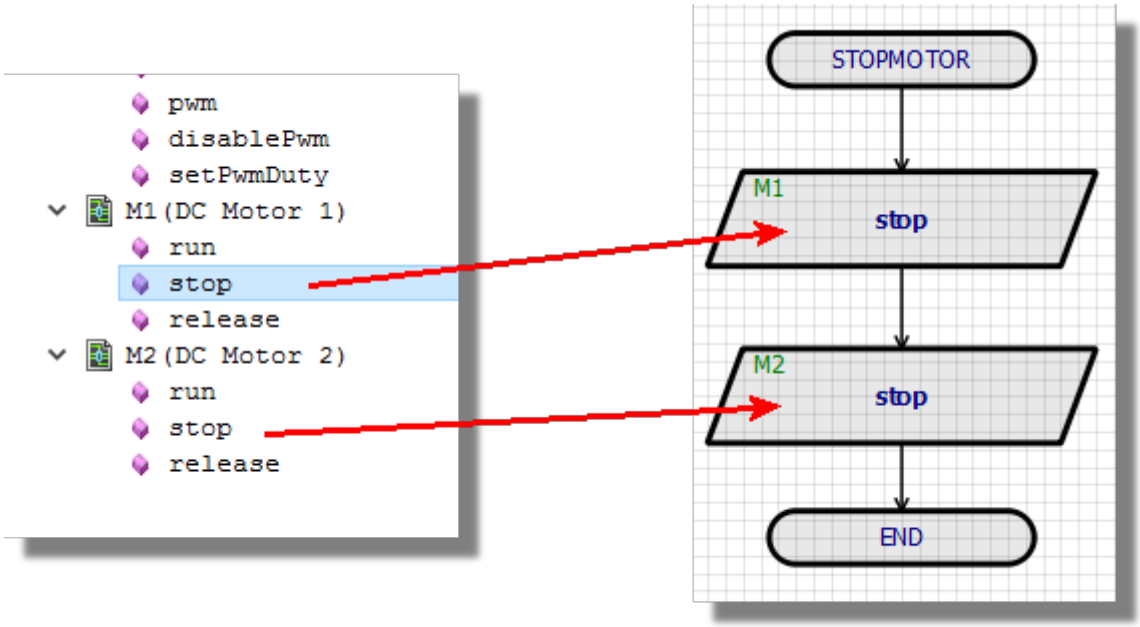
Repeat the process for the second motor to send it spinning backwards.



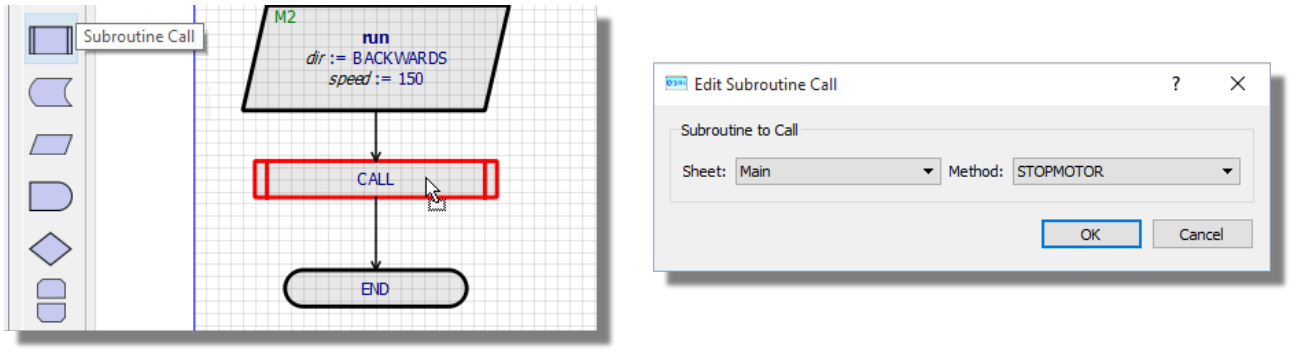
Next, we'll add a sub-routine to stop the motors. Drag and drop an Event block into an empty place, edit the Event block and give it a sensible name.



Now, drag and drop the stop routine for both motors into the sub-routine we've just added.



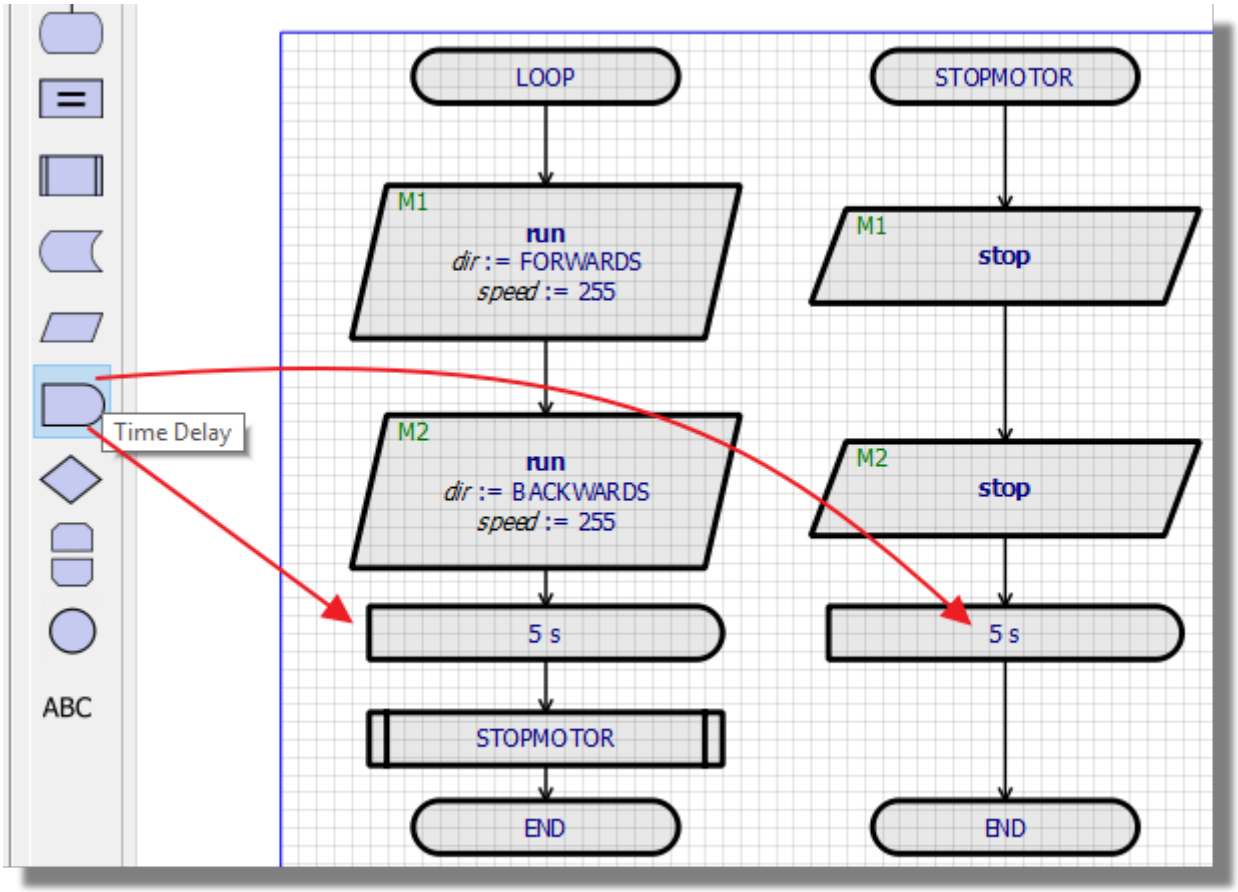
Finally, drag and drop a subroutine call block into the main loop and edit it to call our routine.



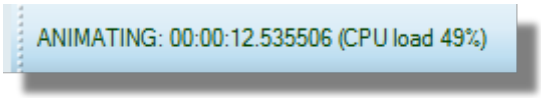
To see this in action press the pause button and then single step the code using the Step-Into icon at the top of the source window. You'll notice that the code executes through the sub-routine and then returns to the main loop.



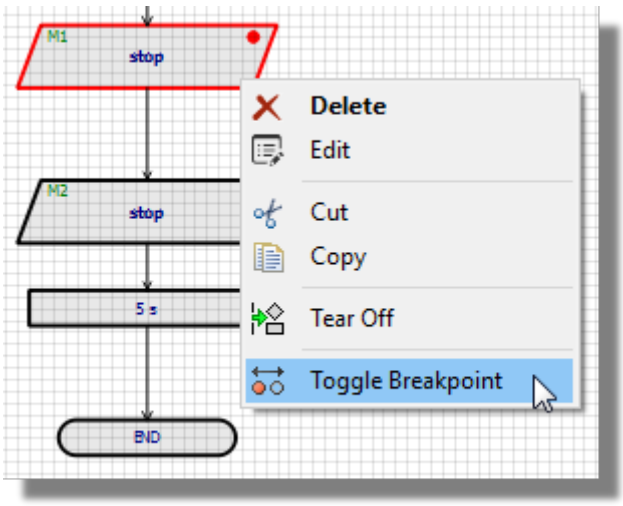
We have most of the pieces of our program in place but we are starting and stopping the motors very quickly which isn't going to help them build momentum. Much better to add a delay in the drive phase and a similar time delay in the stop phase. Again, this is drag and drop from the flowblock menu onto the chart routines.



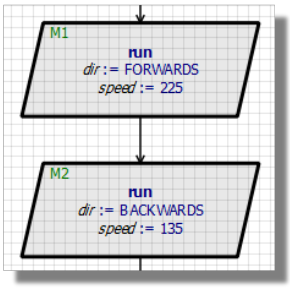
If you run the simulation now you can monitor the drive and spin down loops via the time display on the status bar

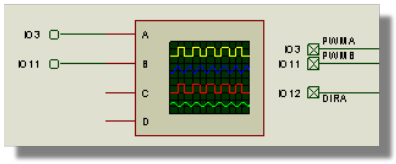


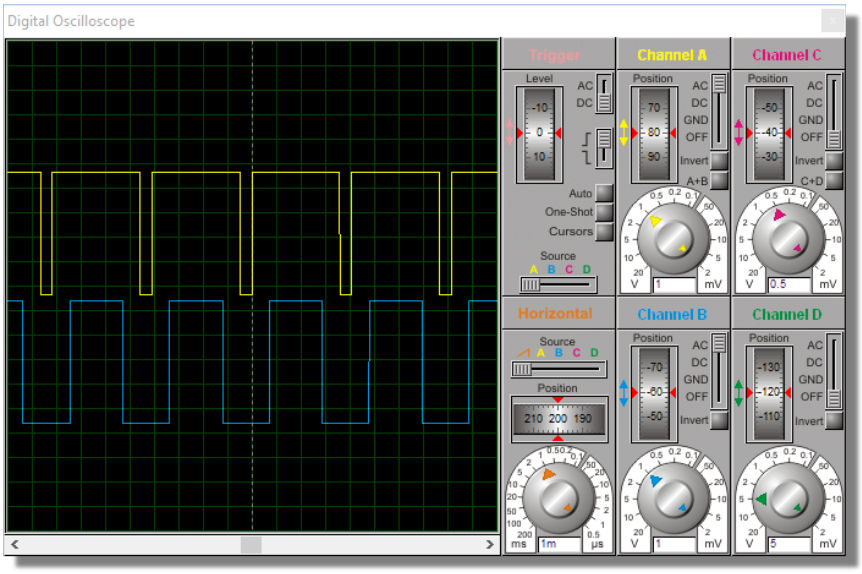
You could also pause the simulation, set a breakpoint and step the code. Unlike the real hardware the motors won't lose momentum when you are at a breakpoint - a real advantage to debugging in software.



For more advanced analysis you could place and wire an oscilloscope on the schematic and watch the PWM waveforms drive the motor.



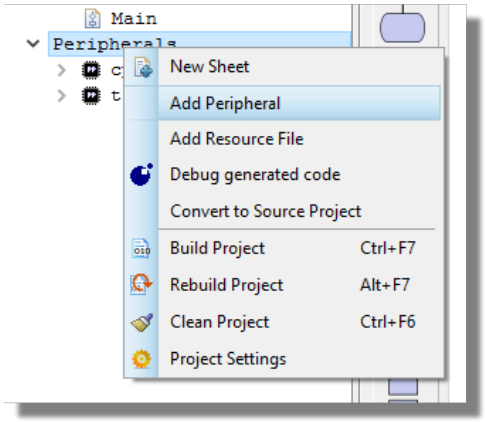


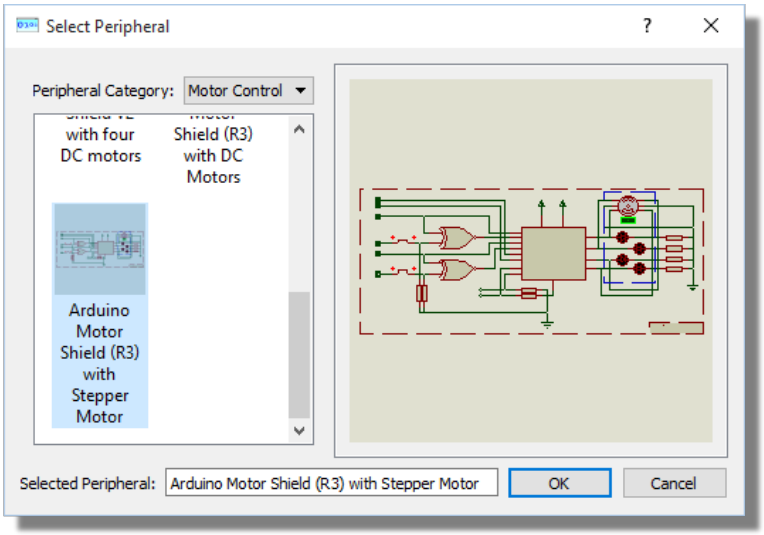


As always, you can program the physical board at any time via the upload button.

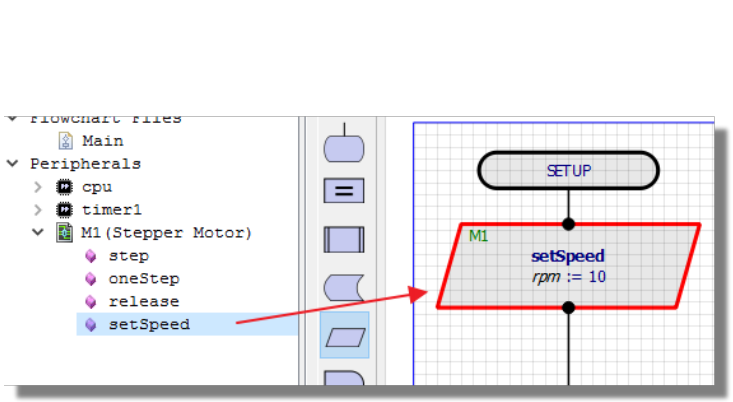
Stepper Motor Control

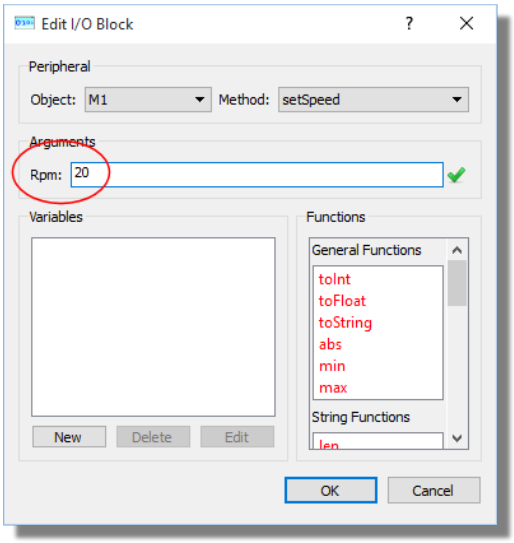
For the stepper motor we are using the same hardware shield (Arduino Motor Shield V1 R3) but configured for a single unipolar stepper motor. Start a new Visual Designer project and bring in the virtual shield via the Add Peripheral command.



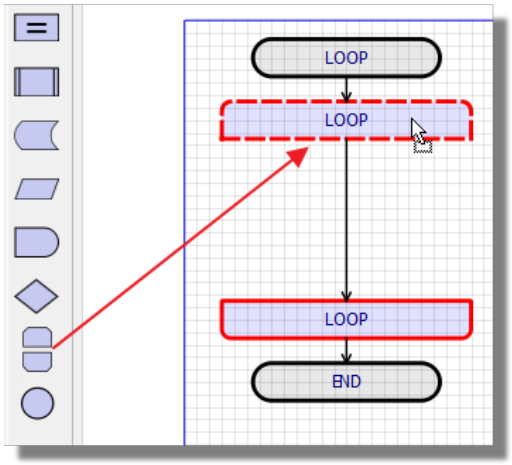


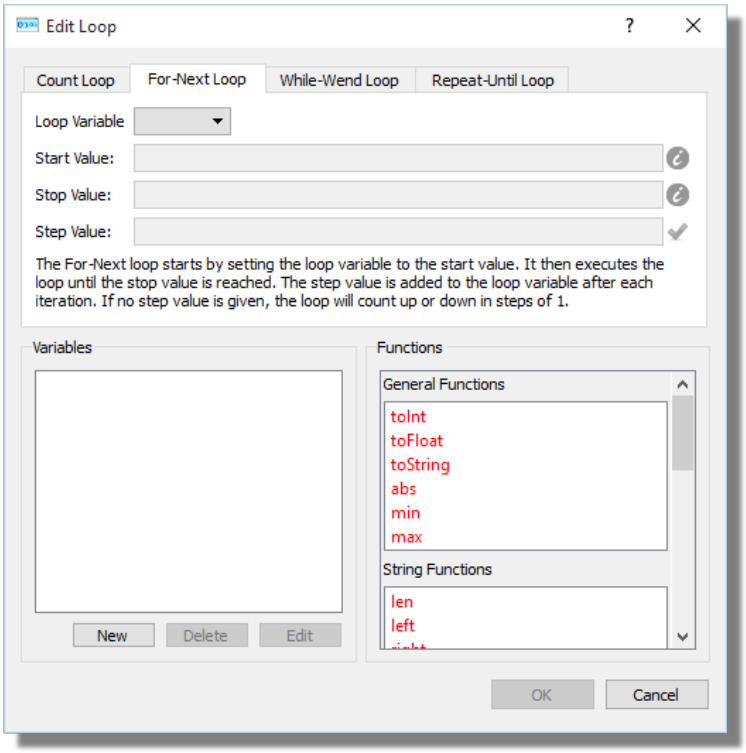
We'll design a small program to step the motor in one direction and then reverse direction and step backwards. Our first job is to define the speed which we can do by dragging the setSpeed() method into the Setup routine.



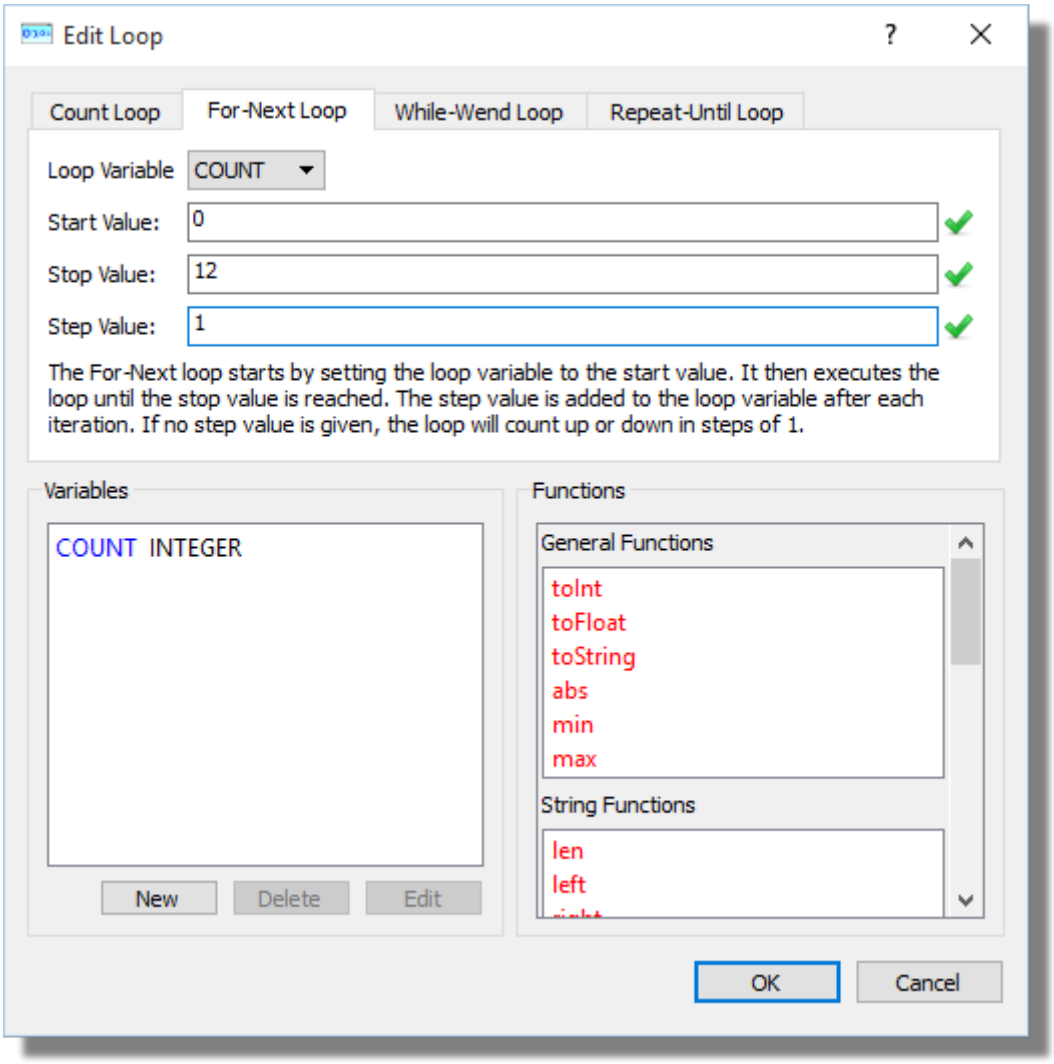


In order to step a set number of times we want to use a loop construct. Drag and drop one onto the Main (Loop) routine and edit the block.

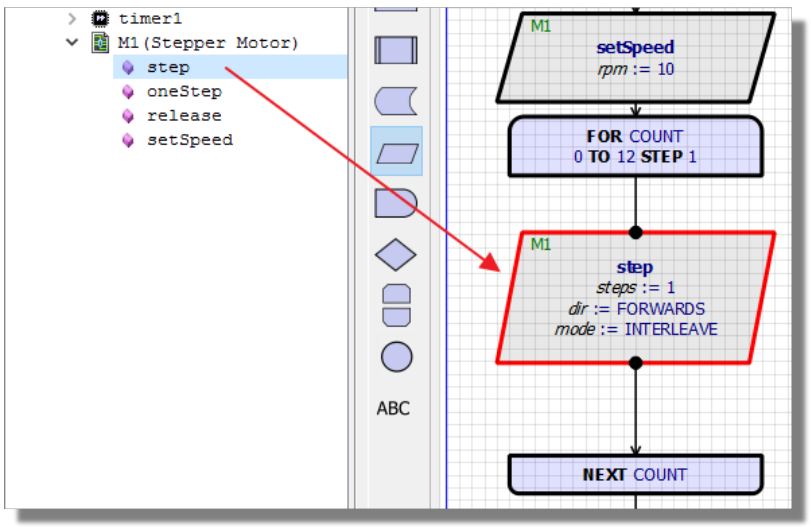


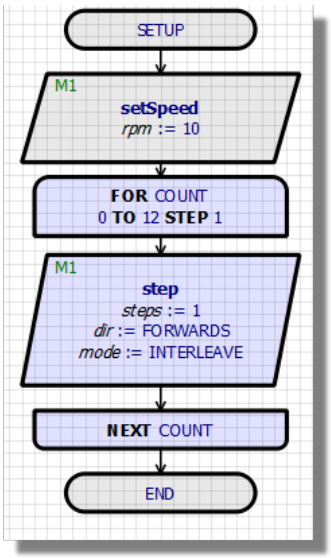


We'll use a For-Next loop here so add a variable for count, initialise to zero and let's go round the loop 12 times.

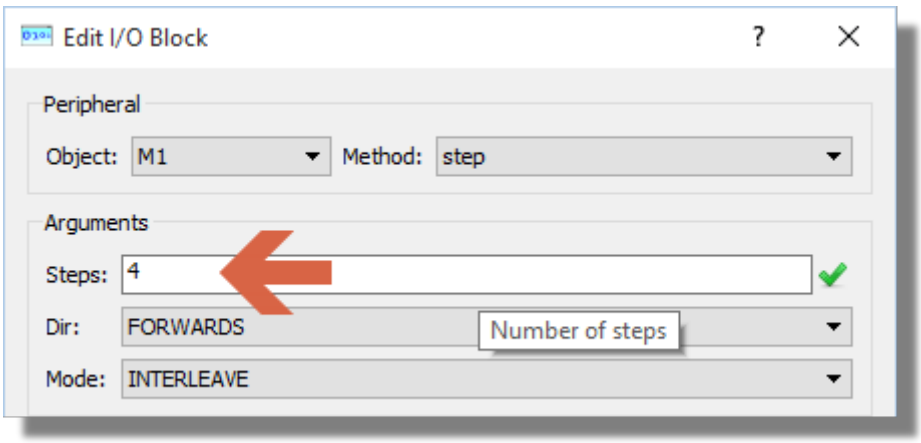


Now, drag the step() method of the motor inside our loop - you'll notice the colour change to indicate it's part of the loop construct.



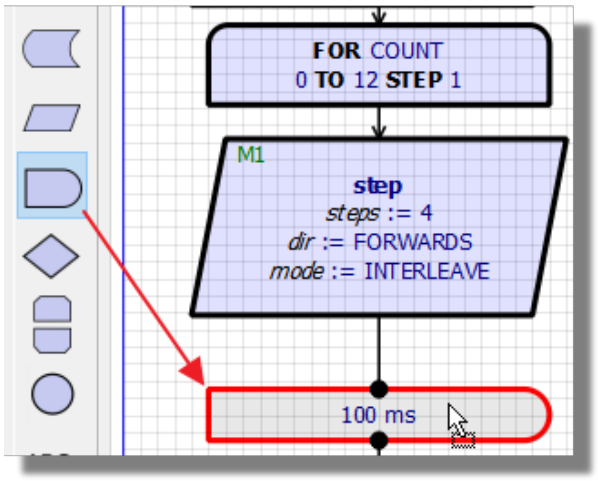


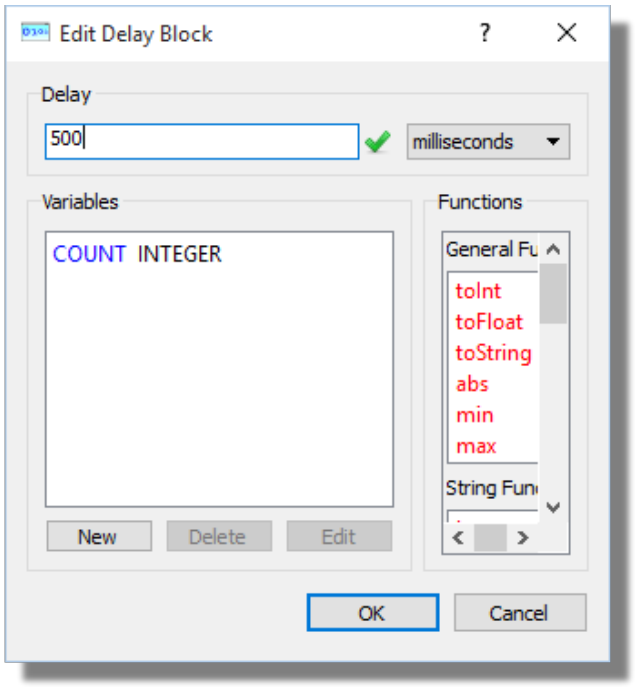
Edit the flowblock and make the number of steps 4 so that we get sensible movement, set the direction to forwards and the mode to interleave.



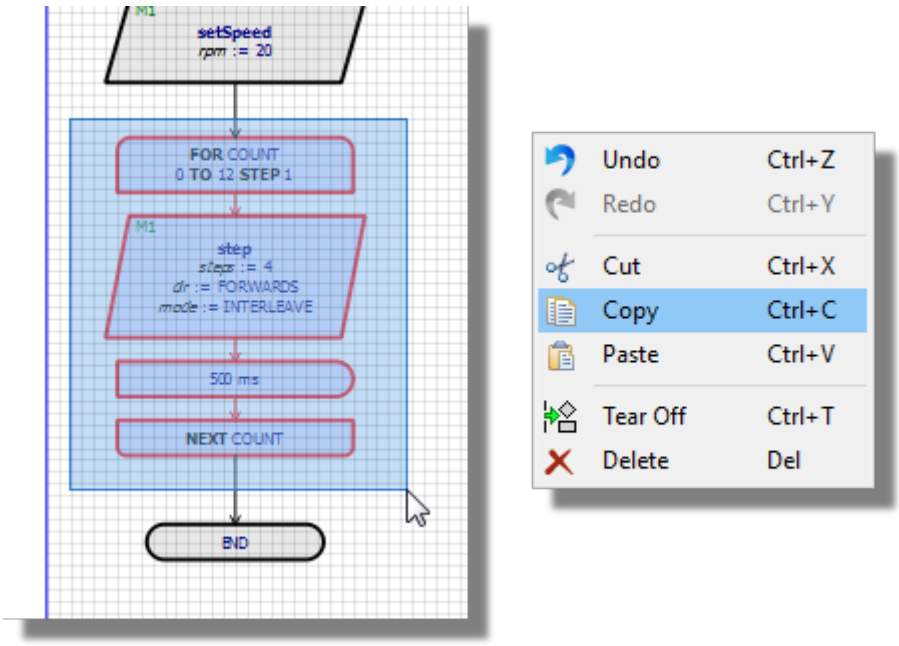
Single Mode means single-coil activation, Double mode means 2 coils are activated at once (for higher torque) and "interleave" mode means that it alternates between single and double to get twice the resolution (but of course its half the speed). Lots more info on this in various internet resources.

Add a modest delay after the step routine to allow the drive phase to complete before we iterate the loop.

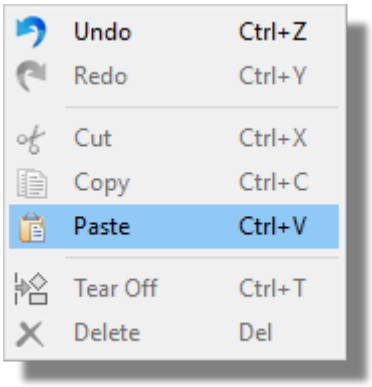




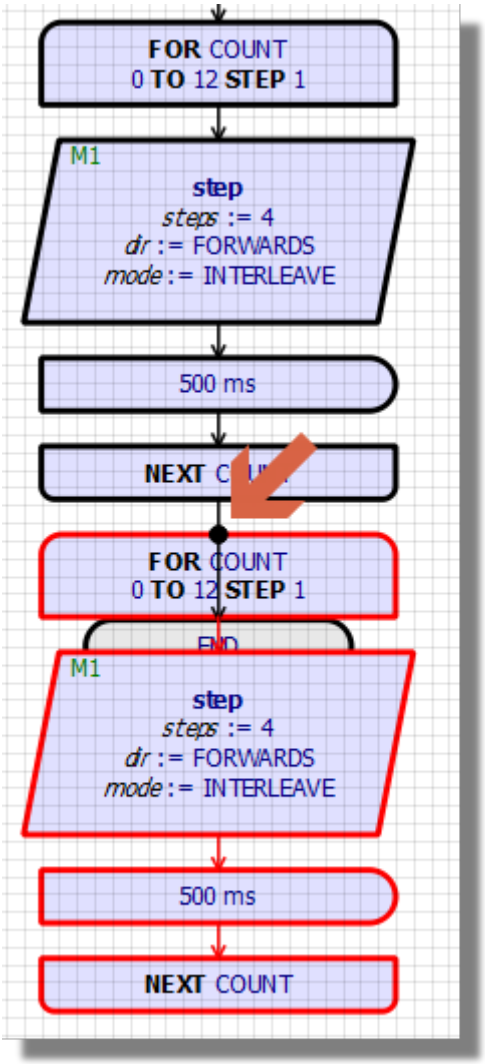
That's the forwards motion in place. Driving the motor in the opposite direction is identical except that we change the direction in the step routine. You can place a second loop and drag methods in or we can just use the trusty copy and paste. Drag a box with the right mouse around the loop and select copy from the resulting context menu.



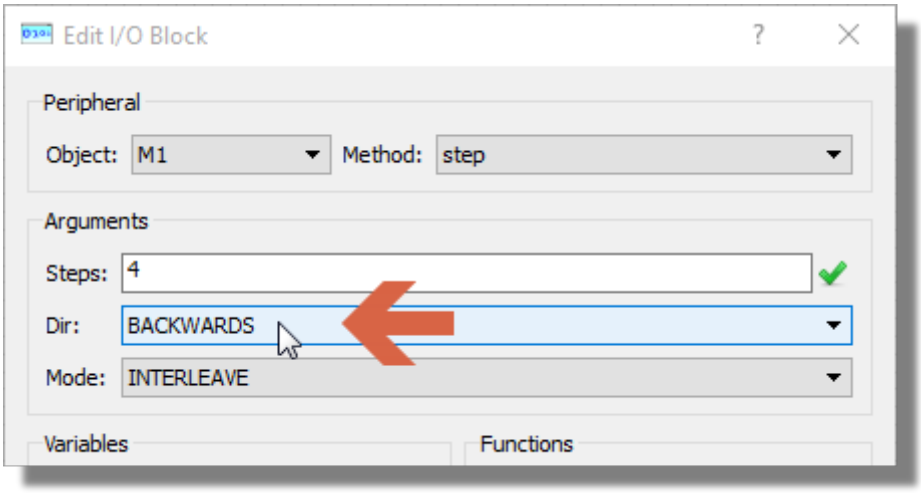
Next, right click (or CTRL+V) to paste a second set of blocks.



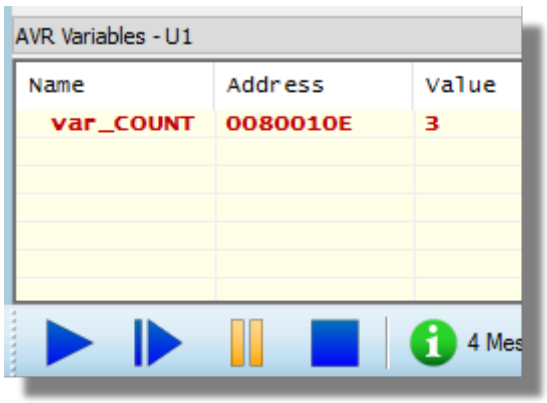
Drag them into place underneath the first loop, releasing the mouse when the top node appears with a dot.



Finally, edit the step method and change the direction.



That's it. Press play and watch the motor step forwards and backwards or pause, set a breakpoint and single-step your program. Note that the counter variable is displayed in the variables window which will tell you instantly how many more iterations of your loop remain.


AVR Variables - U1
Name
Address
Value
var_COUNT
0080010E
3
4 Mes

TUTORIAL 5: SCHEMATIC BREADBOARD

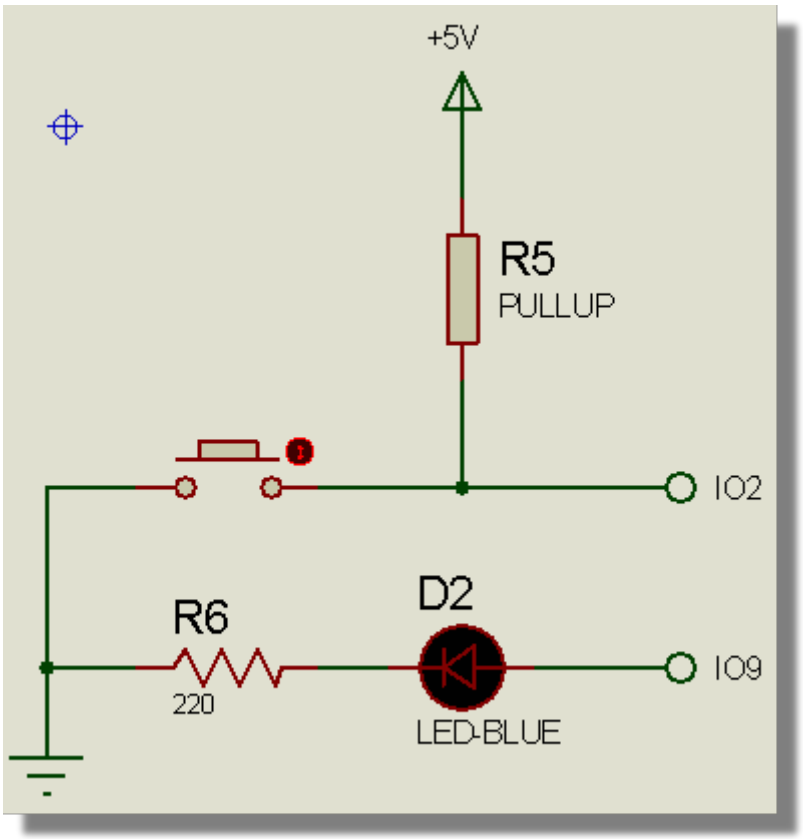
Introduction

So far in all of the tutorials we have used pre-existing Arduino shields or Grove modules for the hardware. There is however nothing to stop us designing our own hardware directly on the schematic . We will cover the basics in this tutorial with a button and an LED. Note however that there are literally thousands of parts that can be simulated in the Proteus libraries. Start by creating a new project with a schematic, no PCB and with Firmware for Arduino Uno using the Visual Designer Compiler for Arduino.

This tutorial is also covered by a short movie on the Labcenter Youtube channel.

Drawing on the Schematic

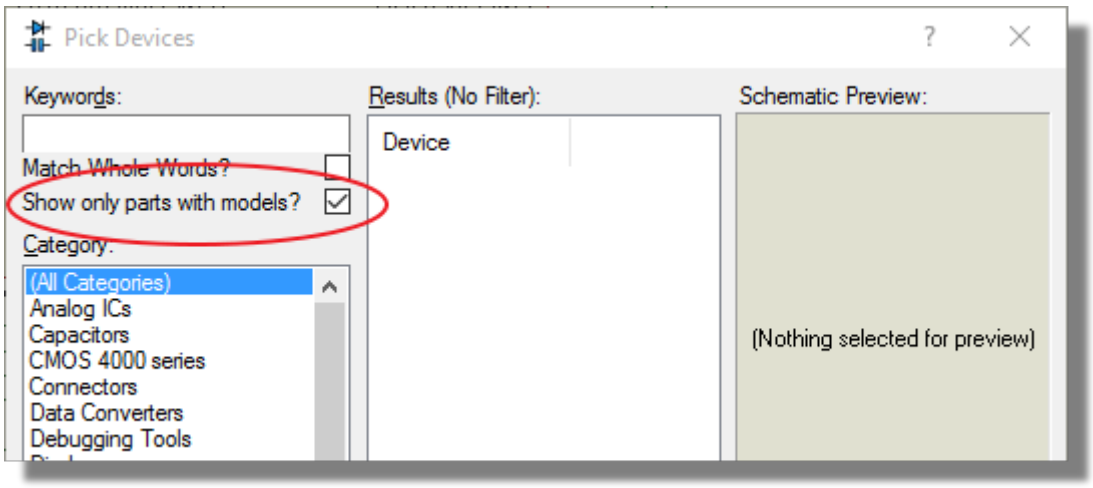
There is a good deal of documentation on schematic drawing in the main Proteus help files so we'll cover only in brief here. Our task is to draw the following circuitry on the schematic;



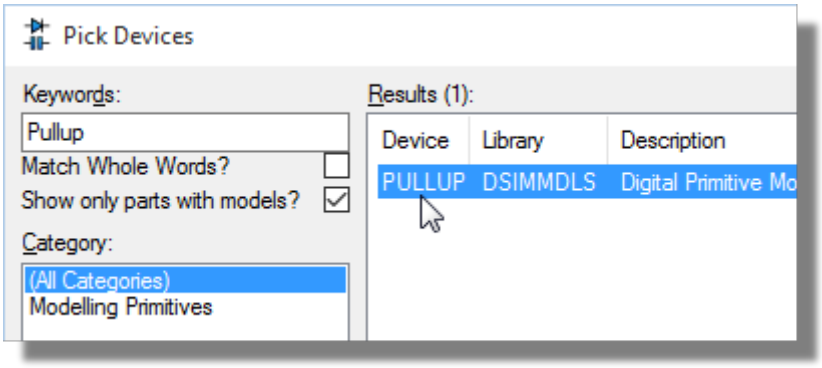
The three main steps are pick, place and wire.

Picking

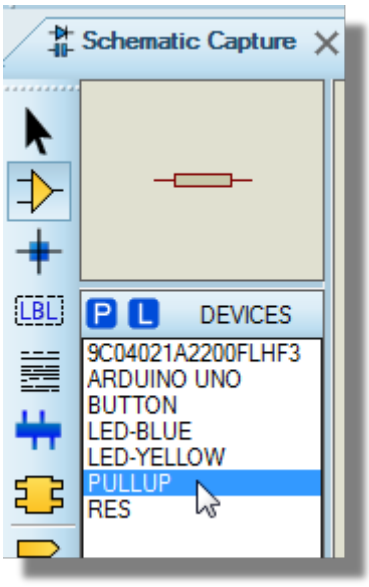
To pick from the library select the 'P' button above the parts bin on the schematic. This presents us with the library browser. We are only interested in those parts that we can simulate so start by checking the box to filter the results.



We will need an LED, a 220 Ohm resistor, a pullup resistor and a button, all of which we can find by typing keywords at the top left of the dialogue form.



To pick a part into the parts bin, simply double click on the result.



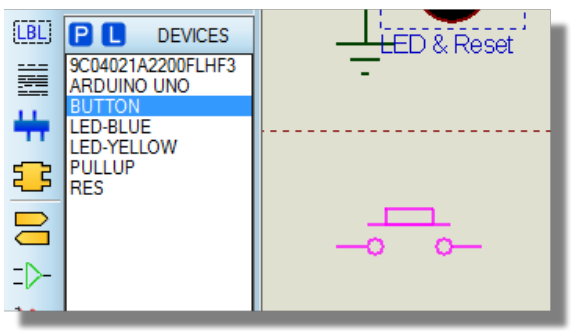
You may be asked if you want to replace the existing part. This will happen when you pick a part already in the parts bin; for example, there is already an LED in the parts bin because it's used in the Arduino Uno Shield.

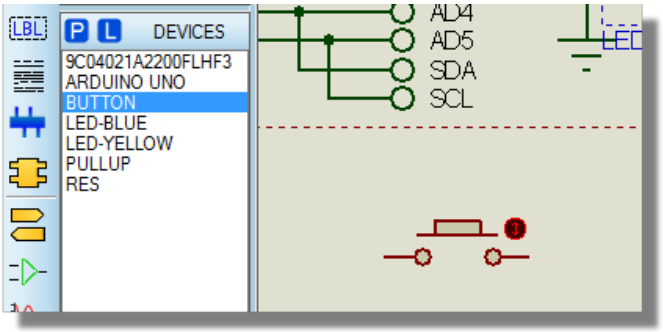
See Also:

- Schematic Tutorial : Basic Schematic Entry

Placing

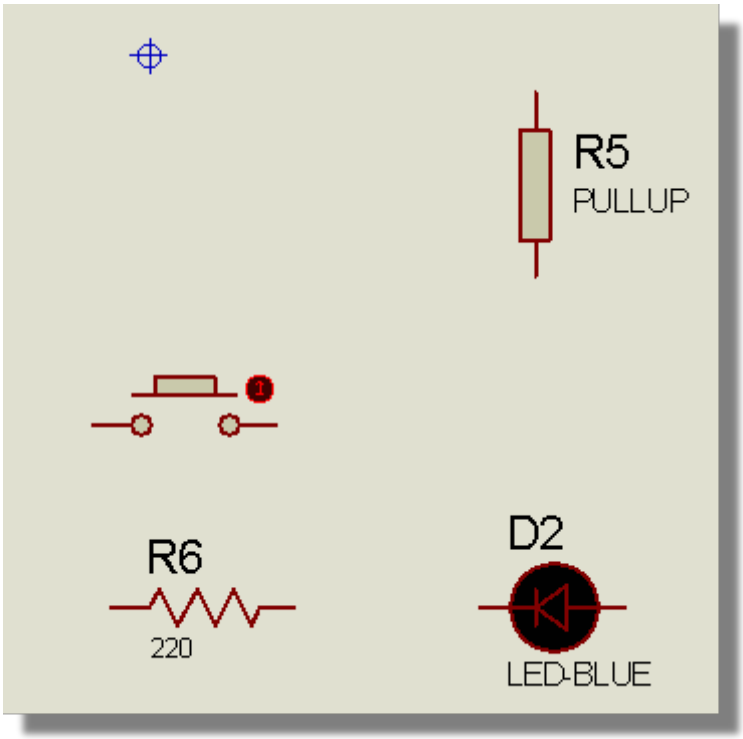
To place a part, select it in the parts bin, left click once on the schematic, then drag into place and left click again to drop.



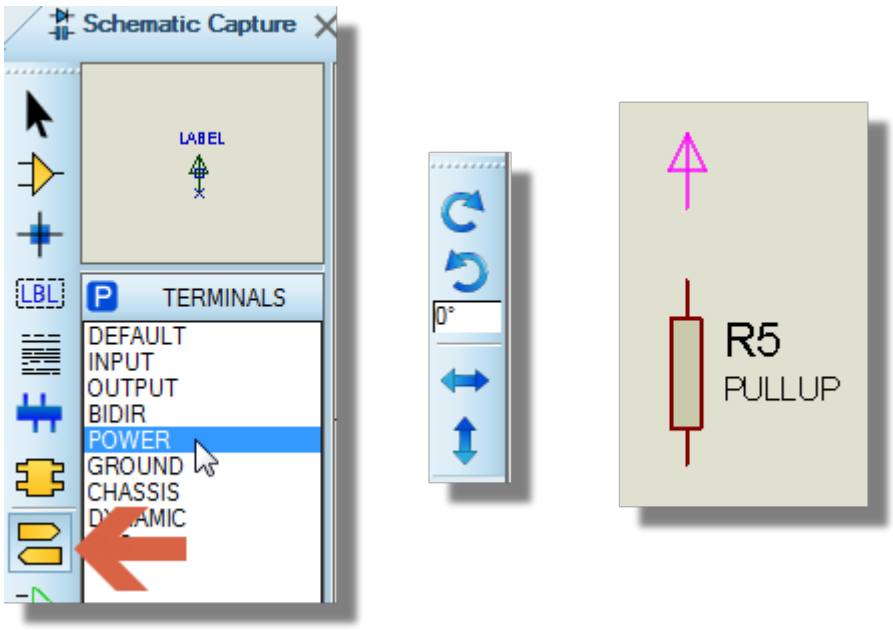


If you need to rotate, you can do this with the plus and minus keys on the keyboard during placement.

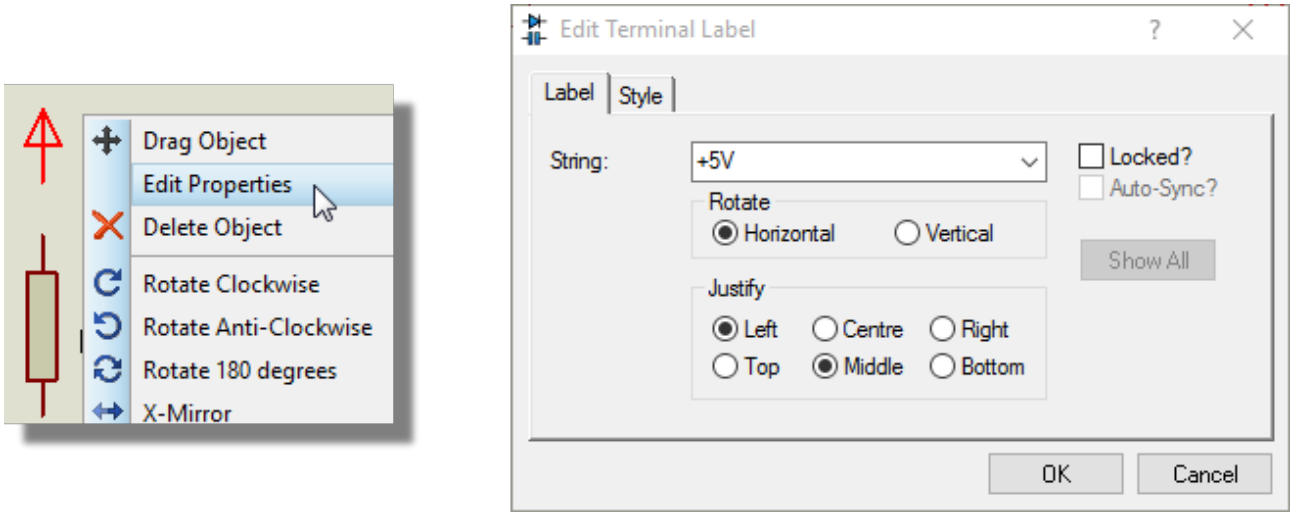
Place all of the parts approximately in position - something like the following:



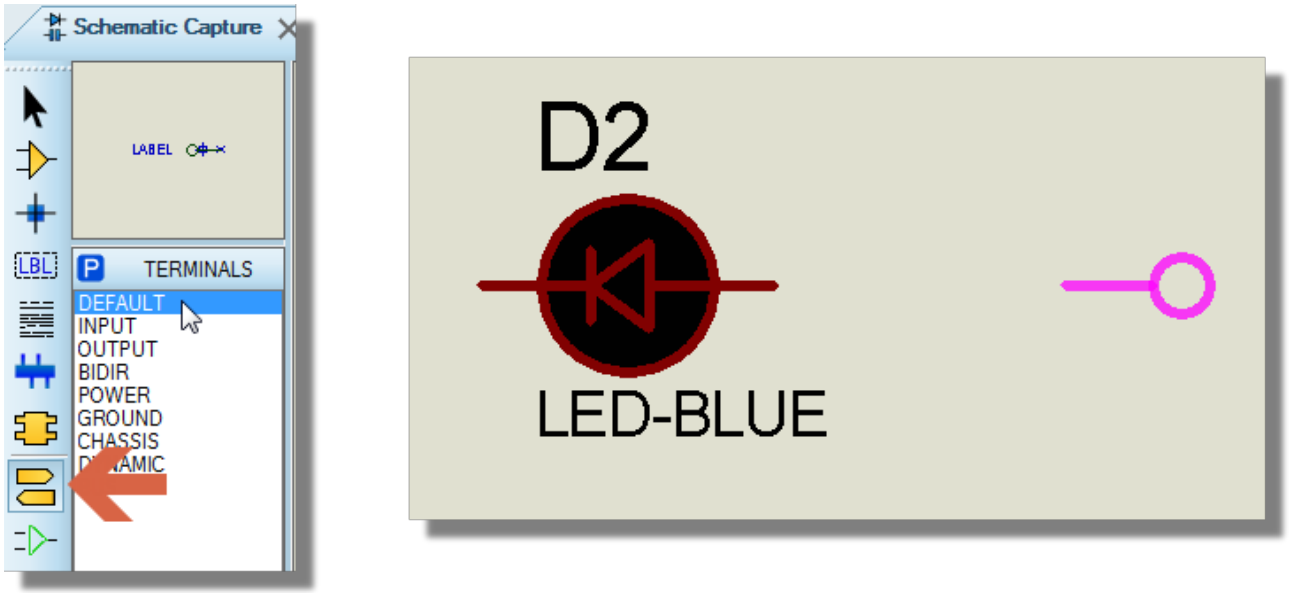
We can apply power and ground by selecting terminal mode and placing /rotating in the same way as we did the parts.



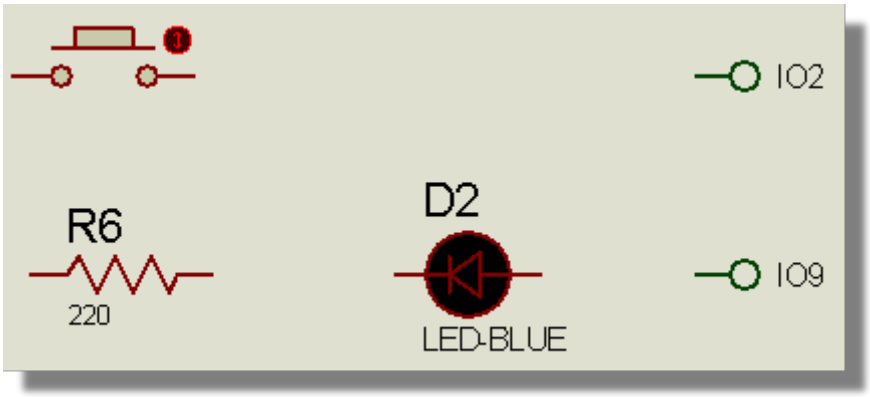
Edit the power terminal and specify the correct voltage.



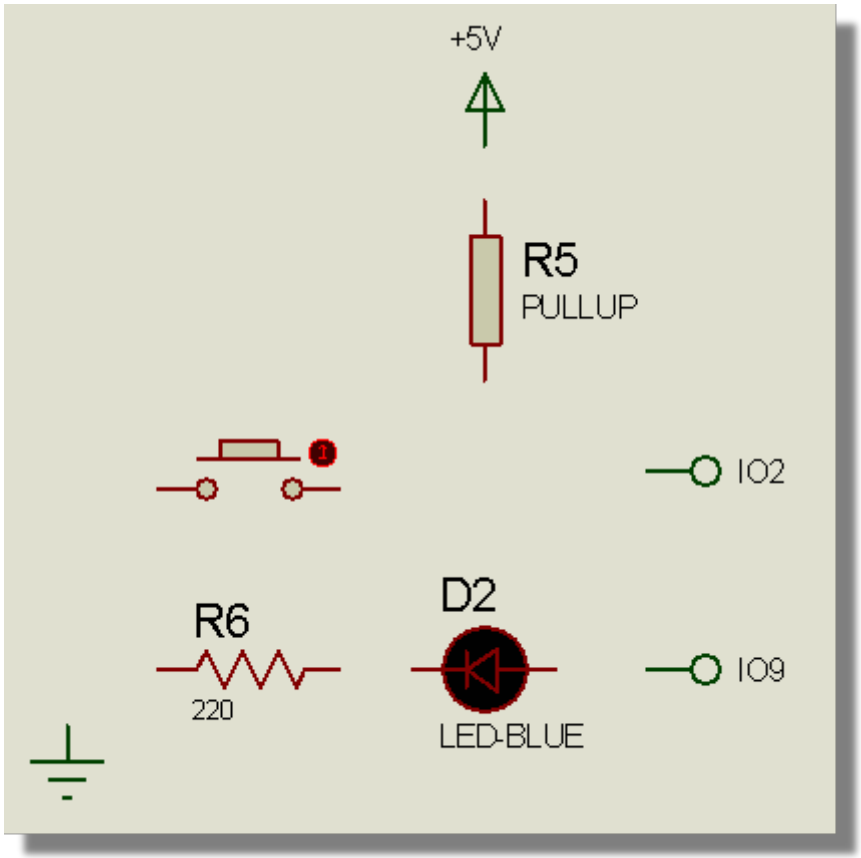
We'll also need a terminal to connect our button and our LED up to the correct pin on the Arduino. We can use default terminals for this.



It is here that the hardware and the firmware designs start to merge. The best way to wait for a button press in firmware is via a pin change interrupt. This means however that we need to connect the button to a pin on the Arduino for which a pin change interrupt is available. On the Uno we can use IO2 for this and can connect the LED to any available IO pin.

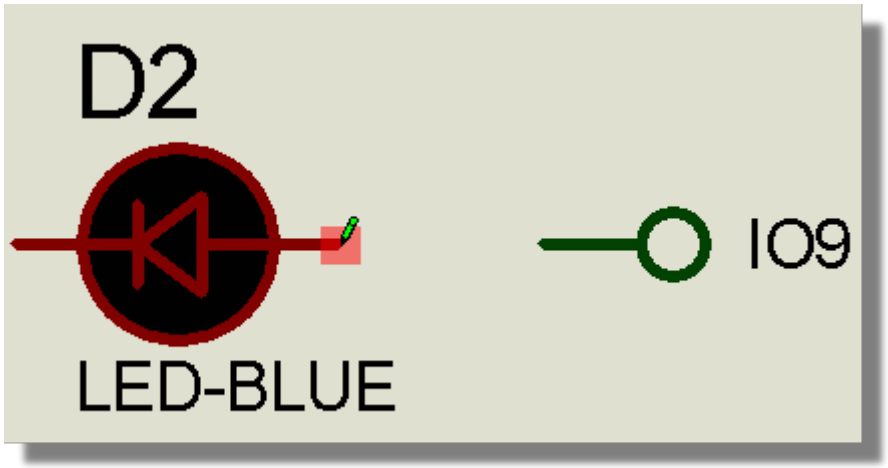


You want it to look like the one below (without the wires at this stage).

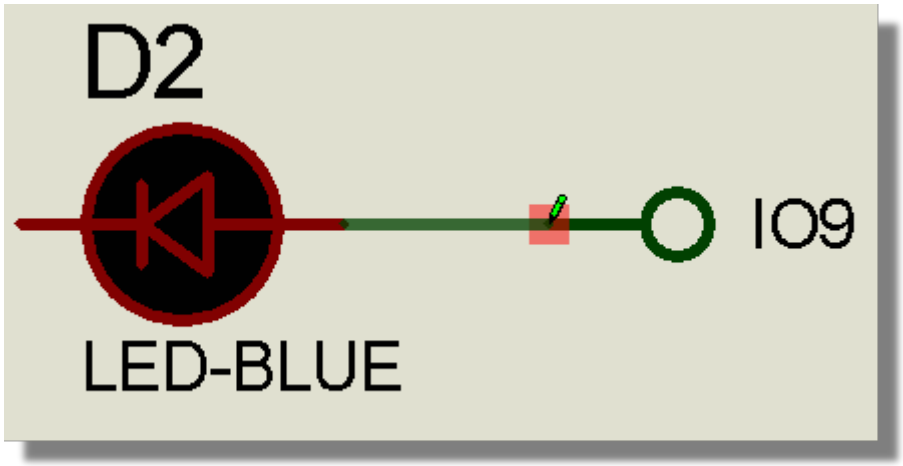


Wiring

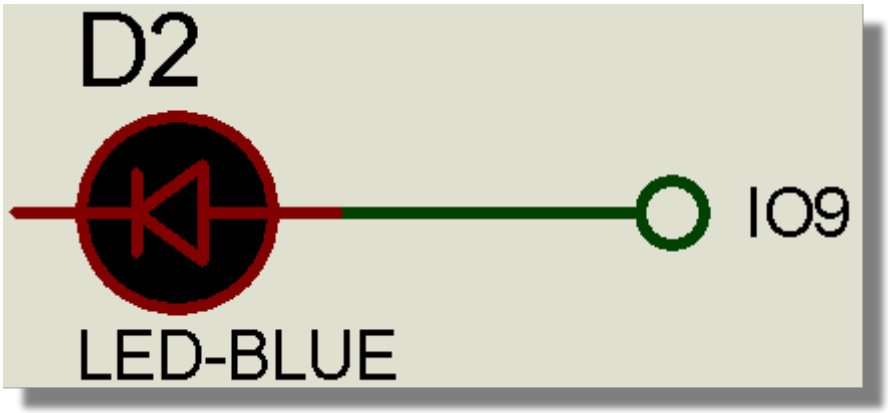
Wiring on the schematic is point to point. First, hover the mouse over a pin until the cursor turns green and then left click



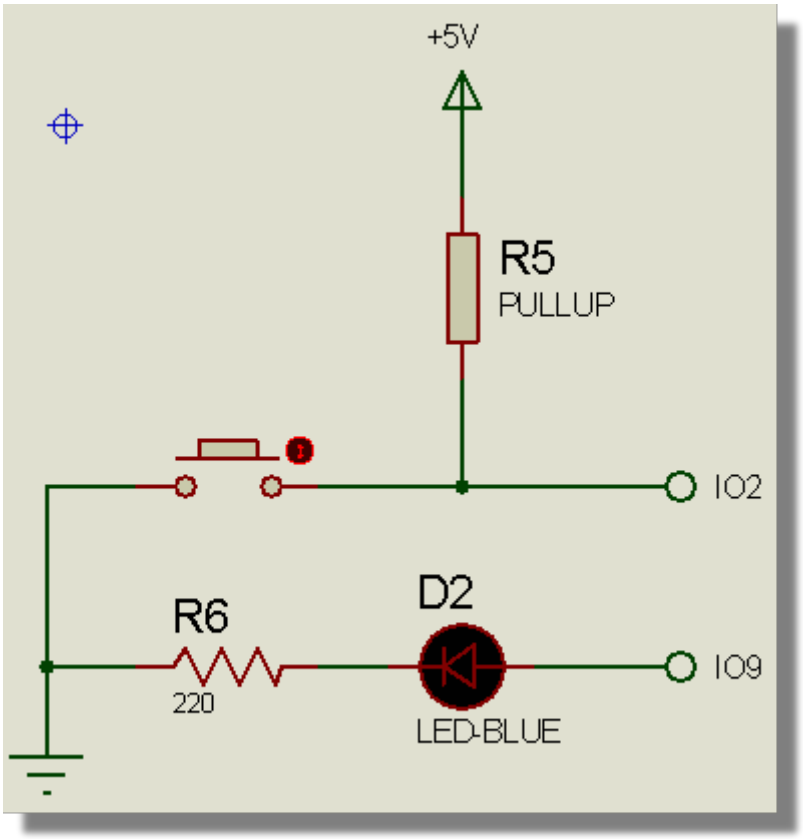
Next, move the mouse to the destination pin/wire until the cursor turns green again.



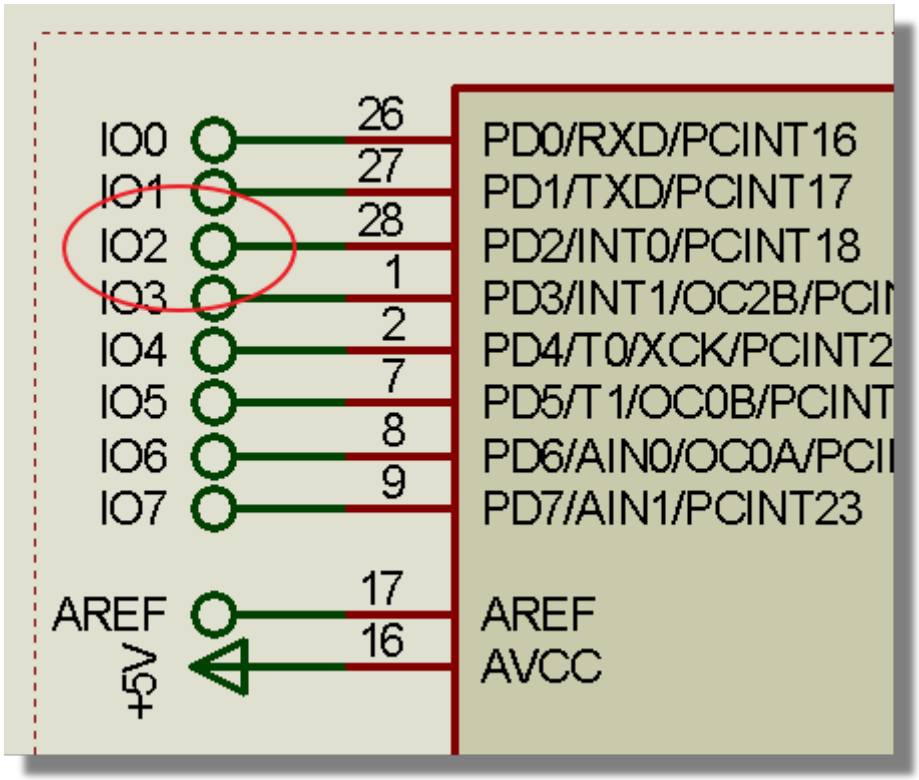
Left click the mouse again to complete the wire.



After wiring is complete your schematic should look like the following.



Remember the two terminals with the same name on the schematic have an 'invisible wire' between them so by wiring the button to a terminal called IO2 we have actually connected it to the IO2 pin on the Arduino chip.

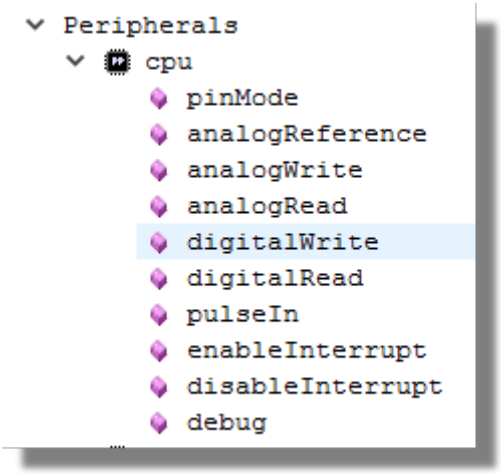


See Also:

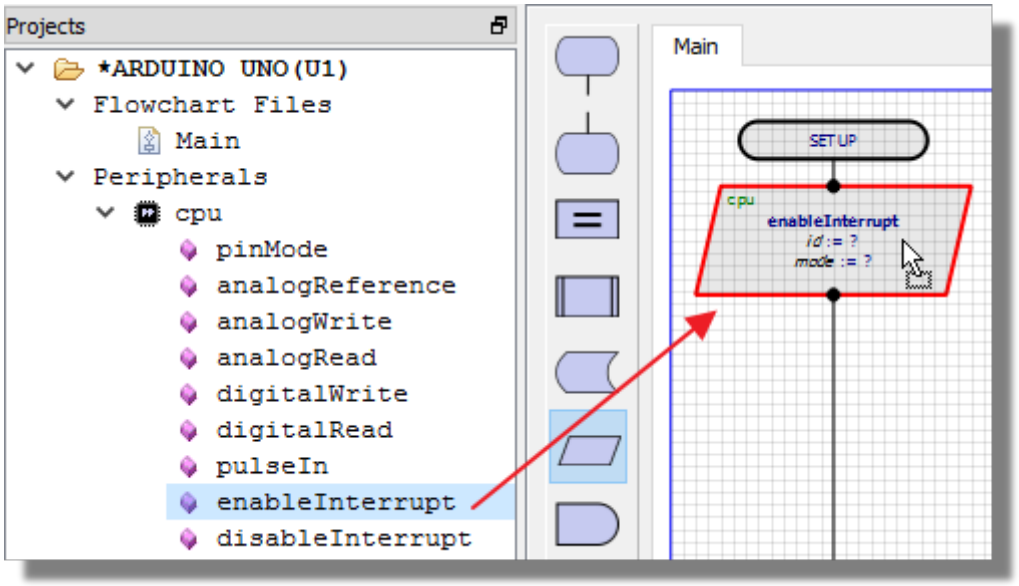
Schematic Reference Manual : Wiring Up

Designing our Program

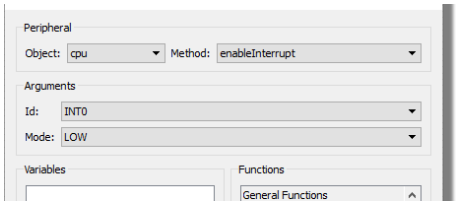
Everything we have done so far is hardware design. The program design itself takes place on the Visual Designer tab. Unlike our other tutorials we don't have any external peripheral methods in the project tree so we will have to drive the electronics directly using the CPU methods.

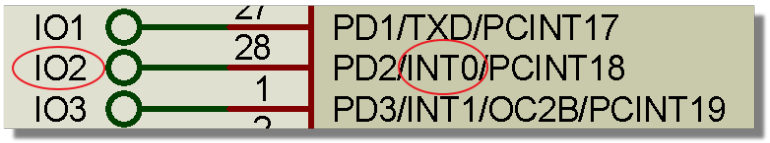


We'll start by dragging and dropping the pin change interrupt routine in to the Setup routine.

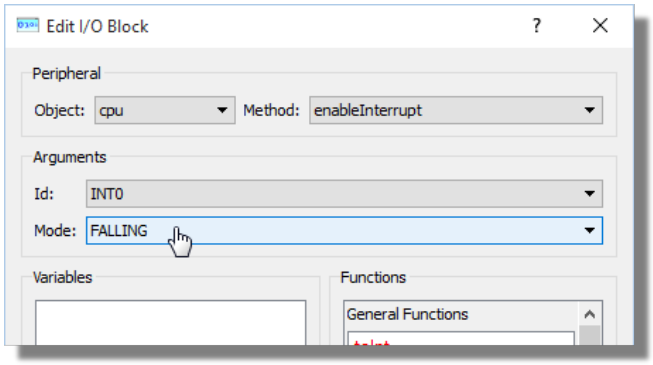


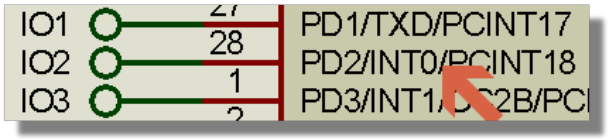
Now, there are two available interrupt pins on the Uno but we need to select INT0 because we wired the button onto IO2.



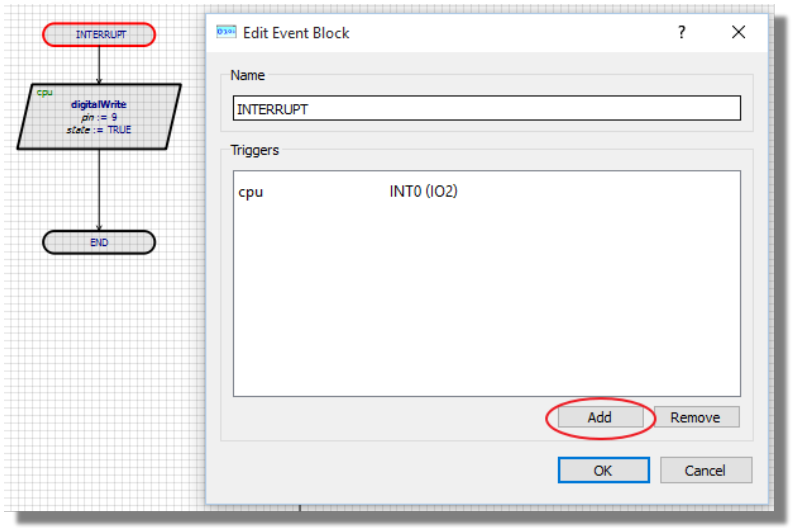


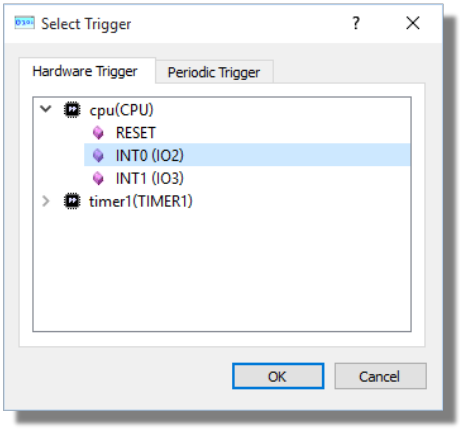
Similarly, we need to set the interrupt to trip on a falling edge because we wired the button to pull low when pressed.



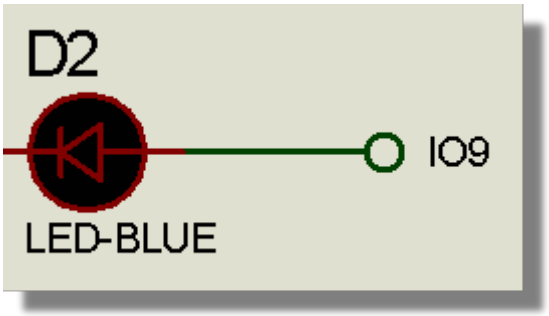


Now we need to handle the interrupt event. Drag and drop an event block onto the Flowchart Editor and edit. Give a sensible name and then specify the trigger as INT0.

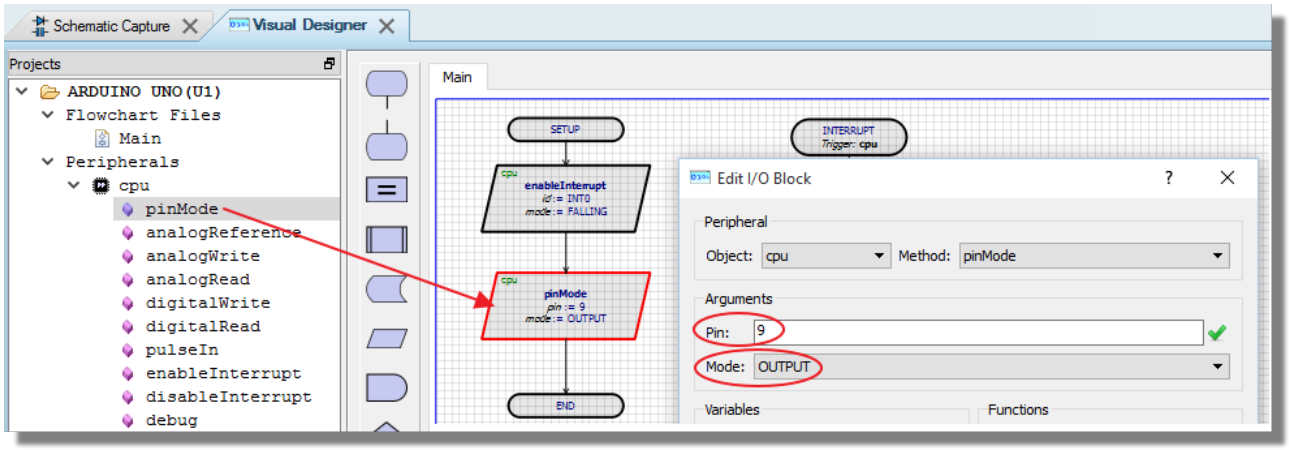




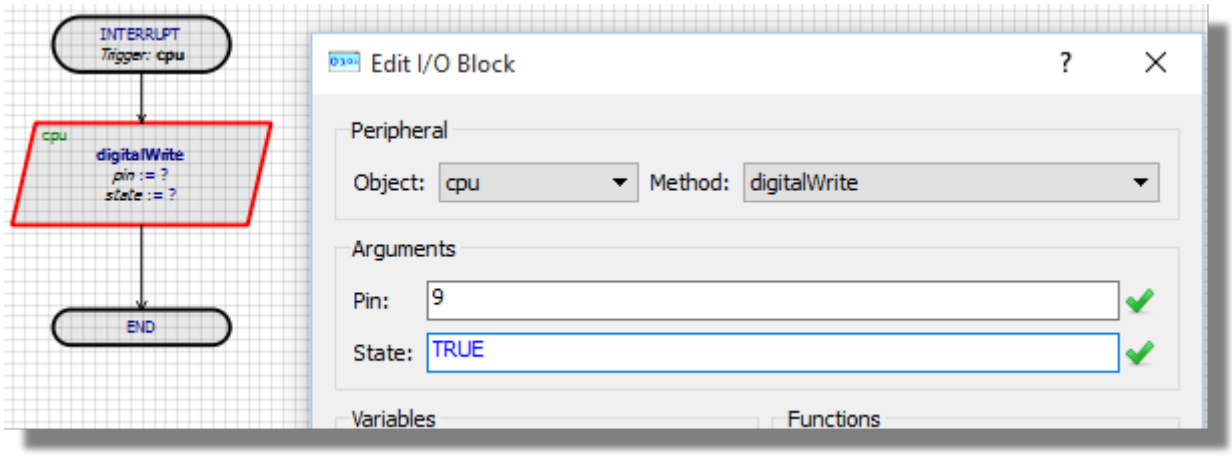
We are going to turn on an LED when the button was pressed so we need to write out a logic high on one of the IO lines. Specifically, it needs to be IO9 because that's where we wired the LED on the schematic.



Start by setting the pin to be an output pin. We can do this directly in the Setup routine by dragging a pinMode routine on to the Flowline.

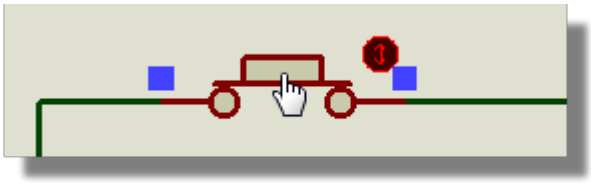


Then drag and drop the digitalWrite method into the interrupt handler, name the pin and set the state to true.

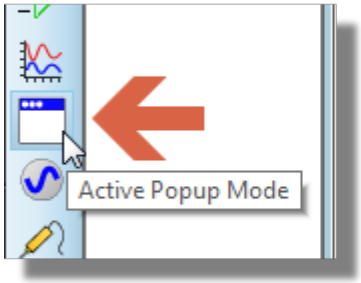


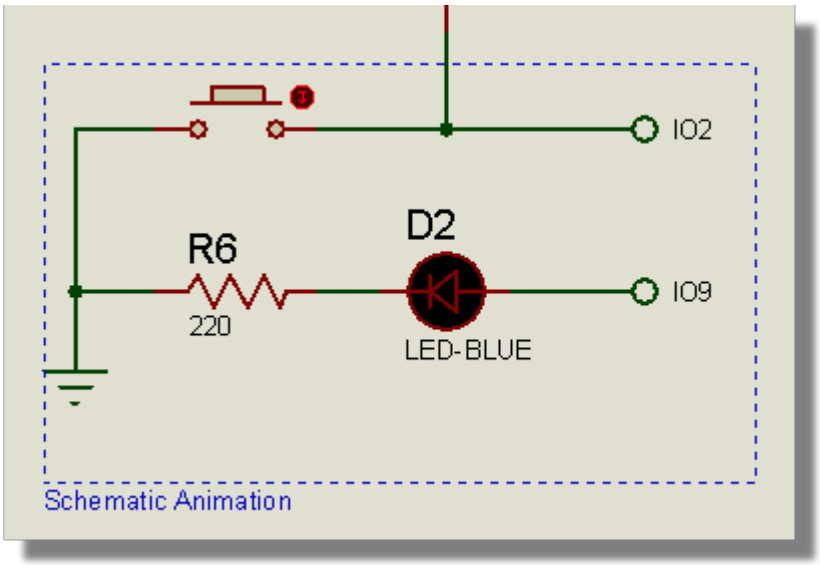
That's it. Press play to simulate, switch to the schematic tab and press the button to test.





To bring an area of the schematic into the Visual Designer we need to specify it as an Active Popup. To do this, stop the simulation, select Active Popup mode and drag a box with the left mouse around the components of interest.

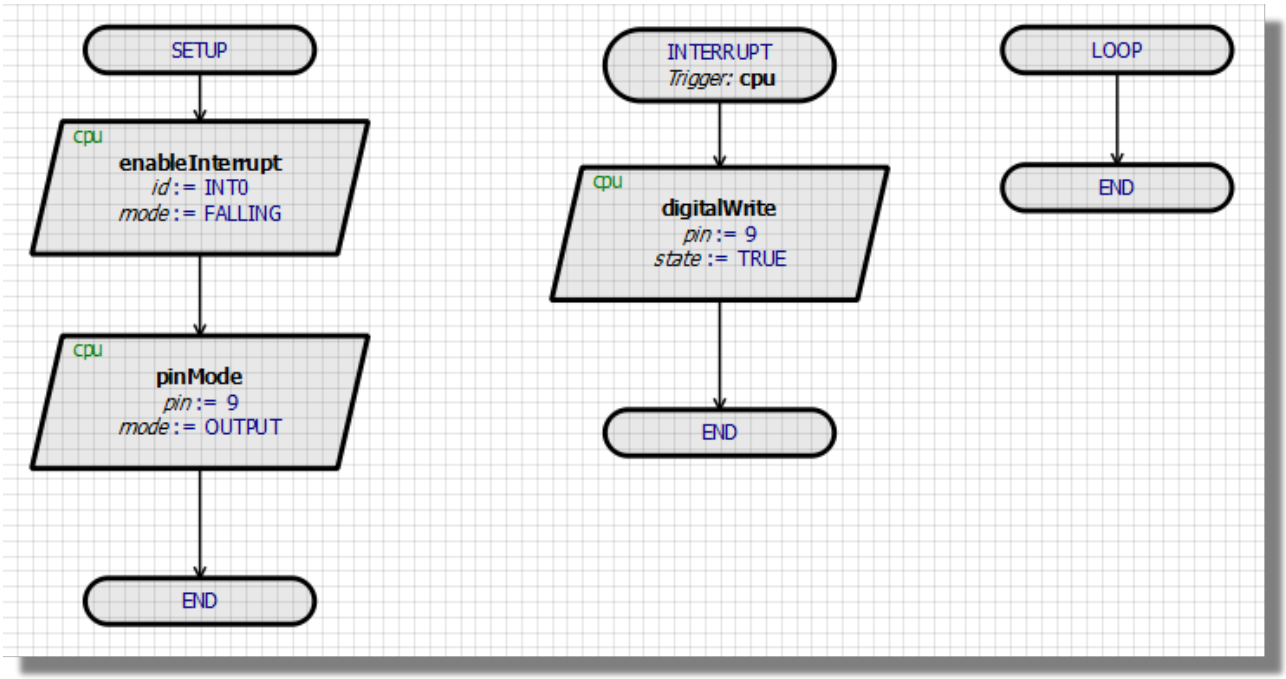




This time, when you press play you can test in Visual Designer via the Active Popups. It's a matter of preference in this case but, when debugging, it's often very useful to be able to see the circuit at the same time.

You can easily add a few blocks to change the program into a toggle or continue with developing the hardware on the schematic.

Your Flowchart project should finally look something like the following:





Interrupts are really useful but be aware that in the Arduino environment you are limited in what you can do. For example, attempting to write to an LCD inside an interrupt routine will simply fail because the Arduino stack for writing to the LCD itself uses interrupts. Caution is required.

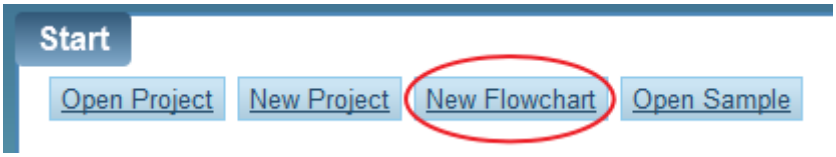
Tutorial 1 : Flashing LED

Introduction

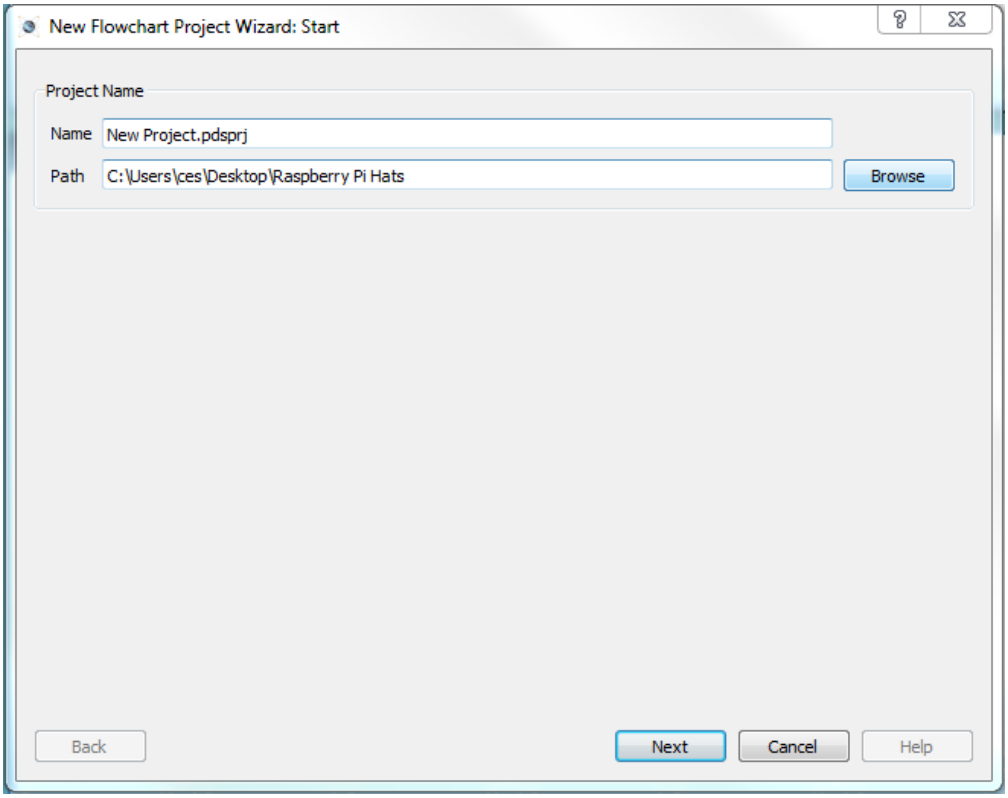
This tutorial covers the basics of project creation, working with flowcharts, raspberry Pi simulation and programming the real hardware. The actual goal of flashing the LED is kept as simple as possible so that we can cover the end-to-end process. Working with more complex peripherals and hats is covered in subsequent tutorials.

New Project Wizard

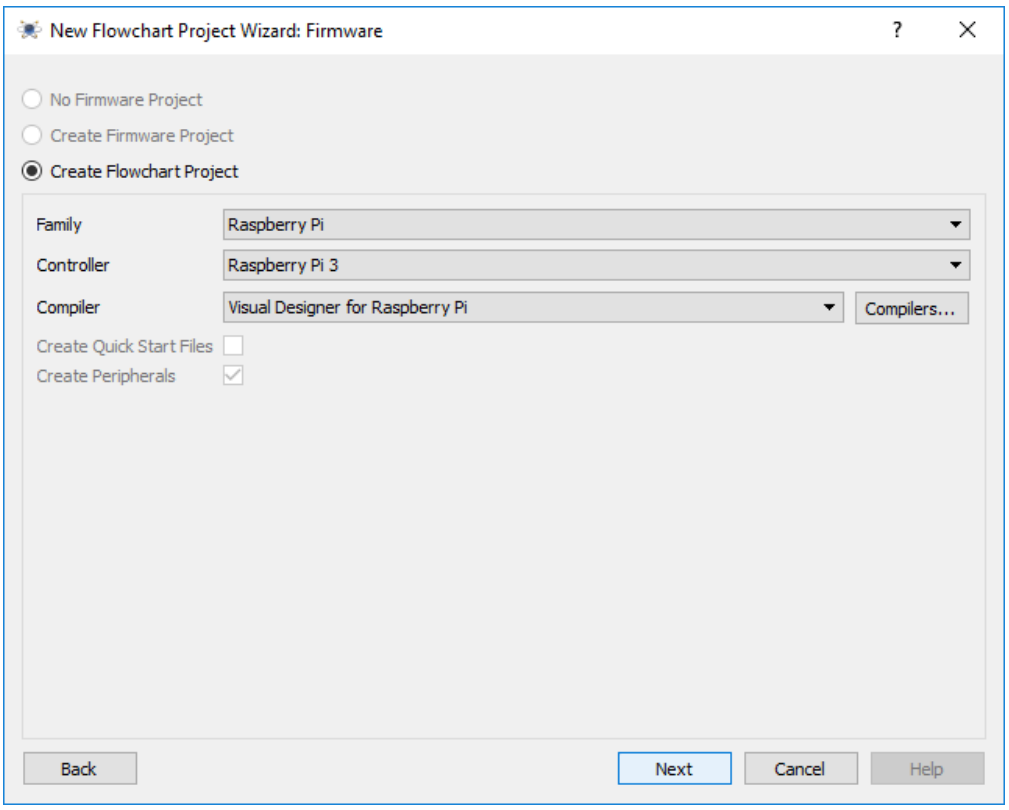
Visual Designer is integrated into the Proteus Design Suite and so we start our Visual Designer Projects from the Proteus Home Screen. The easiest and best way to create a Visual Designer Project is, unsurprisingly, through the New Flowchart Wizard.



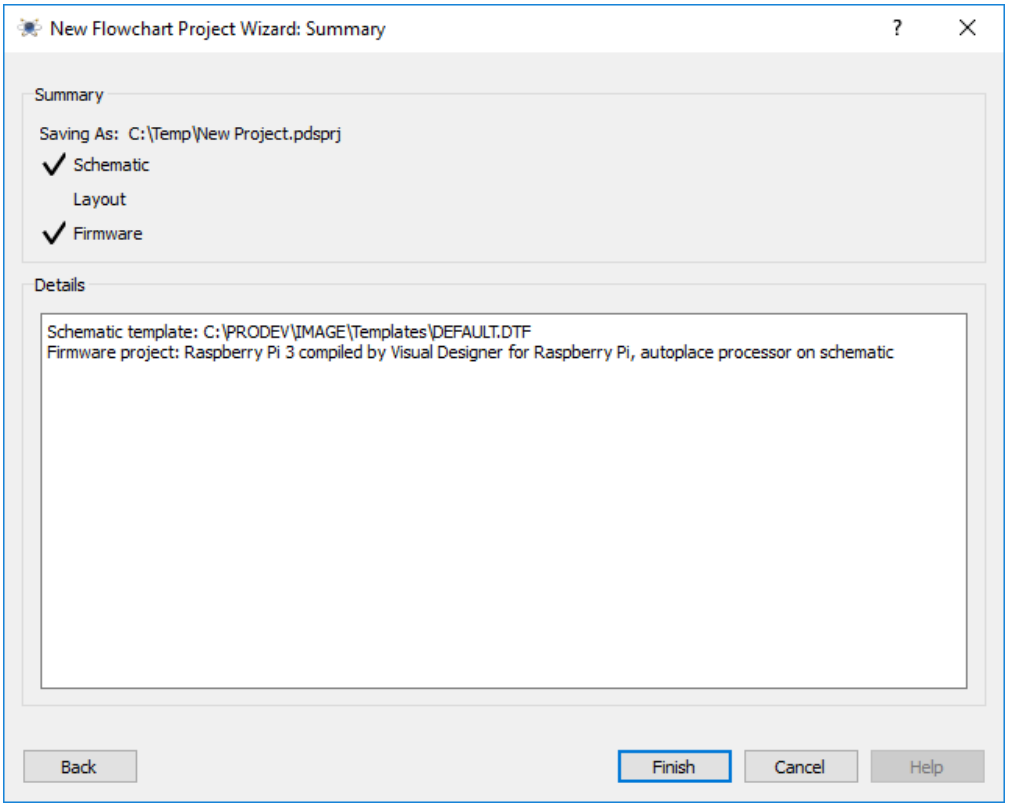
The first page allows you to select the name and destination of your project.



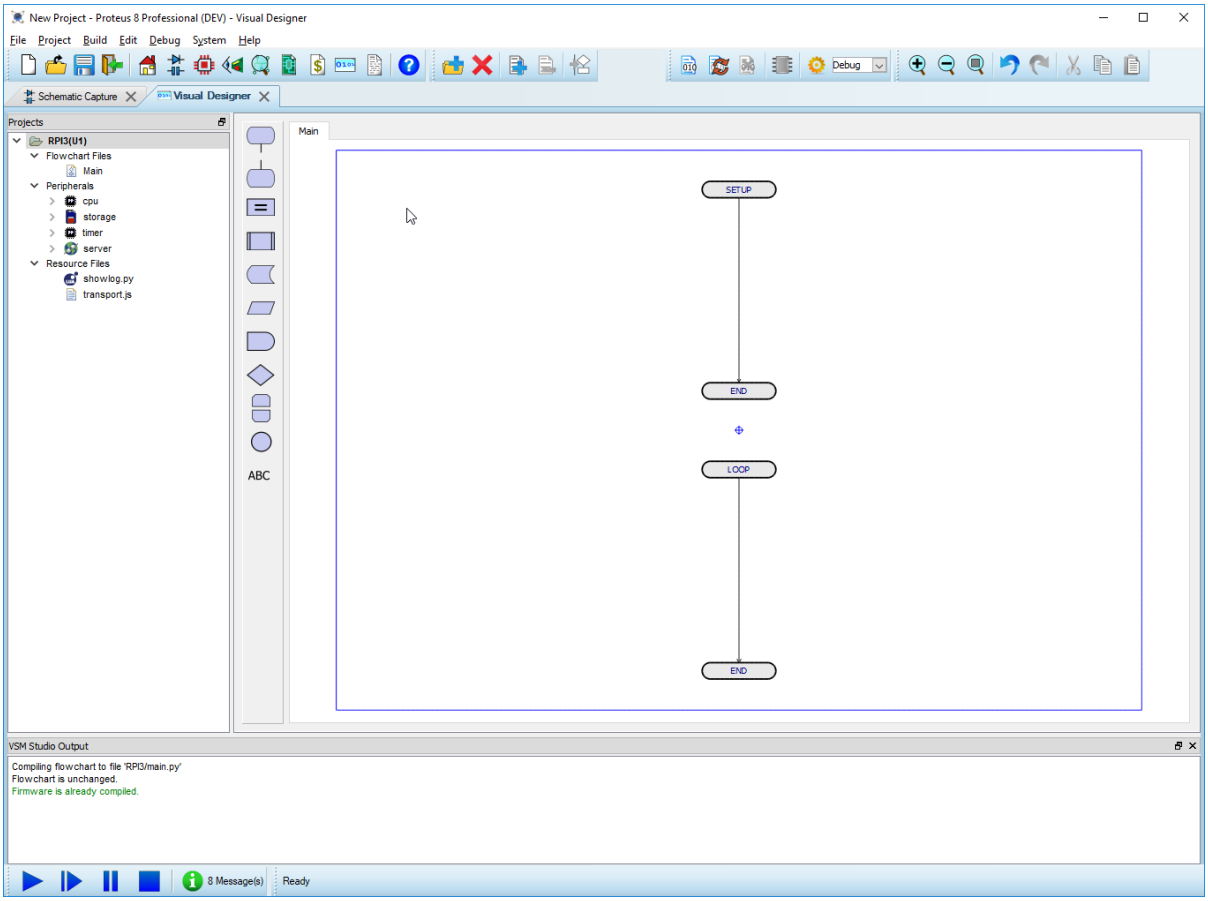
Next up is the firmware screen and it is here that we really define our Visual Designer project. First, we change the radio buttons at the top to the option for Flowchart Project and then we select the Raspberry Pi family and either the RPi3 or the RPi3 + GrovePi Hat from the controller combo box.



Finally, you will be presented with a summary page of your configuration.



After you exit the dialogue form the project will be created and you will see both a skeleton schematic with the Raspberry Pi processor placed and a skeleton chart project with the familiar Setup and Loop routines on the Editing Window.

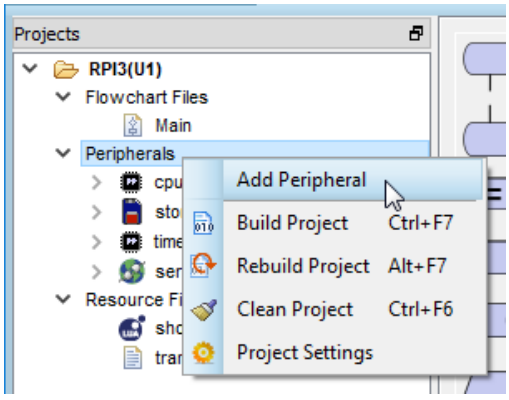


Now we are ready to start designing our project.

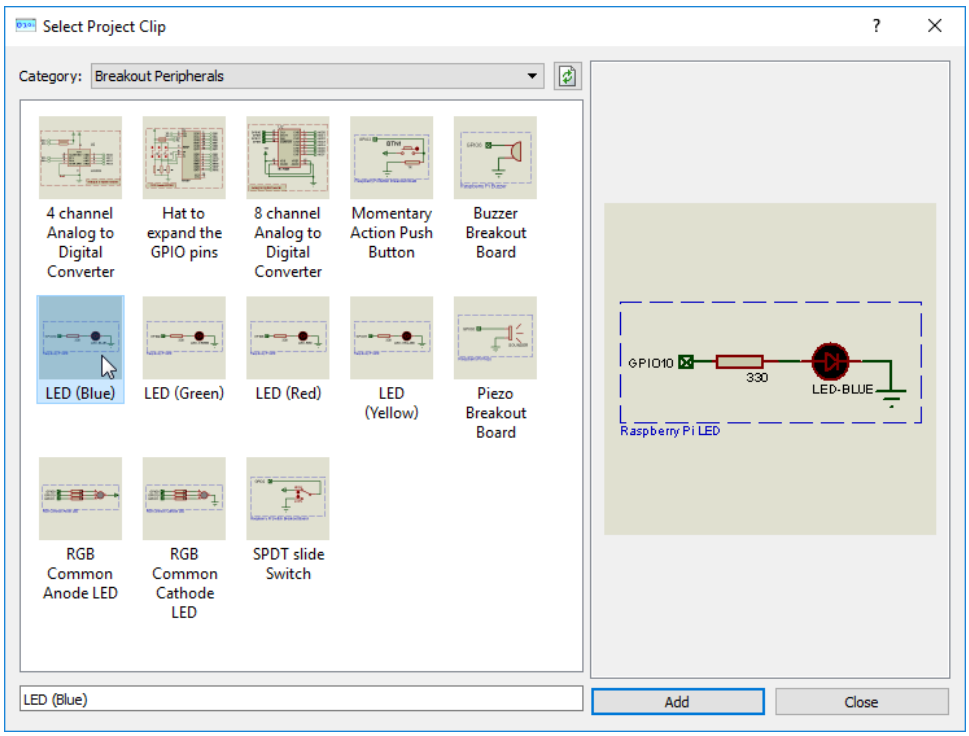
Adding a Peripheral

The first thing to do is to add the peripheral(s) for the project. These can be Raspberry Pi Hats or just little breakout peripherals. The Raspberry Pi does not need a Hat to operate and you can add peripherals which talk to the GPIO Pins Directly.

We'll be adding a Breakout LED. In Visual Designer, we right click on the Project Tree and select the Add Peripheral command. Next, switch to Breakout Peripherals and select one of the LED's.

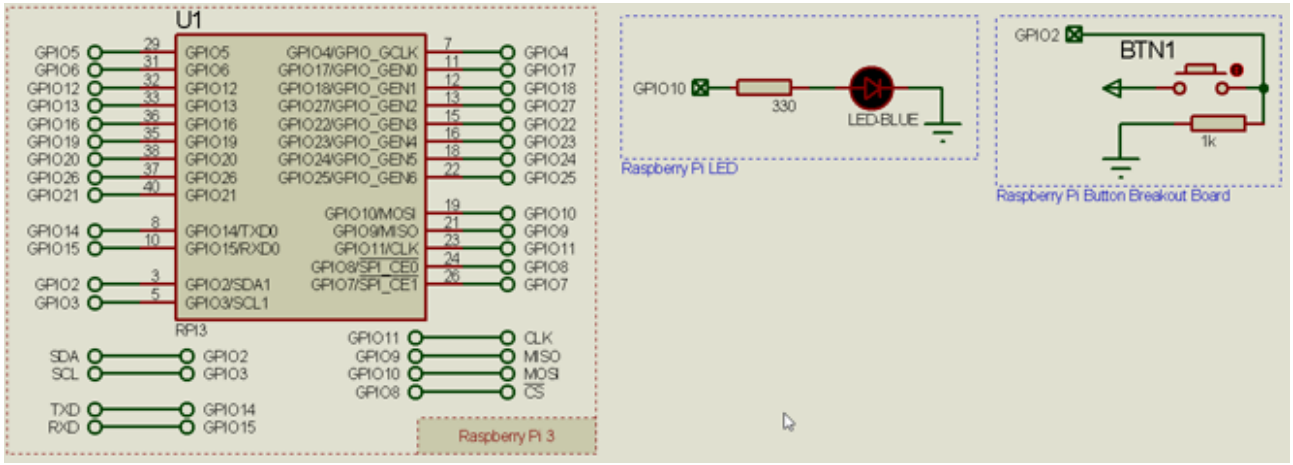


Add the Peripheral from the context menu command in VSM Studio



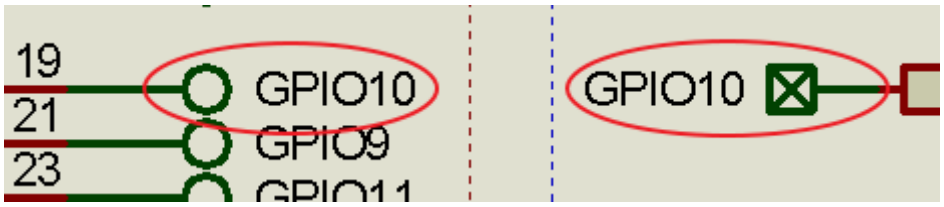
Double click on the LED to add it to the project.

Repeat the process to add a Breakout button. If you then switch to the schematic tab you will see that the 'virtual hardware' has been autoplaced for you.



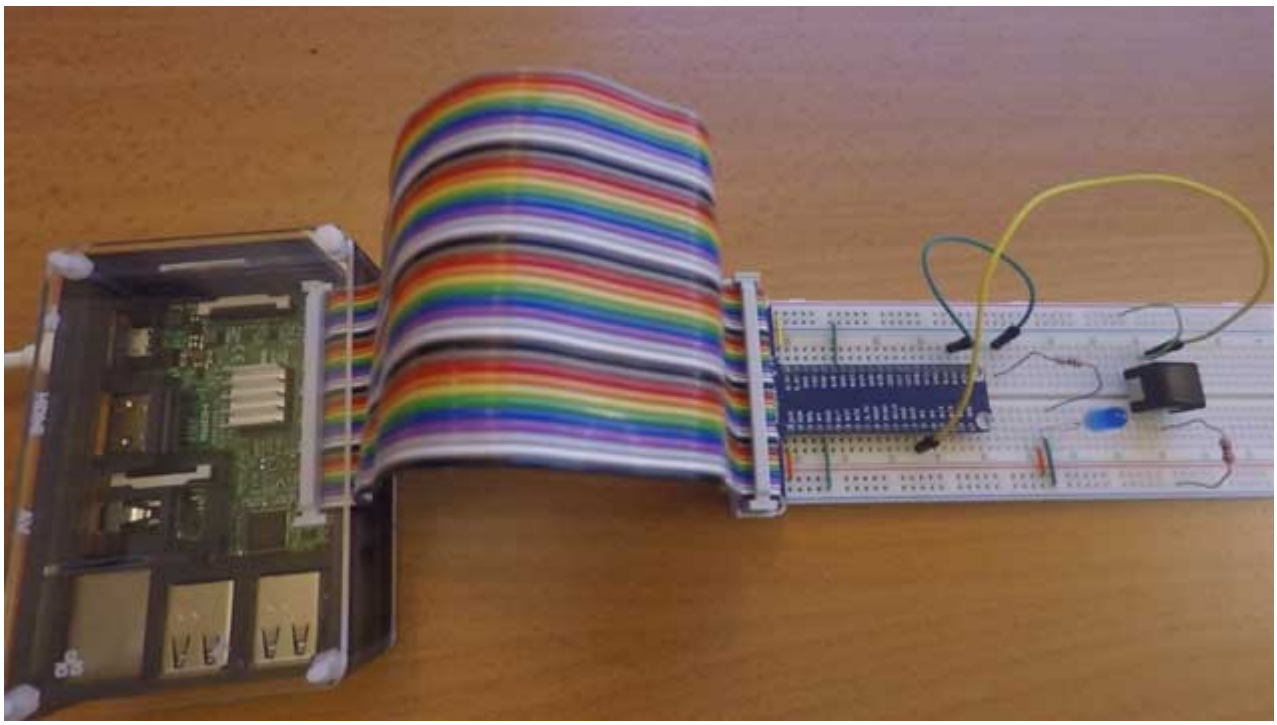
Schematic view with both LED and push button auto-placed.

To avoid clutter on the schematic, connections are made on the schematic by giving terminals the same name. Any terminals with the same name can be thought of having an invisible wire between them.

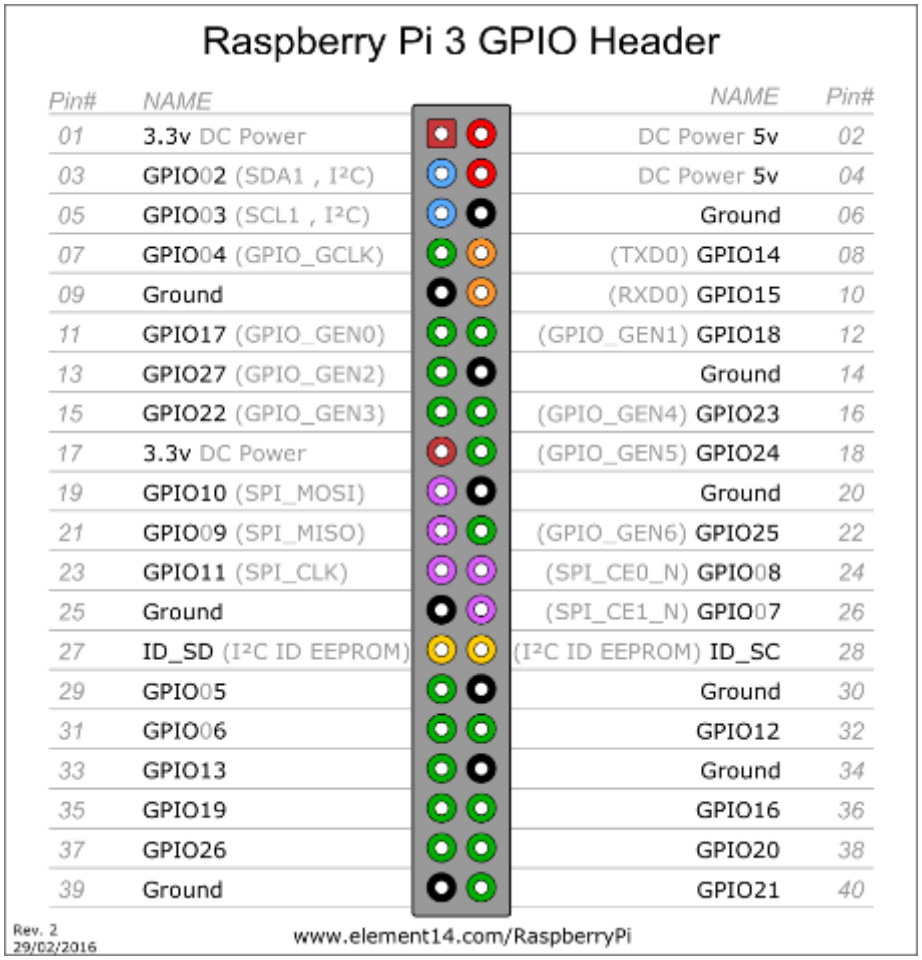


Terminals with the same name are considered connected together.

In real hardware, this equates to connecting the Breakout button and Breakout LED to the GPIO Pins on the Raspberry Pi

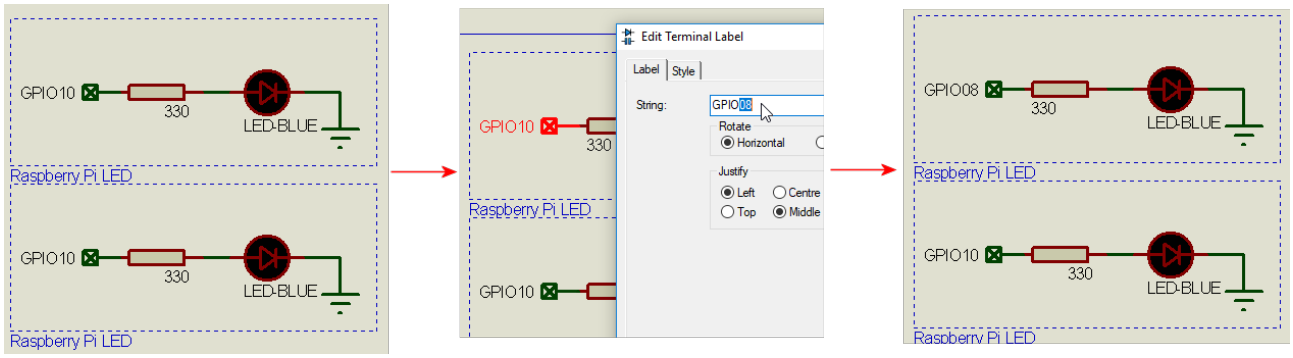


Note that you must attach the peripherals onto the same GPIO pins on the Raspberry Pi as they are designated on the Proteus schematic. You may need the pinout to see which GPIO pins go where.



GPIO pins on the Raspberry Pi

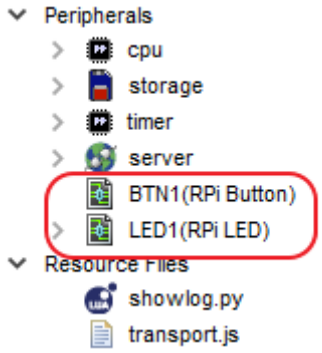
You must also ensure that you don't have two peripherals with the same terminal label on the schematic; if you do, you need to switch to the schematic and edit the terminal labels on one of the peripherals. Some pins may need unlocking.



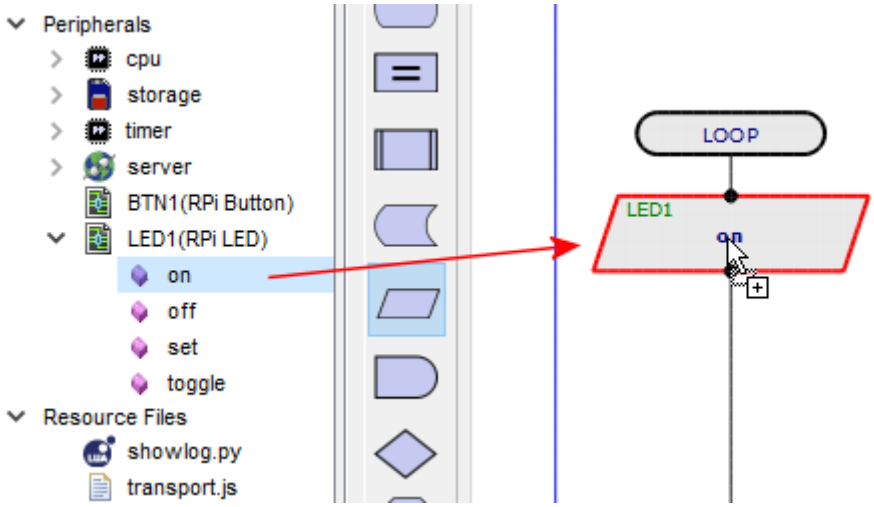
If you have two terminals with the same name they will be connected ! Need to edit one and re-assign to another GPIO line.

Flowchart Design

Back in Visual Designer, you'll notice that the Project Tree now has two entries under the peripherals section.

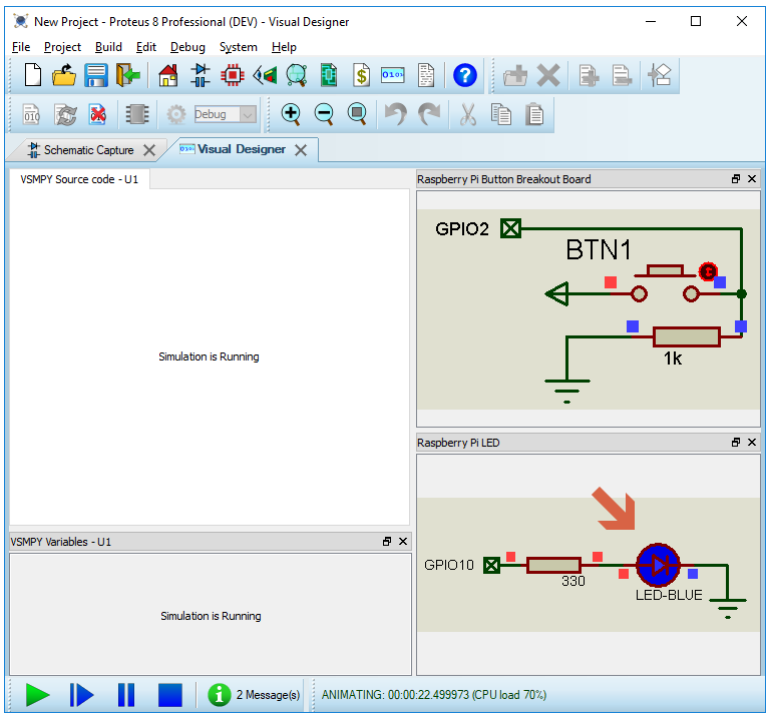


The beauty of Visual Designer is that you can expand the peripherals to find a list of methods. These methods are the primary way in which we interact with the hardware. For example, drag and drop the 'On' method of the LED into the loop routine



Drag and Drop methods onto the program.

To test this simply press the play button. The program will compile, the simulator will start and the LED will turn on. You can see this either via the Active popup at the right of Visual Designer or by switching to the schematic tab.



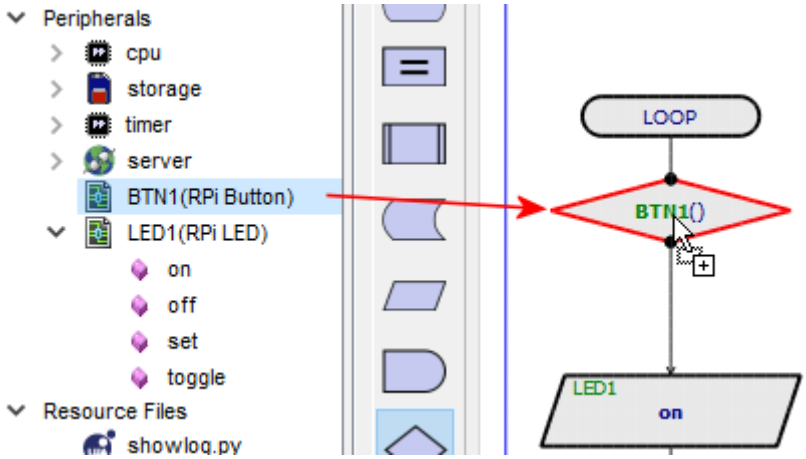
Run the simulation to test results.

Press stop to exit the simulation and we'll extend our program to switch the LED on and off with a button press. You'll notice that the button has no methods in the project tree



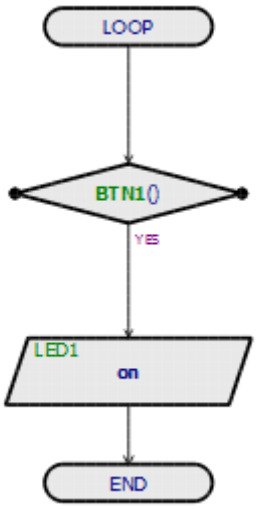
Stop the simulation when finished.

The only sensible information we need from the button is to know whether it has been pressed and for that we have what we call a 'sensor function'. A sensor function returns either TRUE or FALSE depending on whether the basic peripheral function is true or not. For example, the sensor function for a button will return TRUE when the button is pressed and FALSE if it isn't. Similarly, the sensor function for an LED will return TRUE when the LED is on and FALSE when the LED is off. Place the sensor function for the button by dragging and dropping from the button itself onto the loop function of the flowchart

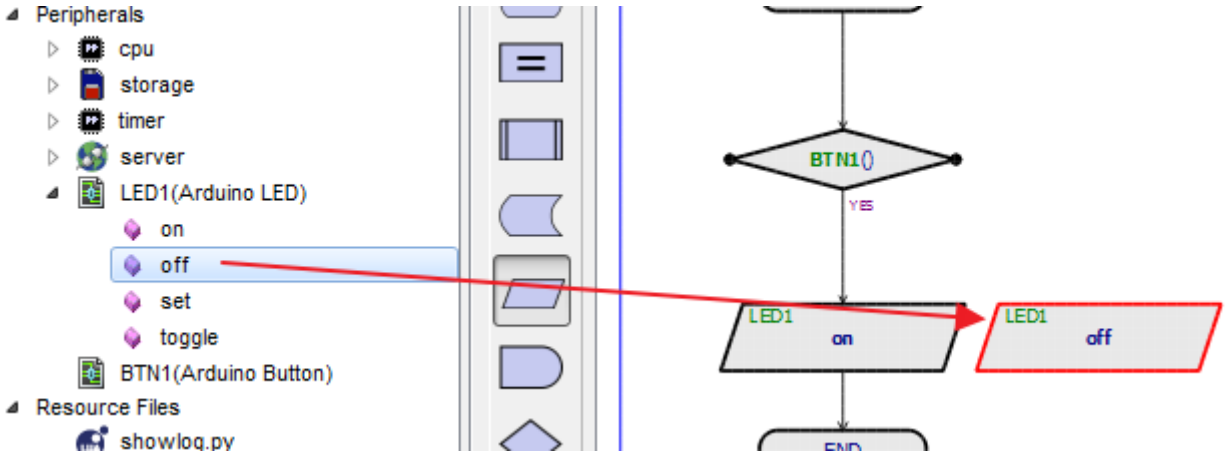


Drag the sensor function onto the flowchart

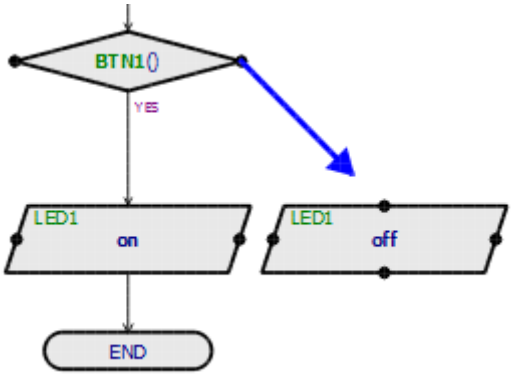
You will notice that it appears as a decision block. Sensor functions always return TRUE or FALSE and a decision block allows us to split our code into two conditional paths. Your loop routine should currently look something like the following.



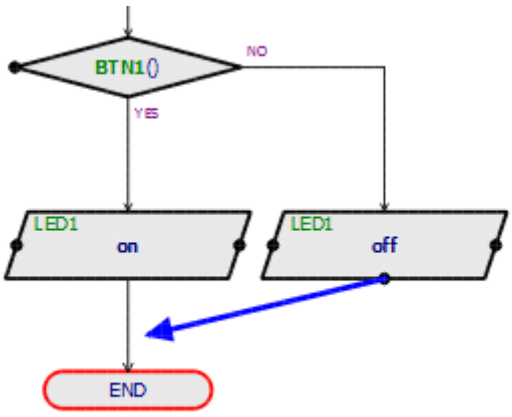
We are constantly testing the button to see if it is pressed and we are turning on the LED if it is. However, we still need to tell the program what to do if the button is not pressed. We want the LED to turn off so lets drag and drop that method into space alongside the ON operation.



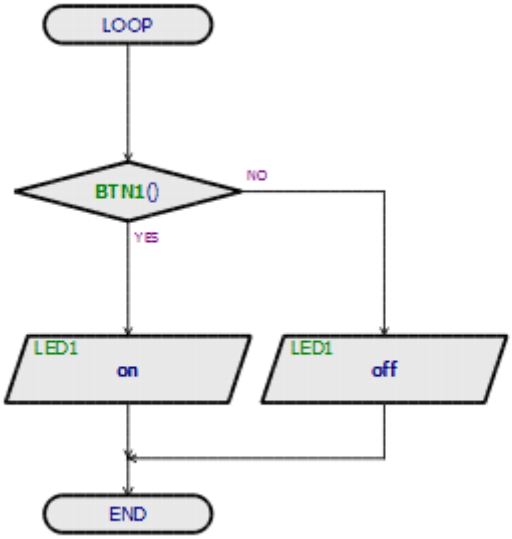
Now, we need to wire from the decision block to the top of the OFF command. To place a flowline click on an output node (the decision block), move the mouse to an available input node (top of the OFF routine) and click the mouse again.



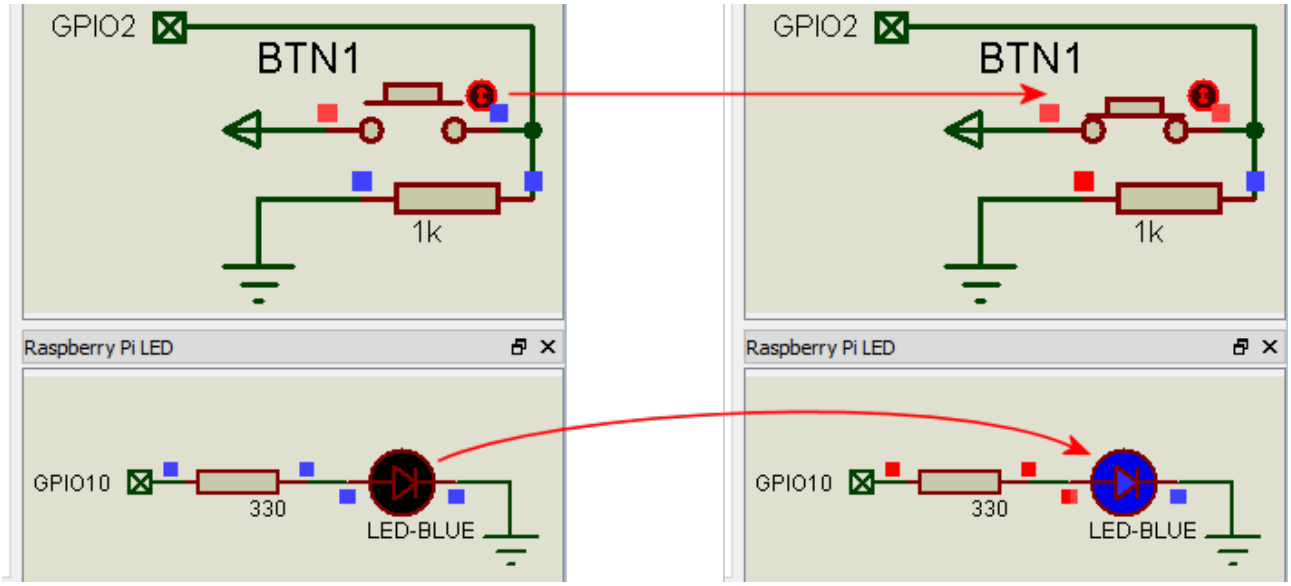
Similarly we need to connect the flow from the bottom of the OFF command back into the main loop.



Your program should now look like the following.



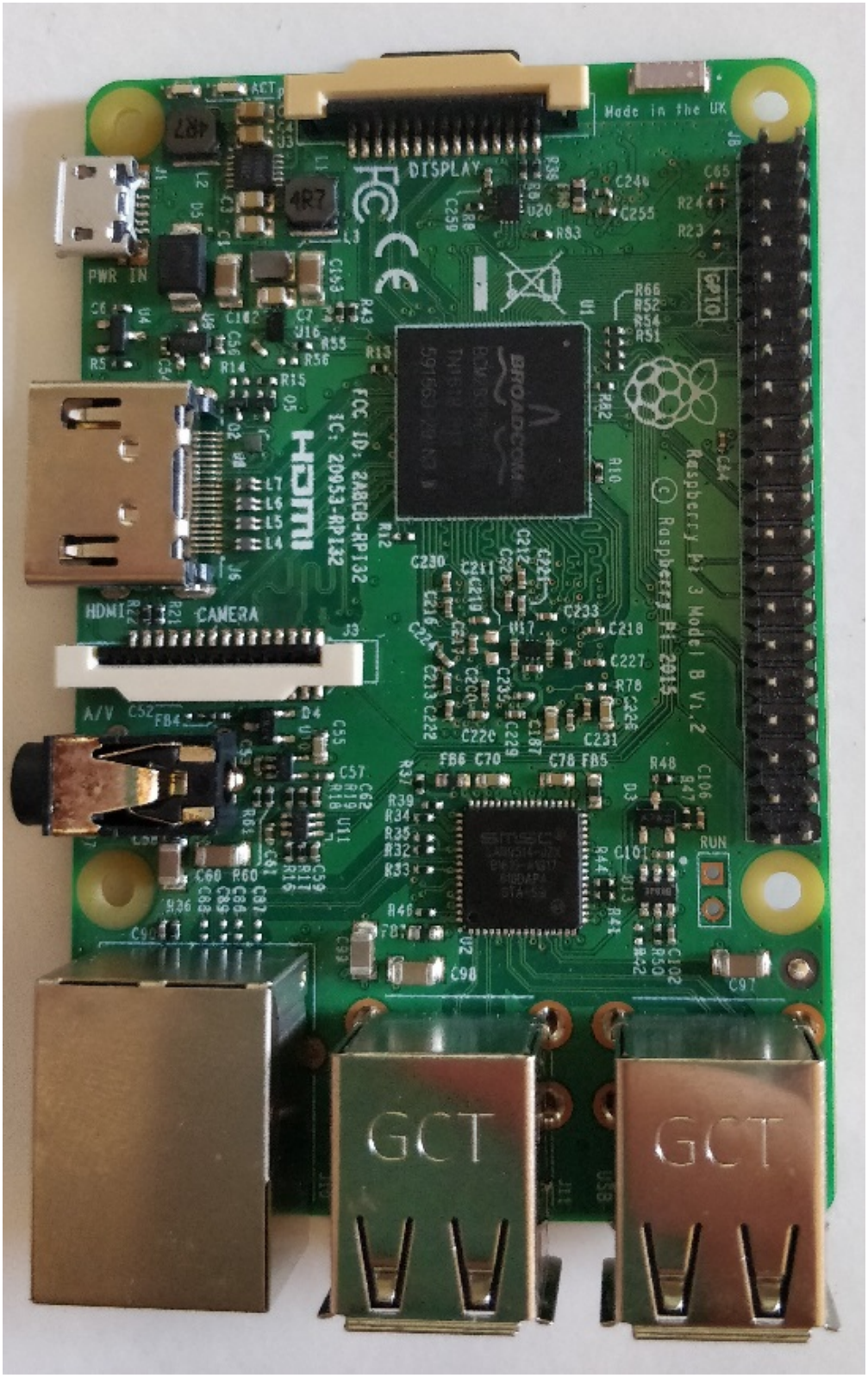
Having made this change, our program now turns the LED off when the button is down and on when the button is released. To compile and test, first press the play button and then use the mouse to press the button down..

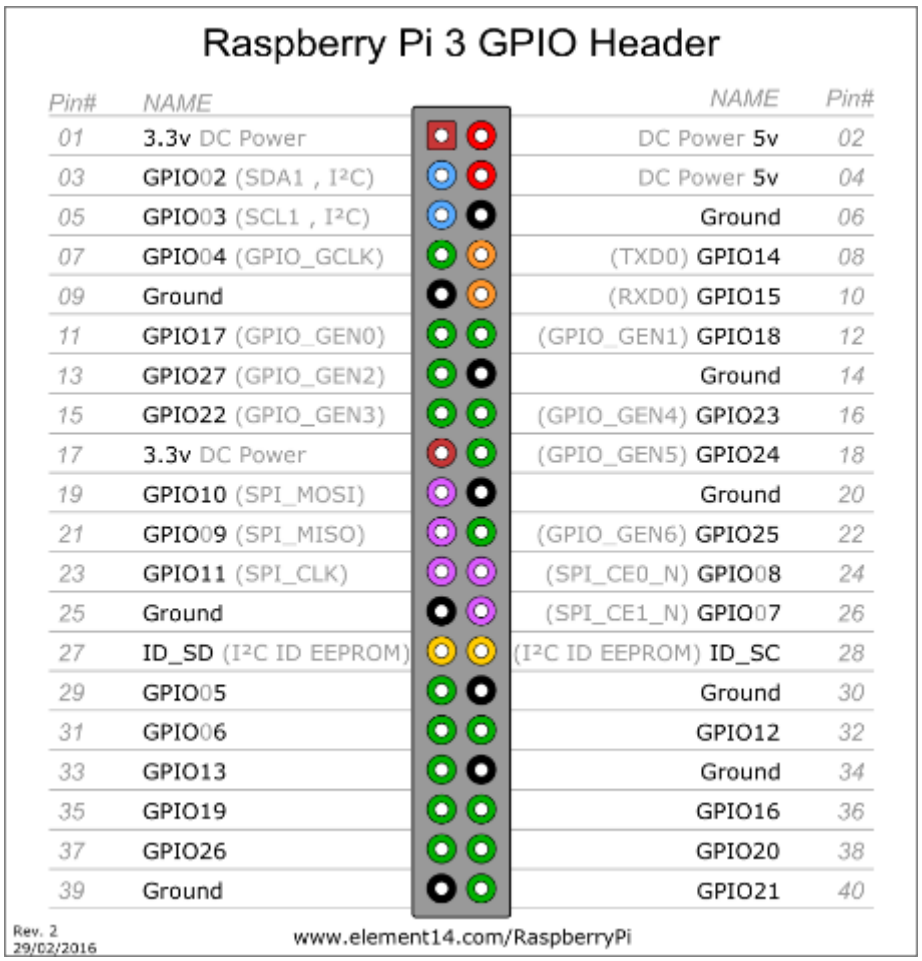


Programming the Hardware

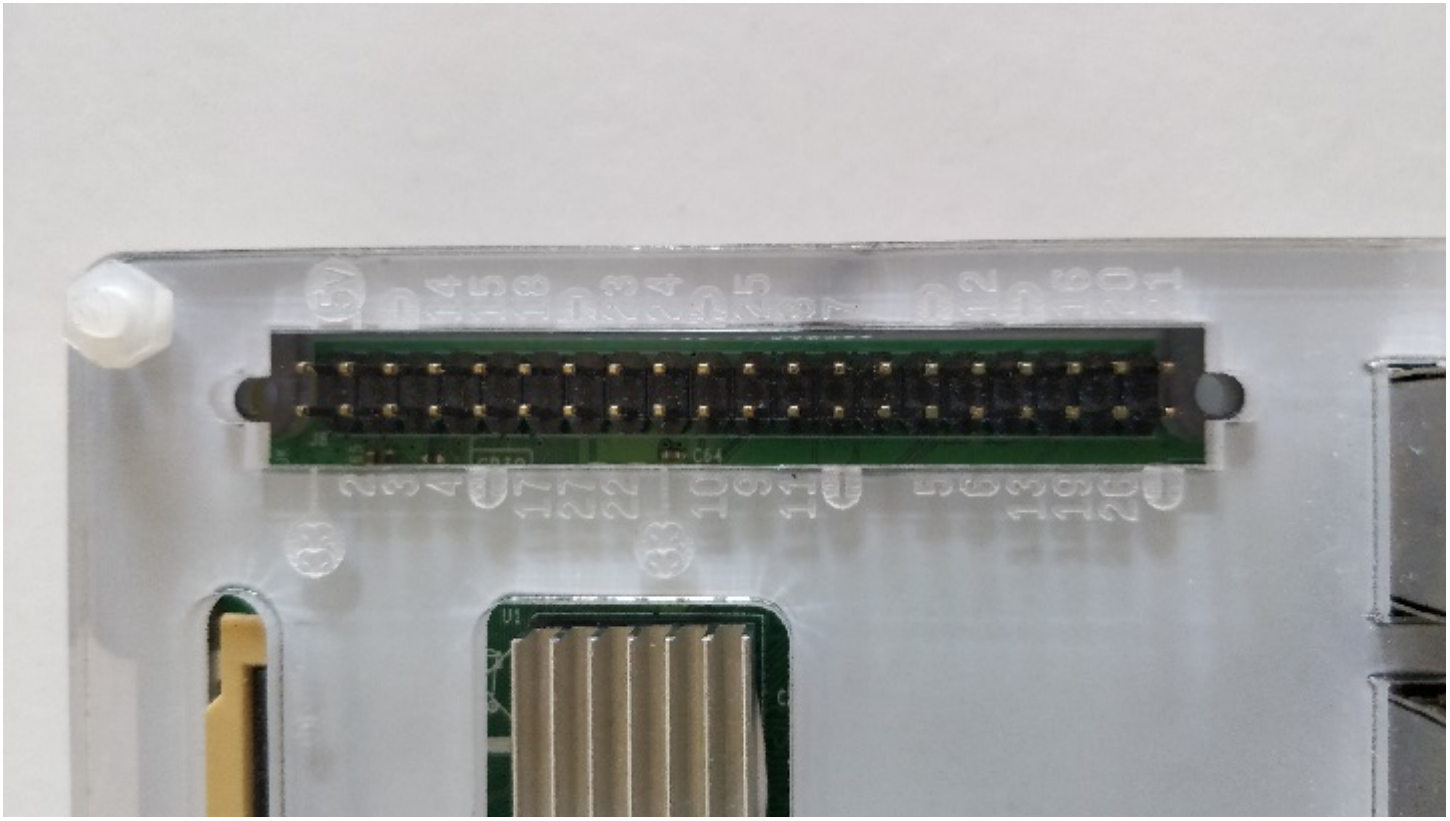


This section assumes that you have completed the initial configuration of the Raspberry Pi.


DISPLAY
PWR IN
HDMI
CAMERA
A/V
RUN
Made in the UK
Raspberry Pi 3 Model B V1.2
GCT
GCT



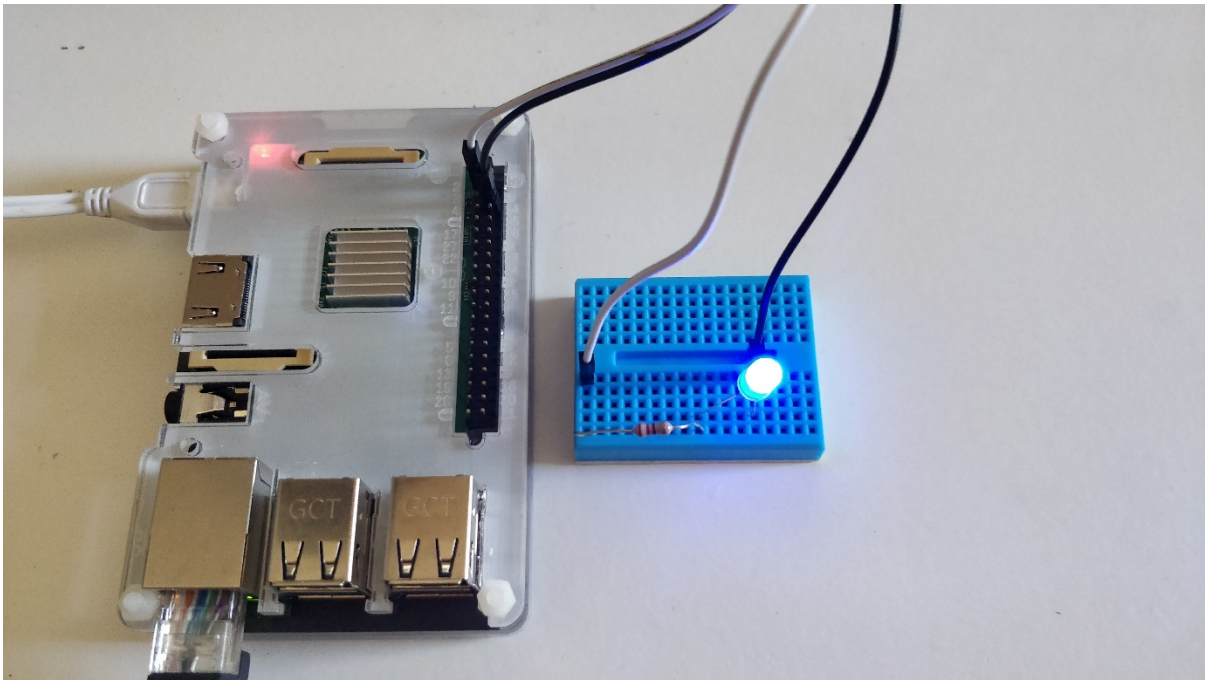
Here on the Left is a Raspberry Pi 3, and on the Right is the pinout for the GPIO Pins. The very first thing you need to understand is the orientation. Both these images are oriented in the same way. The top 2 pins are both power (Left pin is 3.3V and Right pin is 5V). You can find this pinout on the raspberry pi website. Some of the Raspberry Pi I cases have the pinout engraved into it making it very easy to find each pin. Failing that, you can use a template or make one with a little bit of card to make life easier.



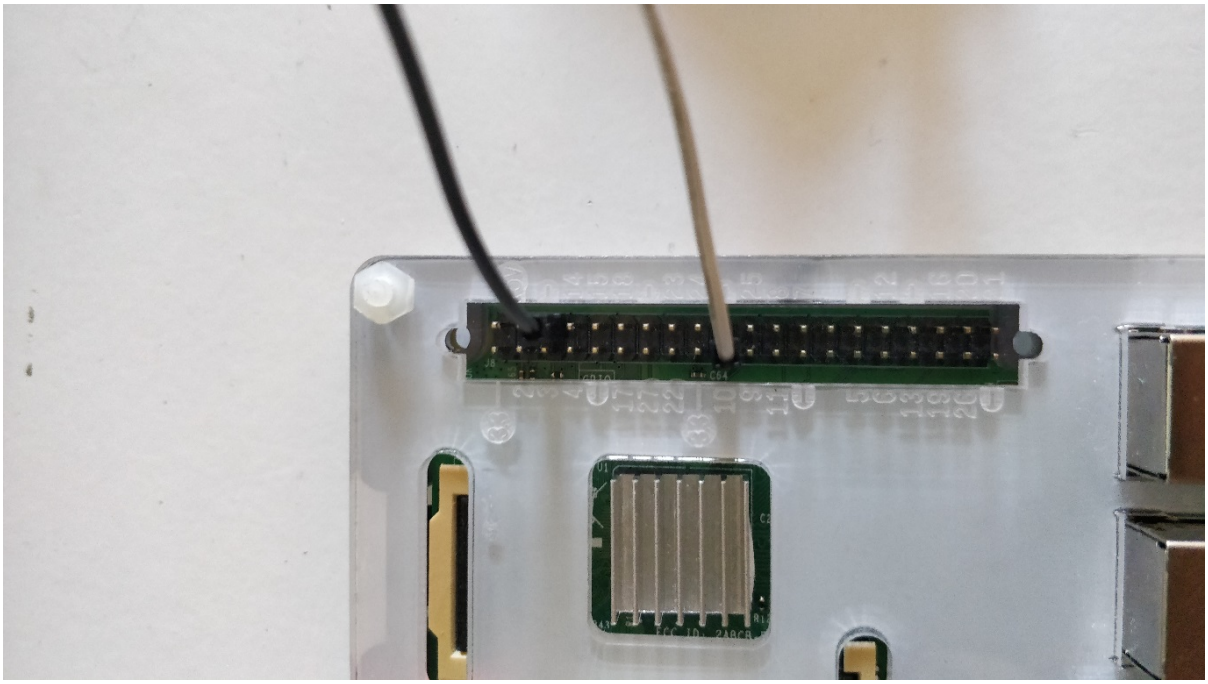


Engraved casing or simple template to identify lines on the header block.

Now in our program we connected the LED to GPIO 10. Not to be confused with Pin 10 which is GPIO 15. GPIO and Pin numbers are very different so make sure you connect the correct one. Just like the schematic we've included a protection resistor to prevent he LED from getting damaged by the current from the Raspberry Pi.



Here you can see we've used Male to Female wires connecting the Raspberry Pi to the Breadboard.

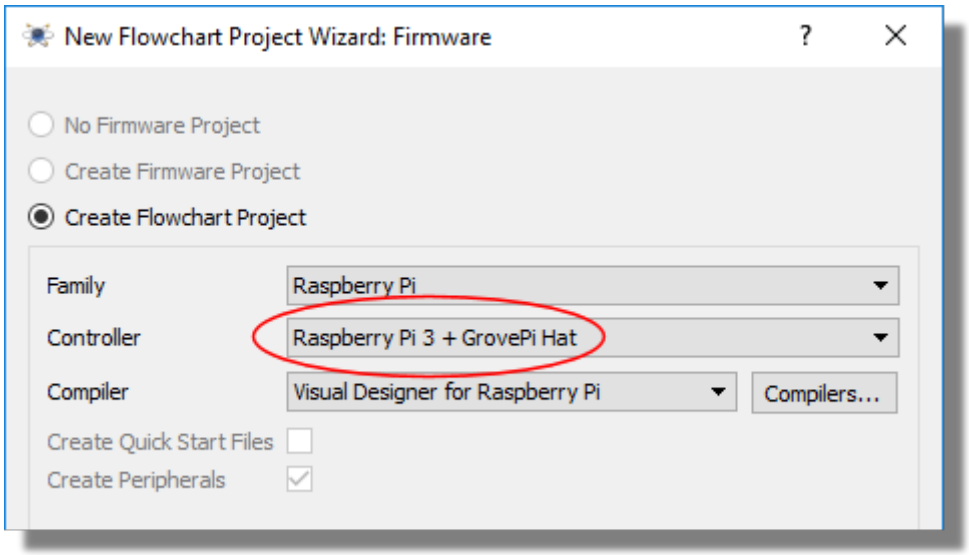


Assuming all is correct your led should flash just like the project.

Tutorial 2 : Night Light (w/ Grove)

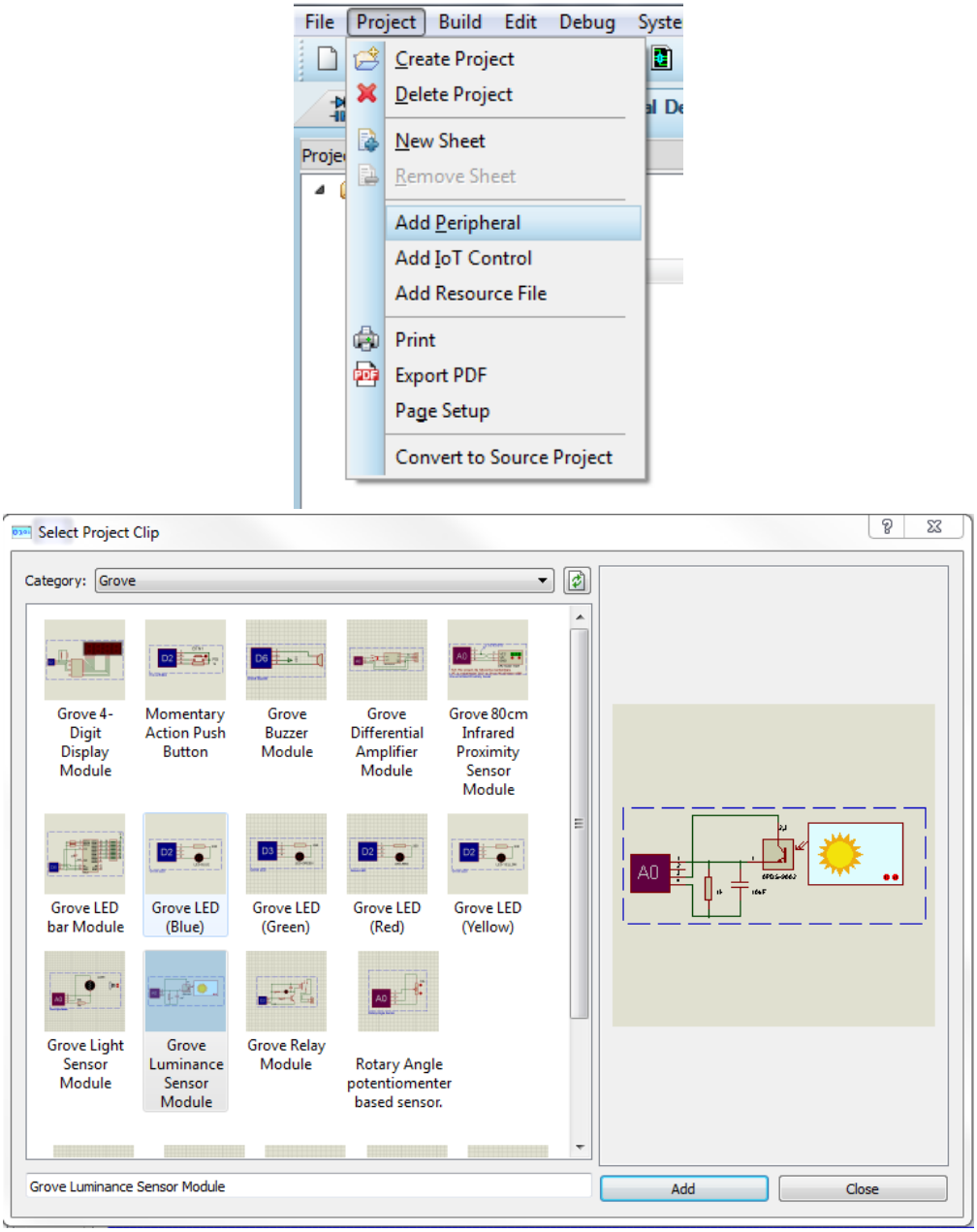
Introduction

In this tutorial we are going to use a couple of Grove sensors and an LED to design a mini nightlight. Project setup is covered in the first tutorial however this time we will want to change the controller to the RPI3 + GrovePi Hat.

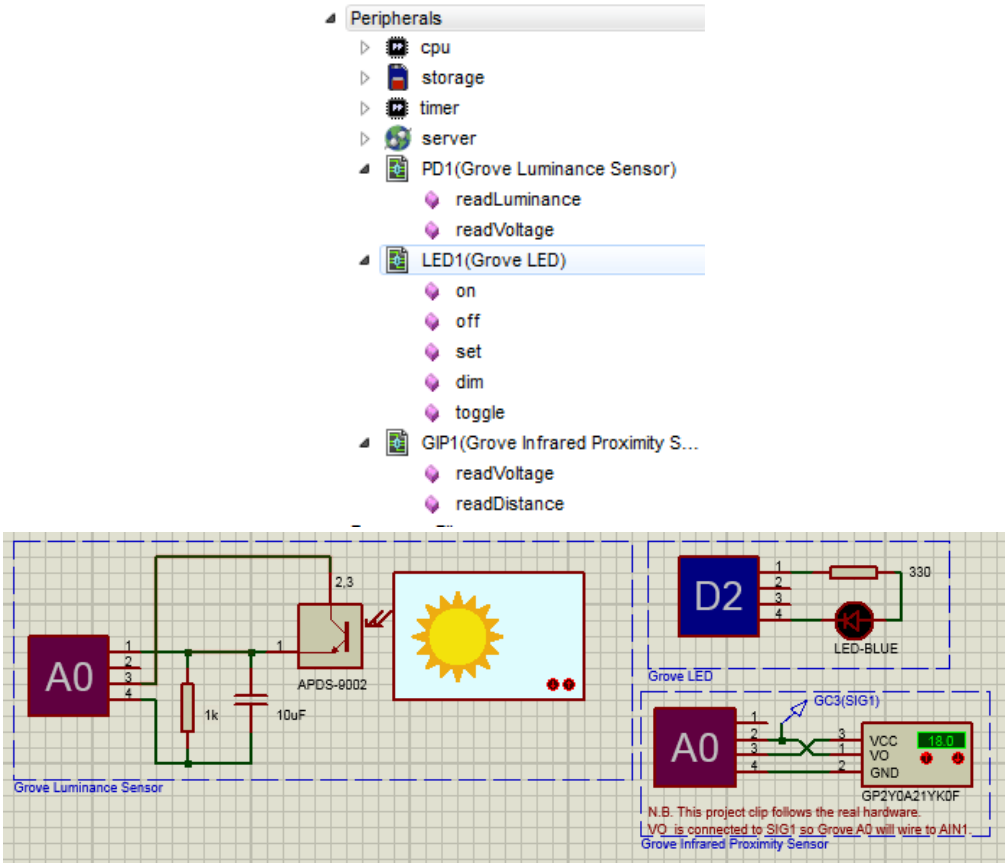


Adding Hardware

We'll use the Grove infra-red proximity sensor and the Grove luminance sensor along with a Grove LED. All of these can be picked via the Add Peripheral command on the Project Menu.



After they have been added to the project, you will see the methods for controlling the peripherals in the Project Tree and the 'virtual hardware' for the peripherals placed for you on the schematic.



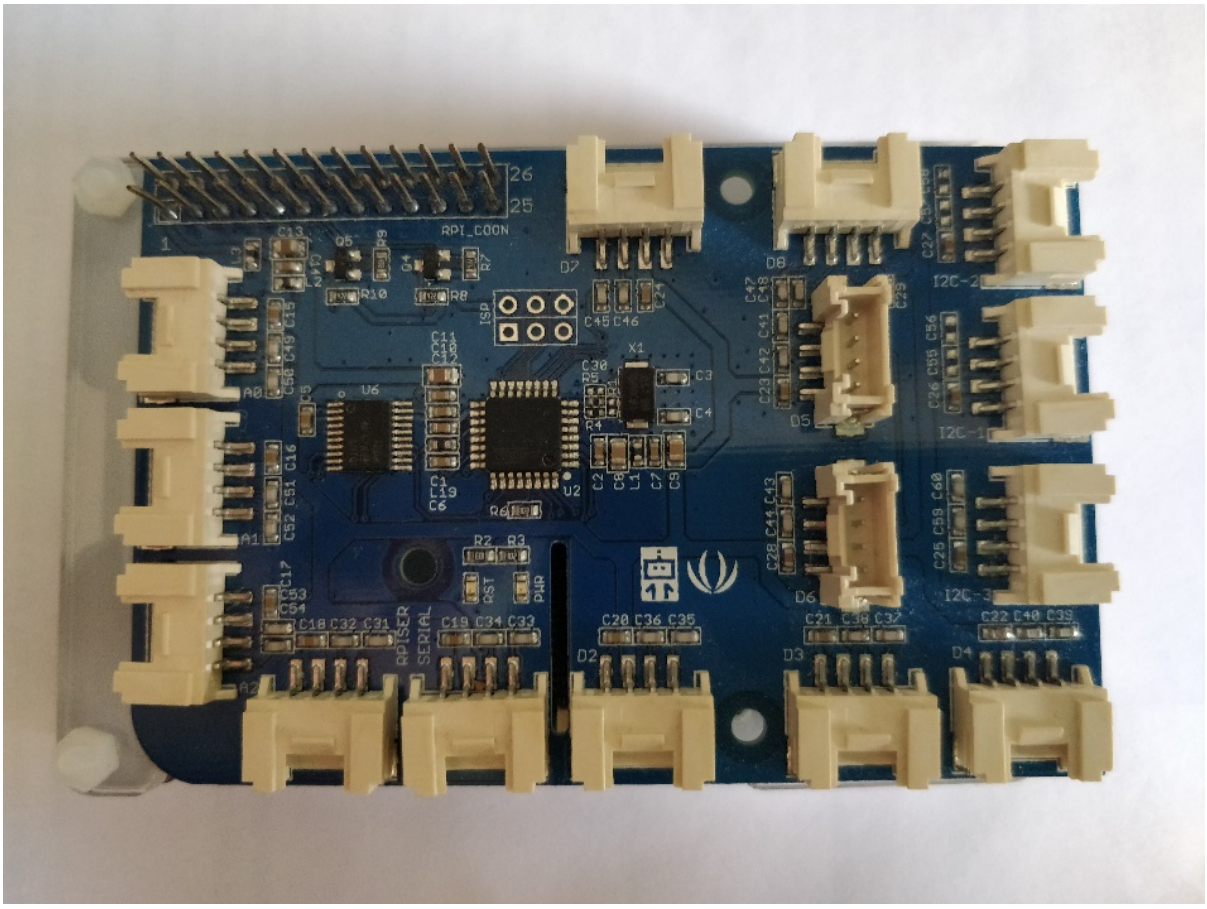
It is vitally important to configure the Grove connectors properly and this is often not as simple as it first appears. Let's take an example:

The real proximity sensor requires only three wires with the fourth left NC. Note therefore that, when plugged into socket A0 the sensor is connecting to analogue pin A1.

This is fine, except if you then plug the luminance sensor into socket A1. The luminance sensor will use analogue pin A1 on socket A1 and therefore clash. You should plug this sensor into socket A2.

Golden Rule

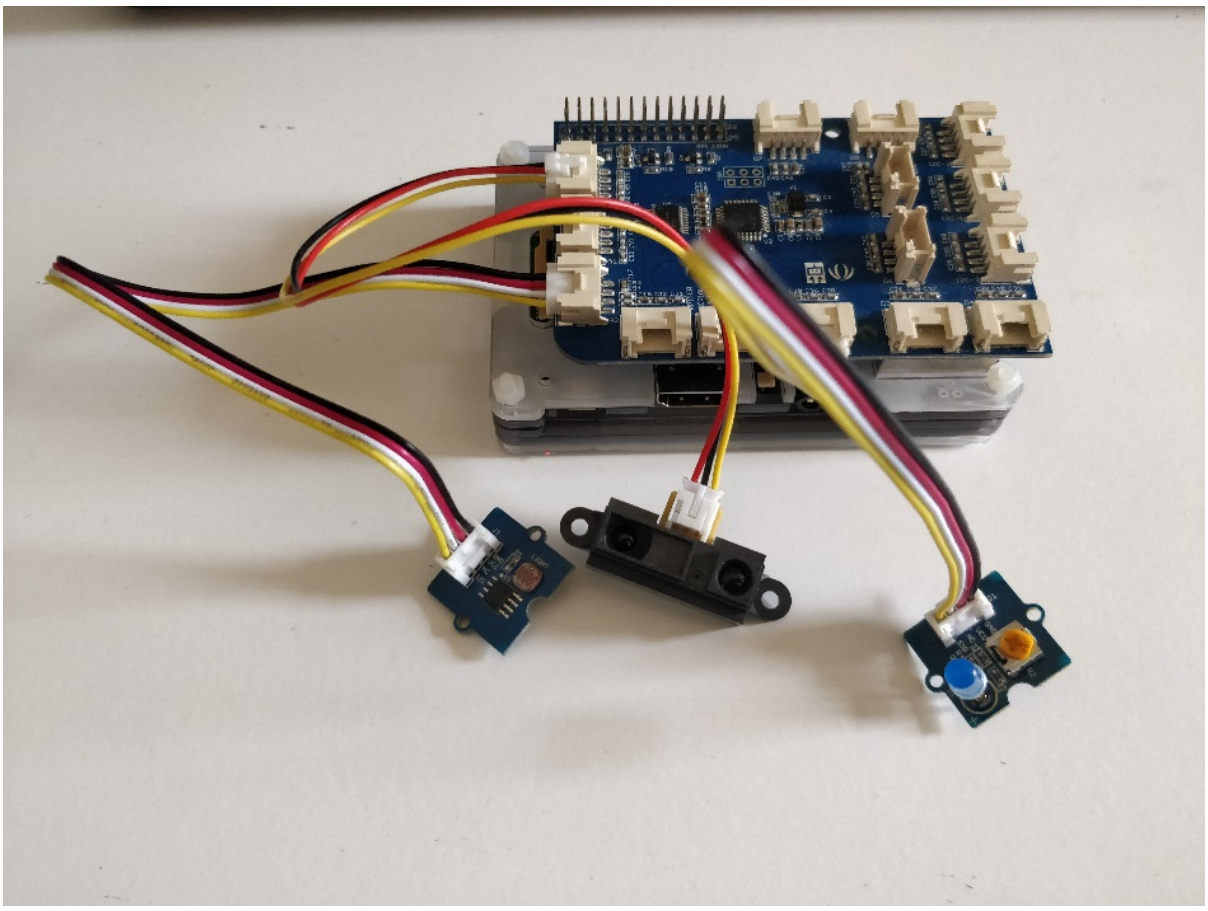
Look closely at the Grove board to see the connections and double check against the sensors for conflicts. Unlike the Arduino the conflicts are not shown on the board and need to be checked separately.



The sockets include connections through to two Raspberry Pi IO pins and there is a pin overlap. For example:

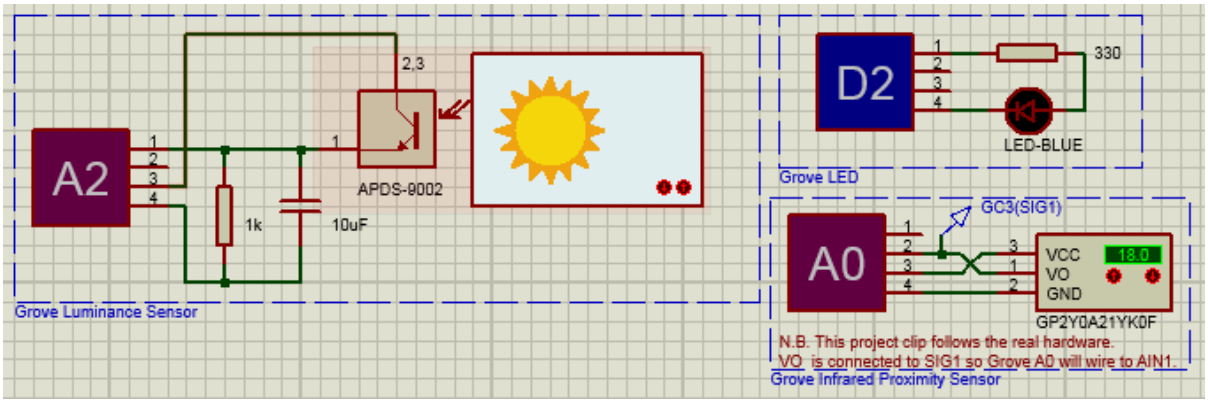
- Socket A0 can use pins A0 and A1
- Socket A1 can use pins A1 and A2
- Socket A2 can use pins A2 and A3.
- etc

Digital sockets are exactly the same.



Grove Peripherals connected.

Our configuration for this project is therefore proximity sensor in socket A0 and luminance sensor in socket A2. The LED can be in any digital socket we like.



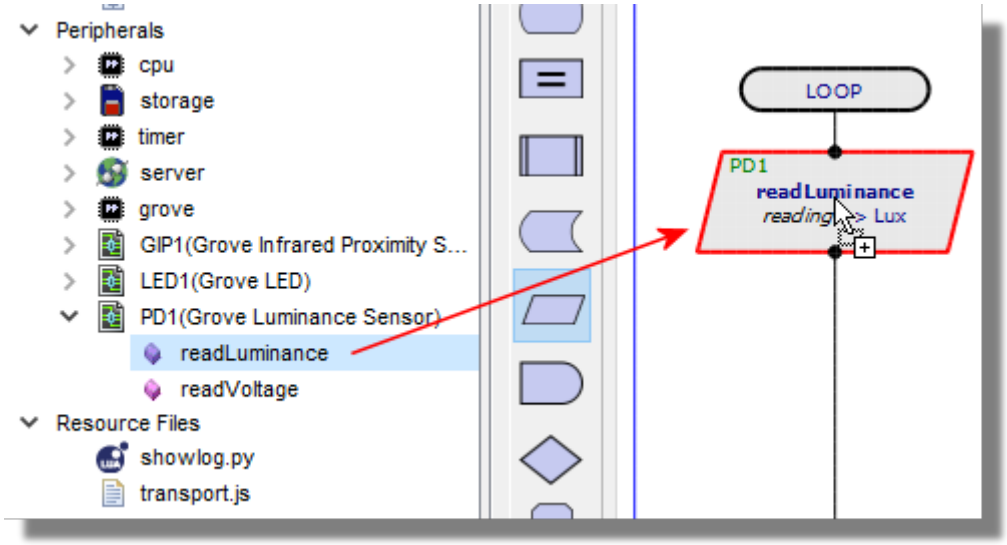
Designing the Program

In our Nightlight mock up we are looking for our LED to light when the following are true:

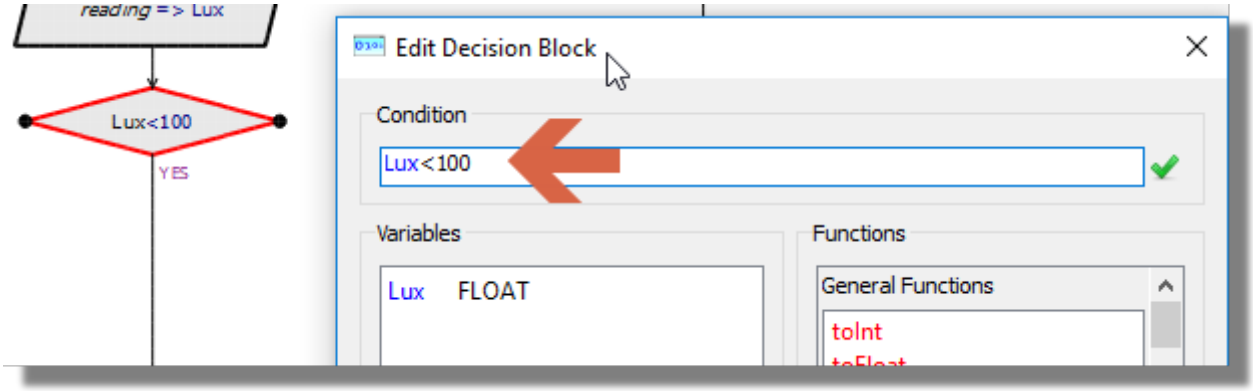
- It's is dark.

- Someone is nearby.

These are both decisions on the flowchart. Start by drag and drop the readLuminance() method onto the loop routine of the chart.



This sensor returns a lux value between 0 and 1000, with 0 being pitch black and 1000 max light. Let's then use our result in a decision block and set our initial test value to around 100.

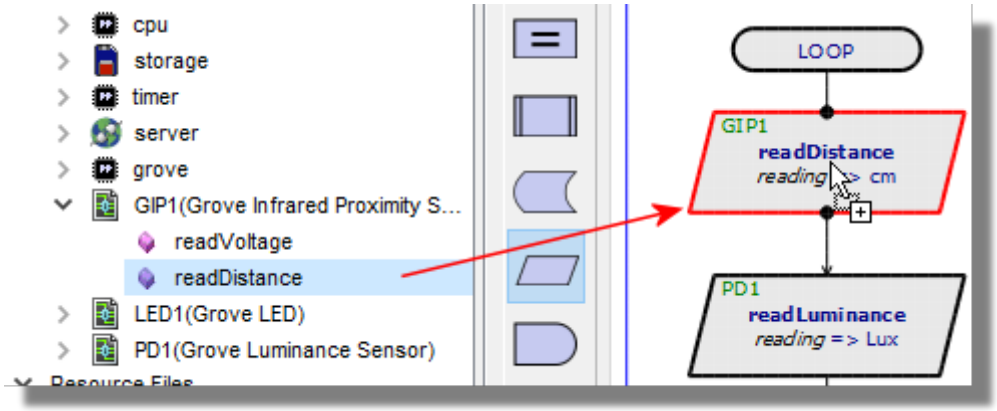


The datasheet for the APDS-9002 photo sensor used in this Grove module provides data on lux values.

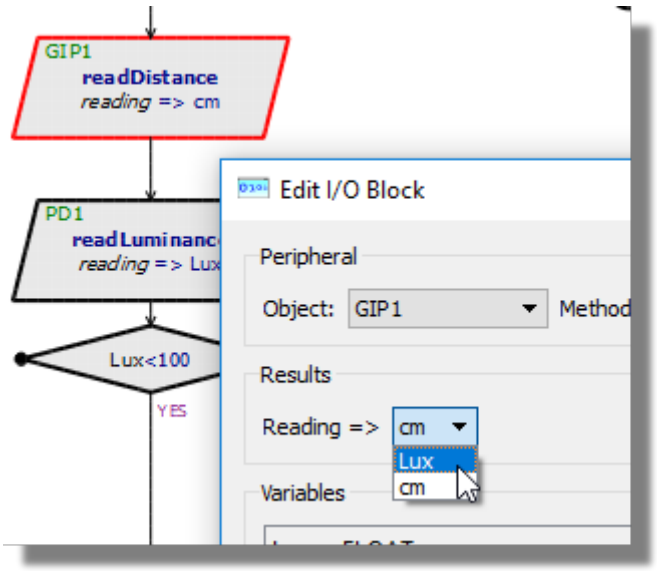
See Also:

http://wiki.seeedstudio.com/Grove-Luminance_Sensor/

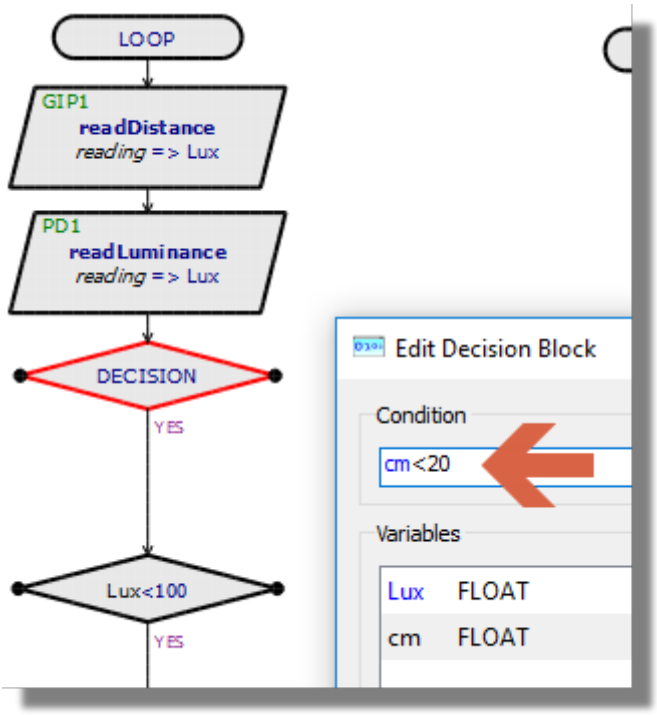
For the proximity sensor, we want to read a distance so drag and drop into the top of the loop.



Set the Variable that it's reading to be LUX



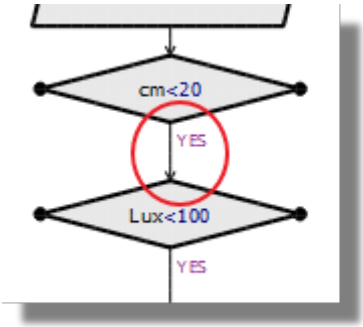
The proximity sensor can detect up to 80cm but we'll set our initial test to 20cm. Add another Decision block above the LUX Decision block, and set to be cm<20:



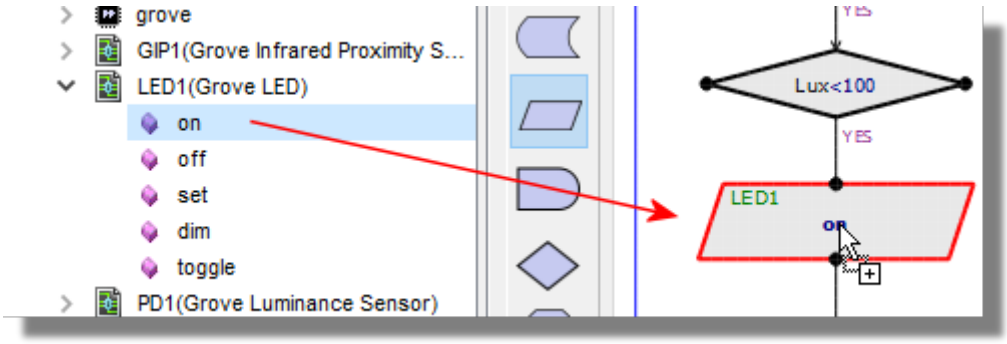
See Also:

- http://wiki.seeedstudio.com/Grove-80cm_Infrared_Proximity_Sensor/

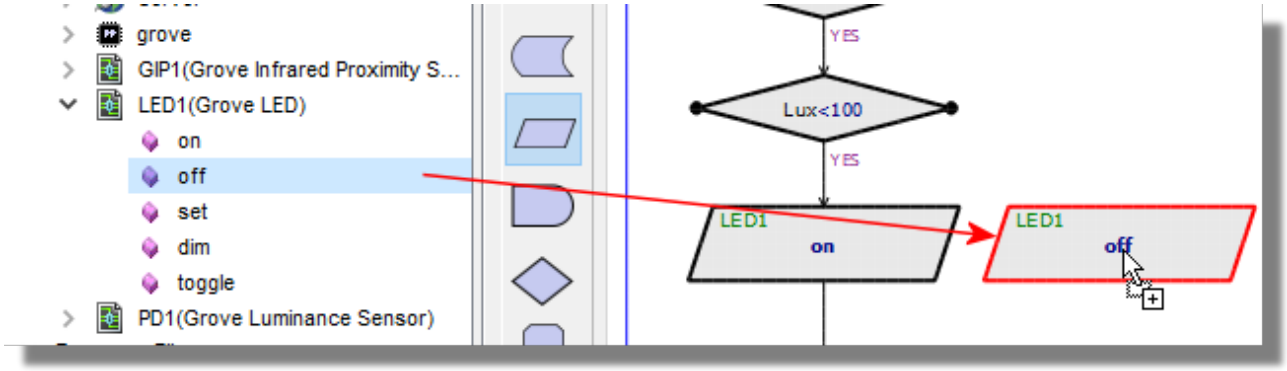
Since, we need both conditions to be true for the LED to turn on, make sure that the second decision is on the YES branch of the first decision.



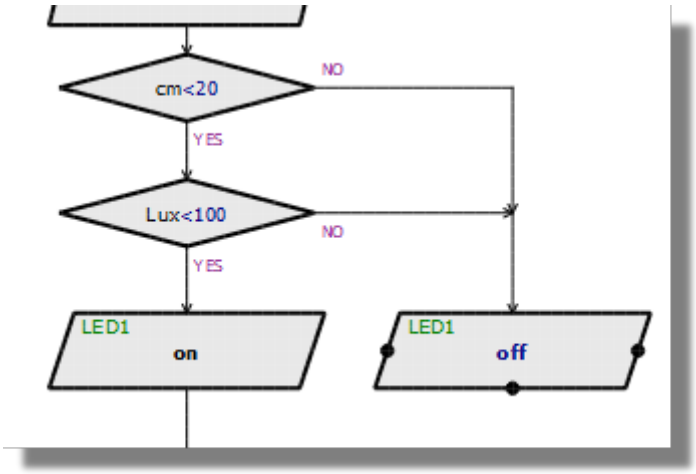
Next, drag and drop the led on method at the bottom of the YES branch.



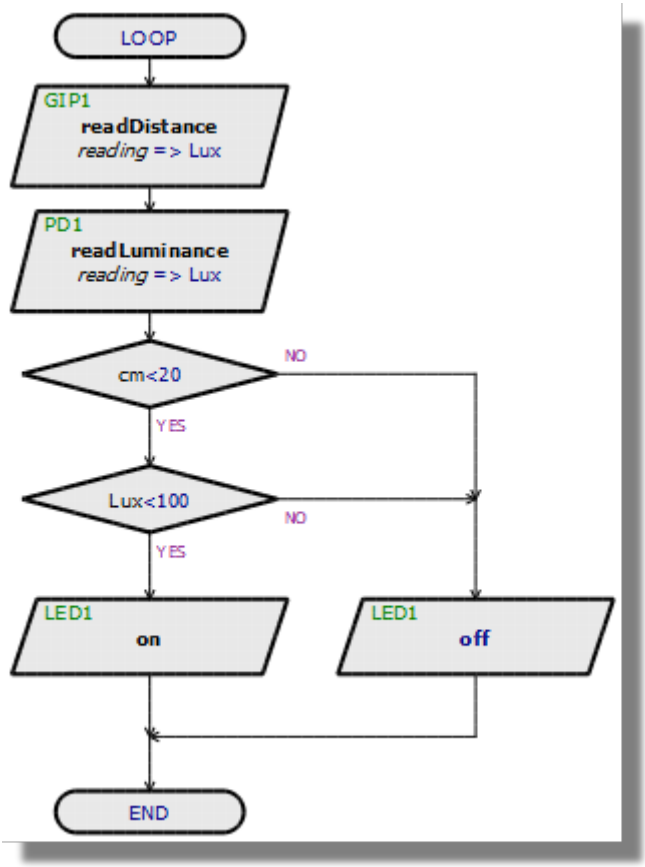
Finally, place the LED off method to the right of the main loop beside the LED on method.



Since we want the LED to turn off if either condition is false we can wire both NO branches from the decisions to this method.



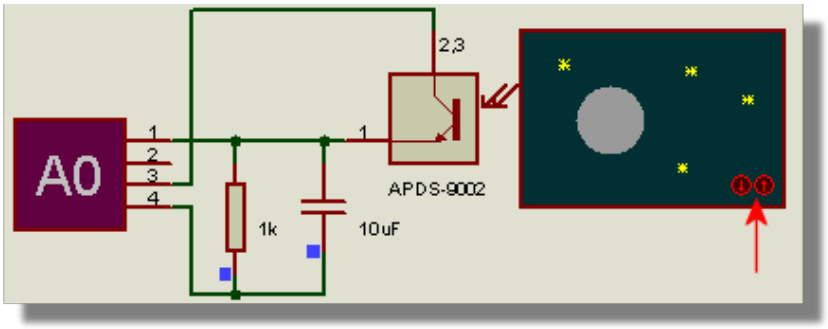
Finally, connect the bottom of the NO program branch back into the main program loop. Your final program should look like the following:

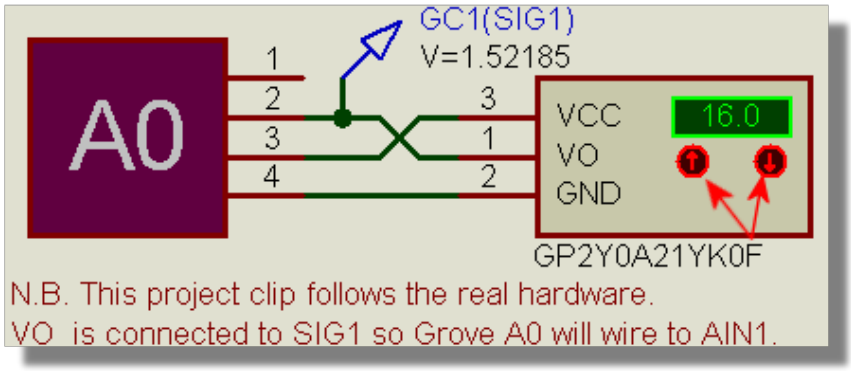


Simulation and Testing

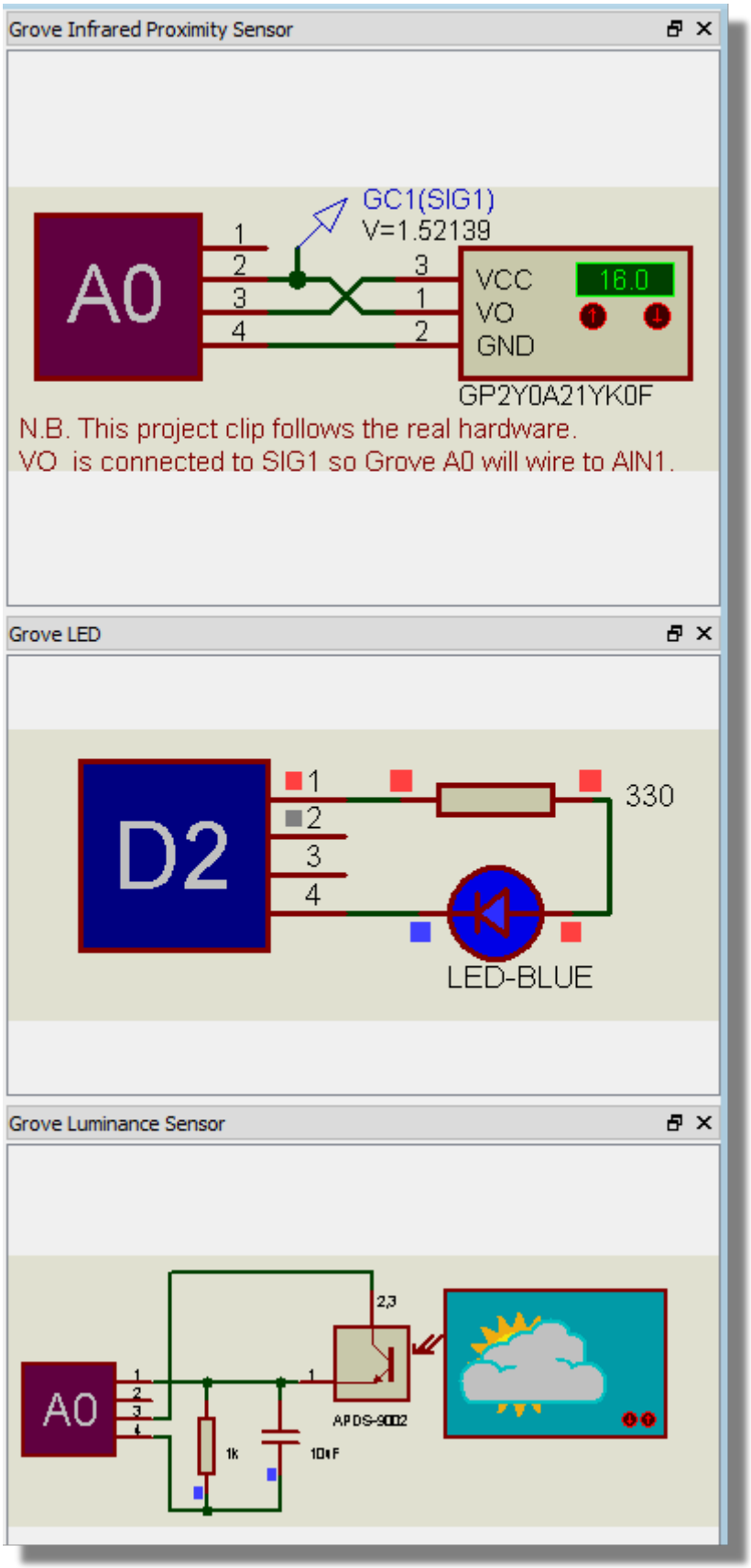
To test our program, all we need to do is press the play button and then adjust the distance display on the proximity sensor and the lux display on the luminance sensor.



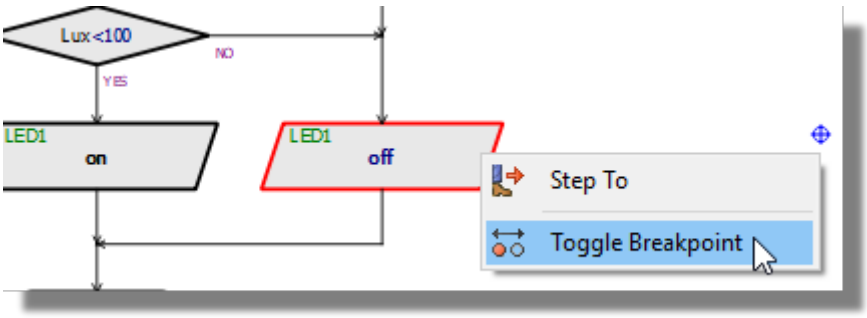




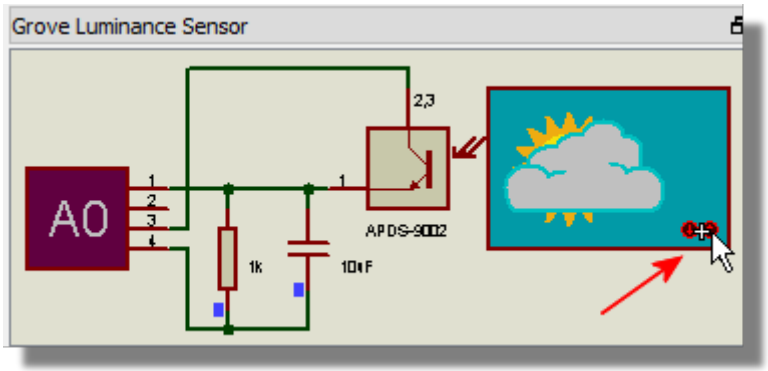
The LED should turn on when the distance value is less than 20 and it's a little cloudy.



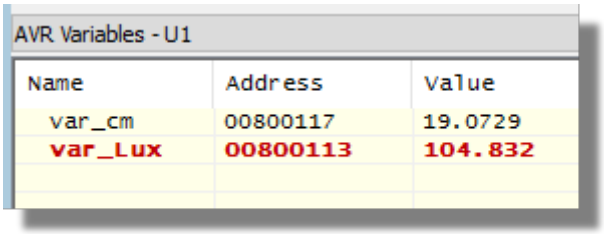
Pause the simulation and set a breakpoint on the LED off method.



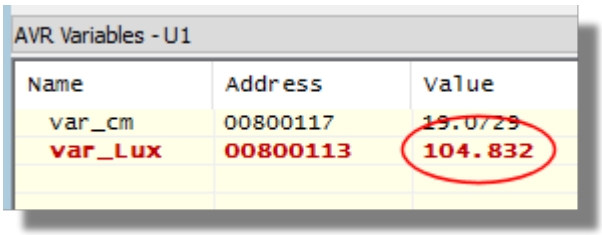
Press play again to run the simulation and change either the distance or the light until the breakpoint triggers

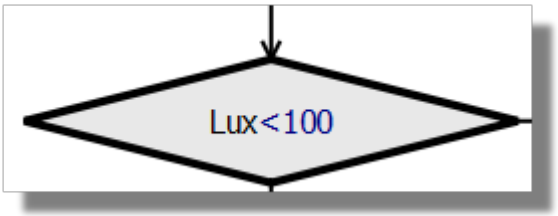


You'll notice that both our variables are available in the variables display.

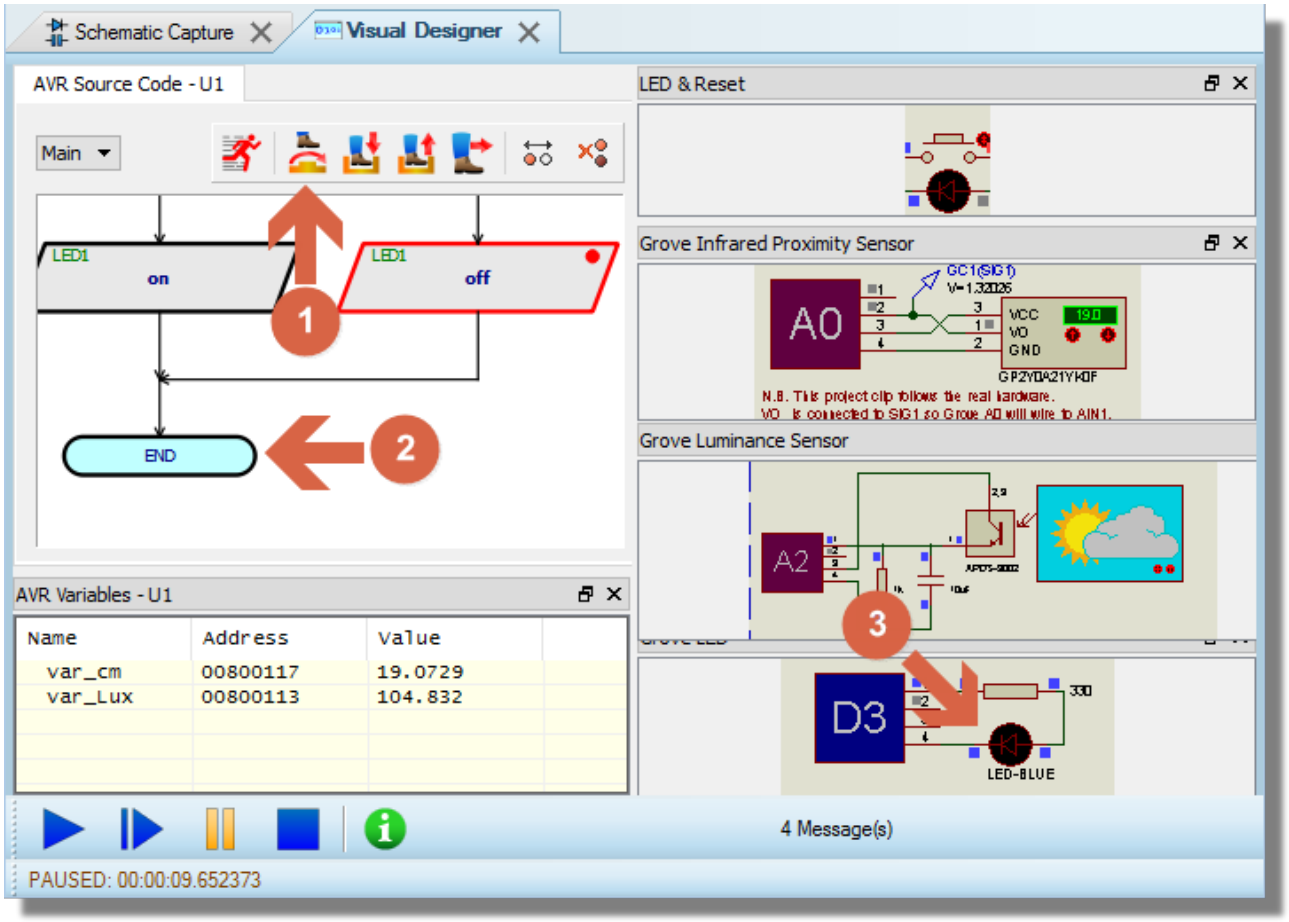


Comparing these values to the tested decisions will tell us which condition has failed.





As you single step via the icons at the top you'll notice the LED turn off.



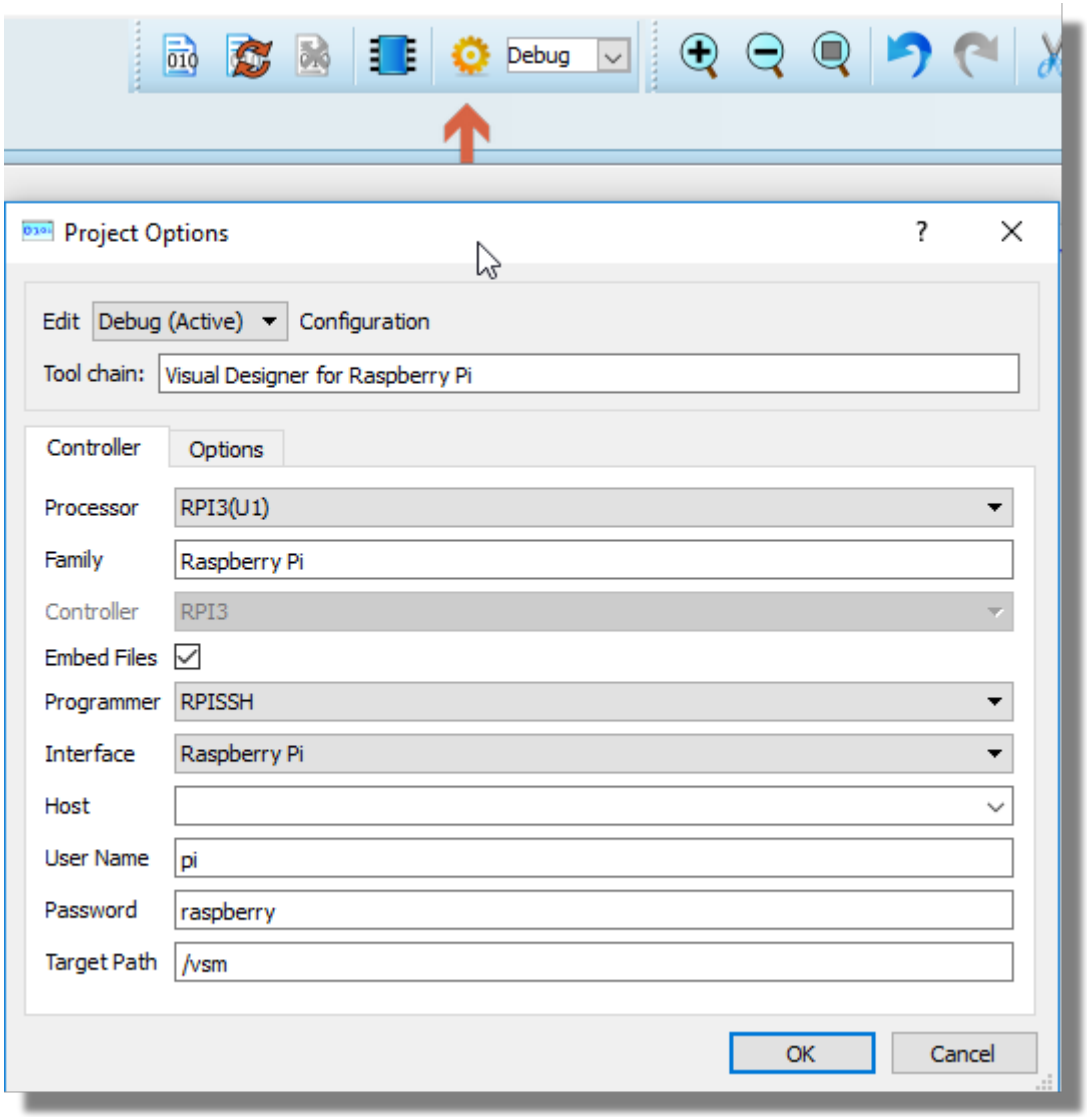
Programming

> ⚠ Before programming the Raspberry Pi you need to first perform the one time confguration as discussed here.

Since everything seems to working in good order, the remaining task is to plug in the Raspberry Pi and program the real hardware. The process here is:

1) Plug the Grove sensors into the Grove baseboard in the same connectors as on the schematic. See also the golden rule discussed earlier.

2) Connect the Grove hat to the Raspberry Pi and connect the Raspberry Pi to the same WIFI as the computer. Now within Proteus within the project options change the Programmer from NONE to RPISSH and the raspberry pi should appear.



3) Click the program upload button in Visual Designer.



4) The program will now run on the Hardware.

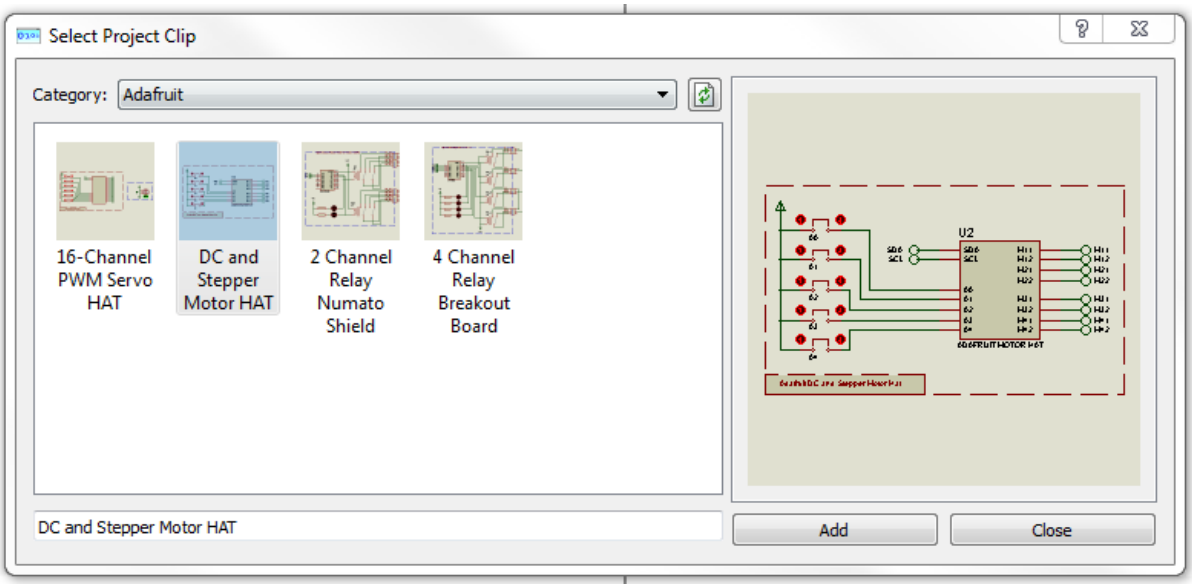
Tutorial 3 : Motor Control

Introduction

This tutorial shows how you can easily control both DC motors and Stepper motors with Visual Designer. We'll do this using the Adafruit DC and Stepper motor hat for Raspberry Pi.

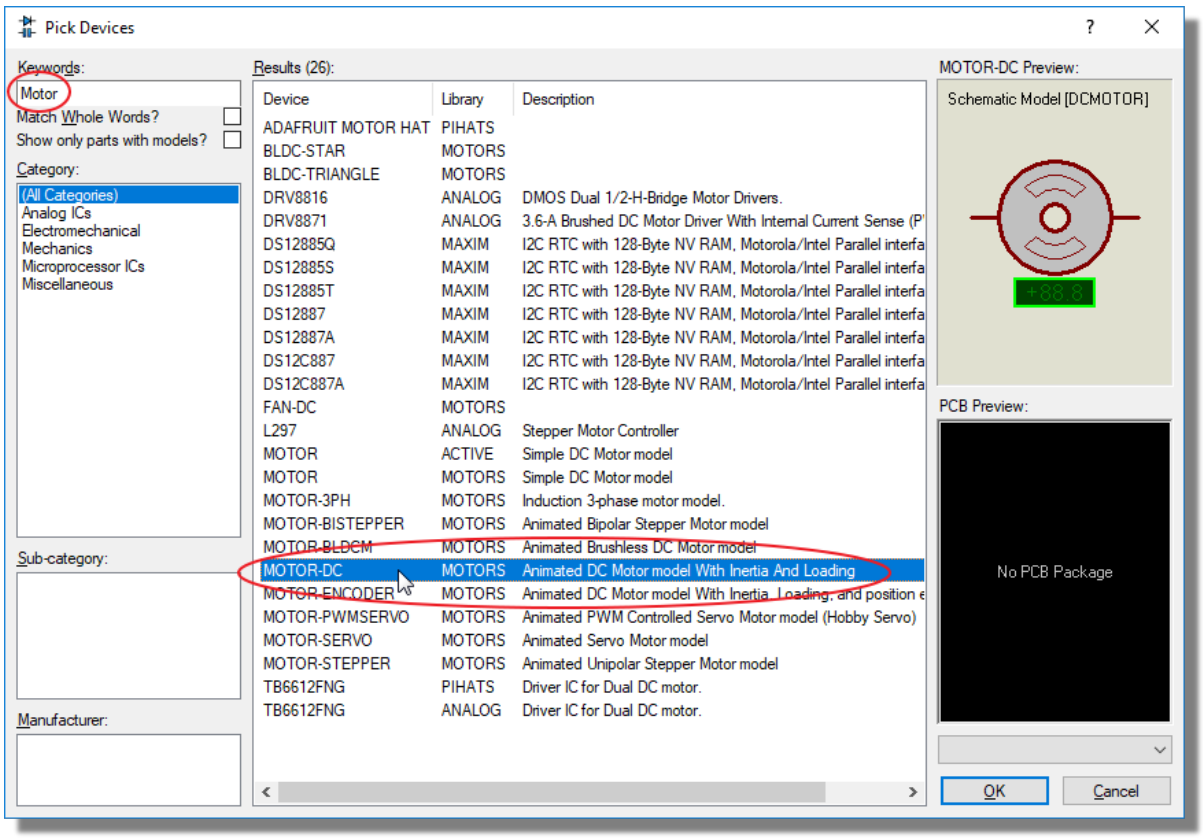
Setup

Start by creating a new Visual Designer Project and then Add the Raspberry Pi Hat with DC and Stepper Motors from the peripheral menu.



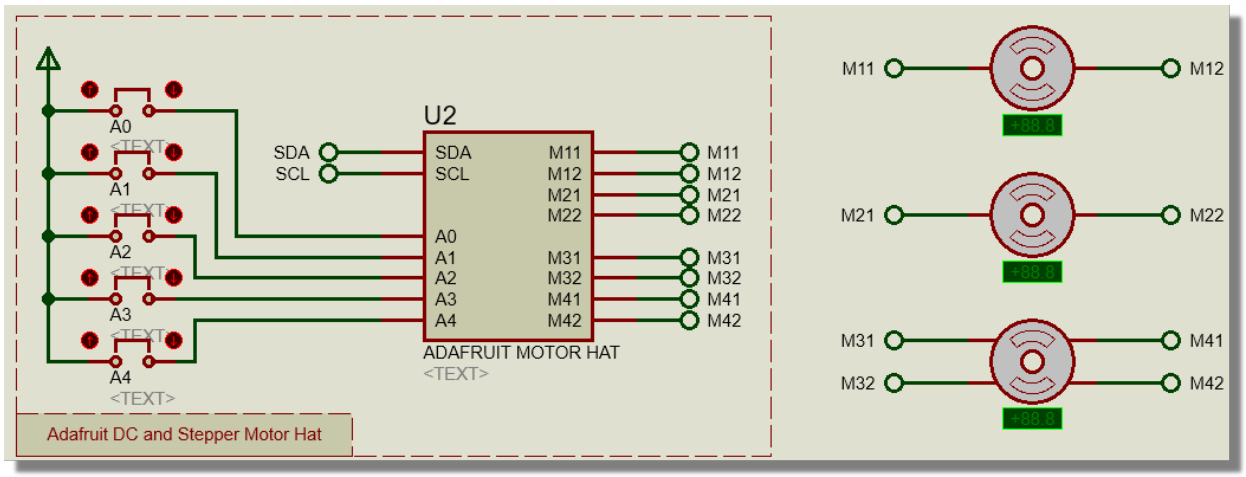
Picking the Adafruit Hat from the Add Peripherals command in Visual Designer

Unlike the Arduino Hat the Raspberry Pi Hat lets you connect the motors yourself. This means you need to first find the motors via the schematic pick devices form (MOTOR-DC & MOTOR-BISTEPPER) and double click to insert them into the project.



Using the Pick Devices Dialogue on the Schematic to add the motors to your design.

Then connect the part to the hat using the terminals The usual rules apply in that terminals with the same name are deemed connected together

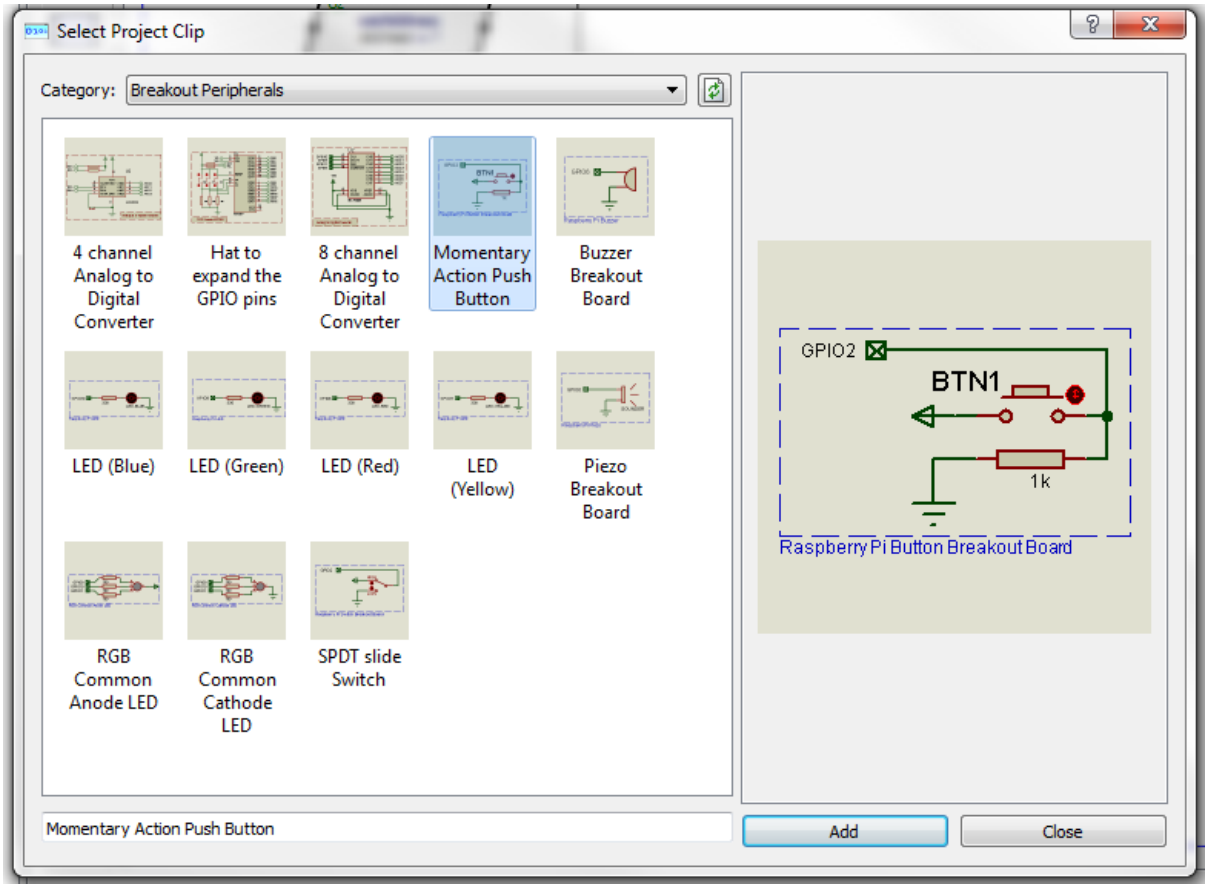


Here you can see we have connected one stepper motor and two DC motors with connections made by terminals.

Programming

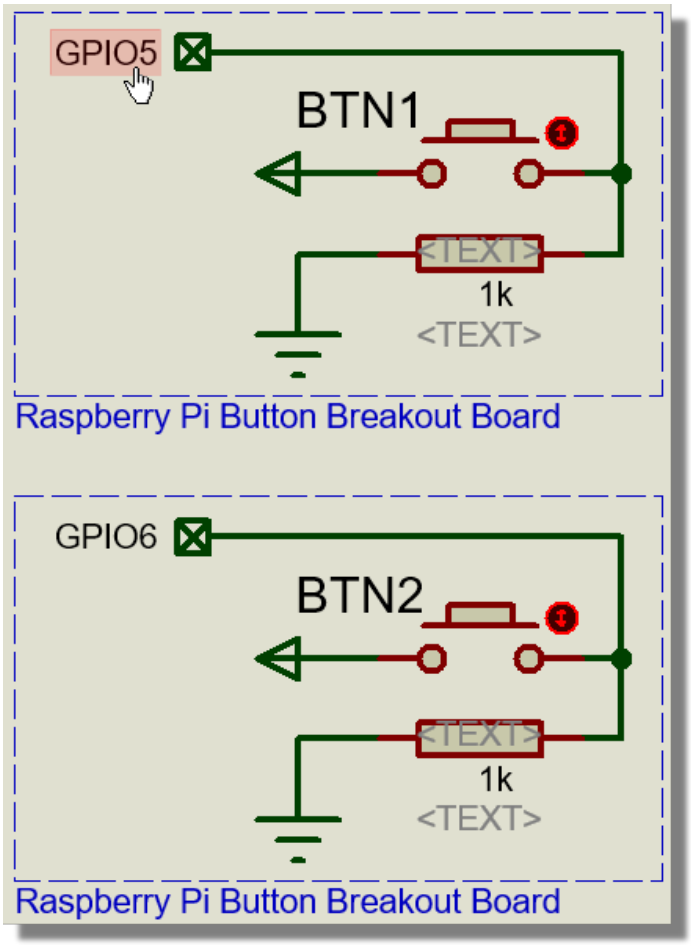
For this project we will have 2 buttons, one will start a dc motor and the other will stop that motor. After we have this project working we will then simulate it and work with the debugging tools.

Firstly, you will need 3 components, 2 buttons and a motor. Add the two buttons using the add peripheral method.



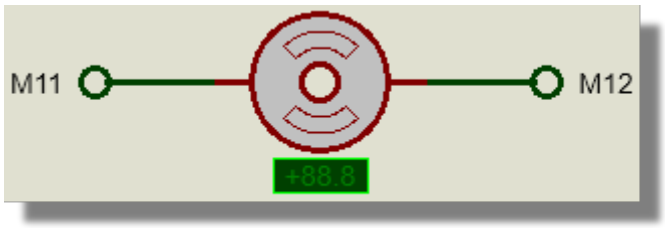
Add Button Peripheral.

Make sure you change the GPIO pins so that they are not both using the same pin



Make sure the buttons are assigned to unique GPIO Lines

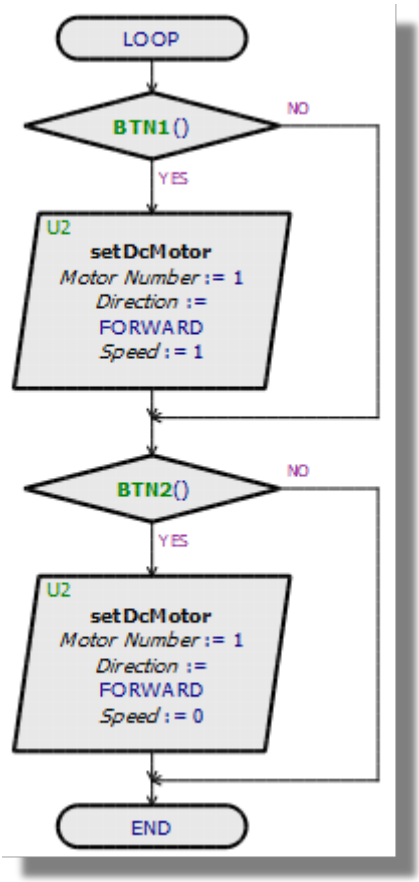
You should already have at least one DC motor. If not then make pick one from the libraries and make sure it is connected to motor 1 (M11,M12) of the hat. It should look something like this:



Correctly connected DC Motor on Motor 1 of the Hat.

> Note that if you want the motor to appear in Visual Designer during simulation you will need to draw an Active Popup box around the motor.

Now you can drag the code blocks on. We want to change the speed of motor 1 each time a button is pressed. Button 1 will start the motor and button 2 will stop the motor. You won't need any code in setup. Your finished program should look like the following



Motor Control flowchart. Uses the sensor functions for the buttons to make decisions.

Testing and Debugging

Now that the project is ready to go we can test it out in simulation. The remainder of this tutorial is essentially a detailed look at simulation and stepping commands to make sure that you understand how to control a running simulation and how you might debug problems with your code.

The simulation is controlled from the animation control panel at the bottom left of the Proteus application



Here you will find the play, step, pause and stop buttons. The play button will start the project or continue if the project is paused.

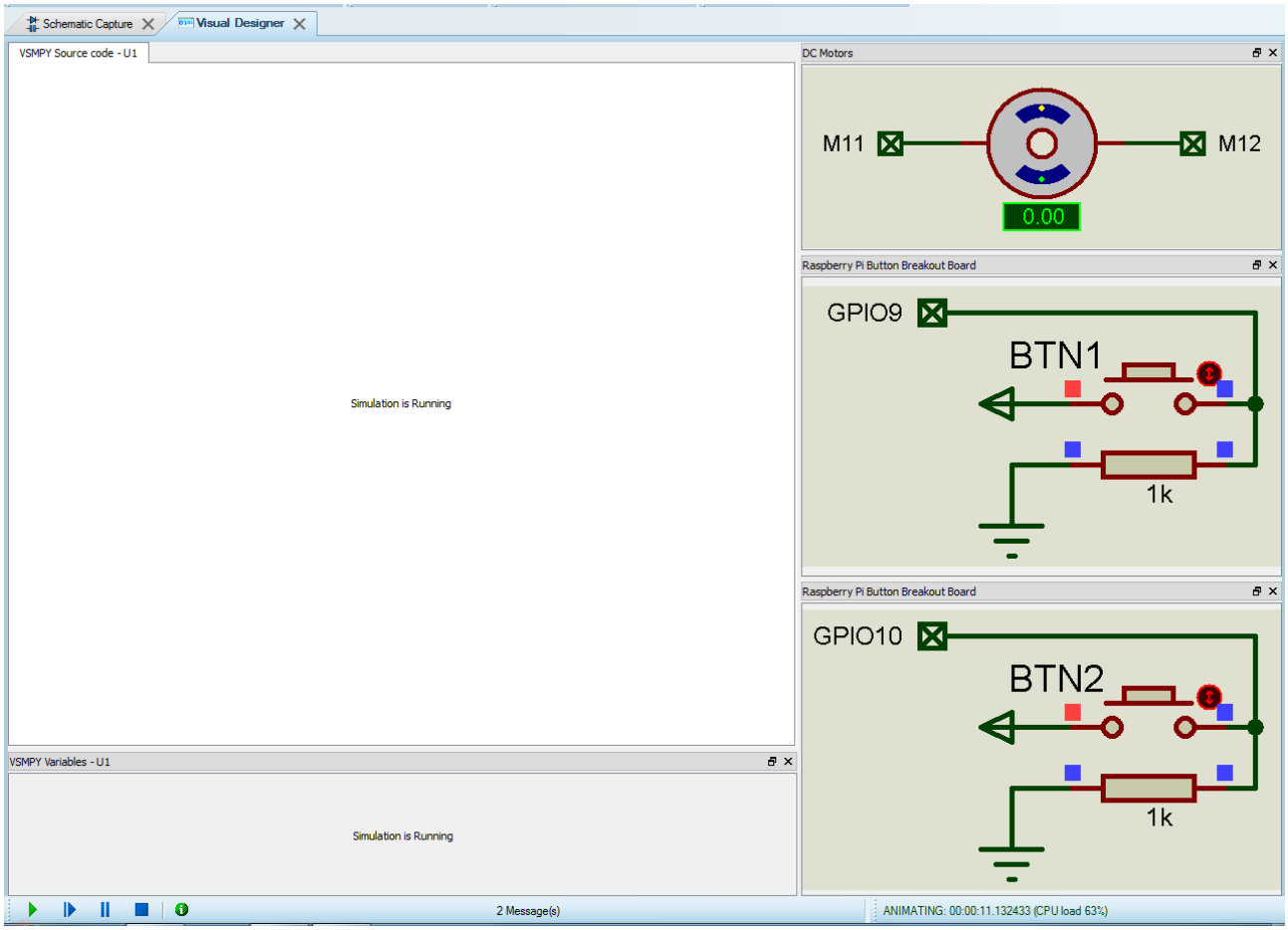
The step button will allow you to slowly step through the project pausing after every block of code.

The pause button will pause time allowing you to look at variables, voltages and other elements. The pause button will not stop the project and you can restart it with the play button.

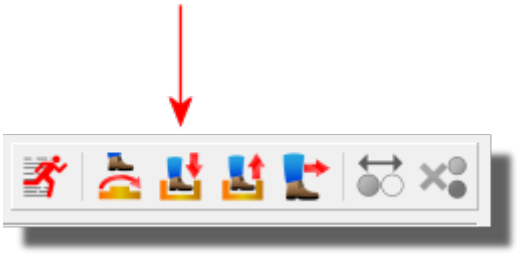
The stop button will stop the whole project and allow you to edit and modify it.

Basic Single Stepping

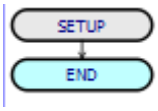
We will start by simply pressing run, the program will start, and you will enter simulation mode. Press button one to start the motor going. Let the motor gain some speed then press button 2 to stop power to the motor. The motor will now slow down. Once you have tested this and confirmed that everything is working properly you can stop the program using the stop button.



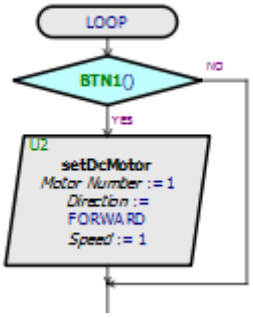
Next, press the step button once to start then pause the program. At this stage the program code is currently sat in the setup block. Next, pres the Step Into button to get to the end of the Setup block:



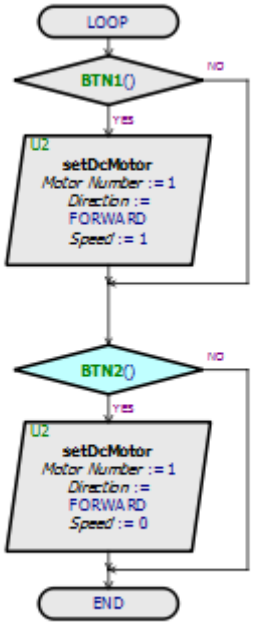
You will now see the End of setup light up. This shows that the setup function has ended.



Pressing the Step Into button again will advance the program to the first decision block of the loop routine.

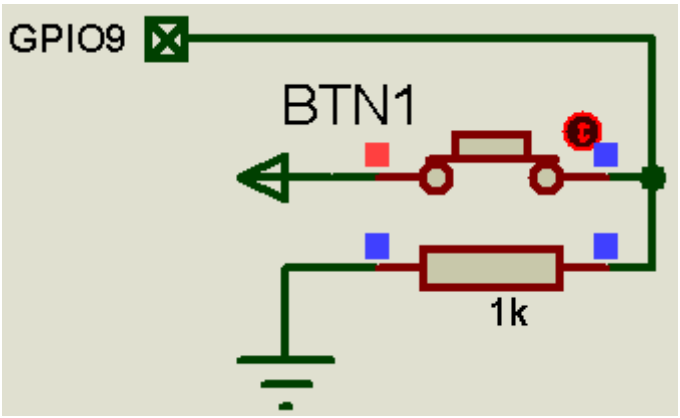


If button 1 is pressed the code will follow the 'Yes' line, if its not pressed it will follow the 'No' line. In our case the button is not pressed so when we keep stepping the code the program will move through to the next decision block. We'll then continue running around this loop routine until a button is pressed.



Finally, note that you can lock a button in place by pressing the red vertical arrows in the side of it. This way it will be held down even when you take your mouse away



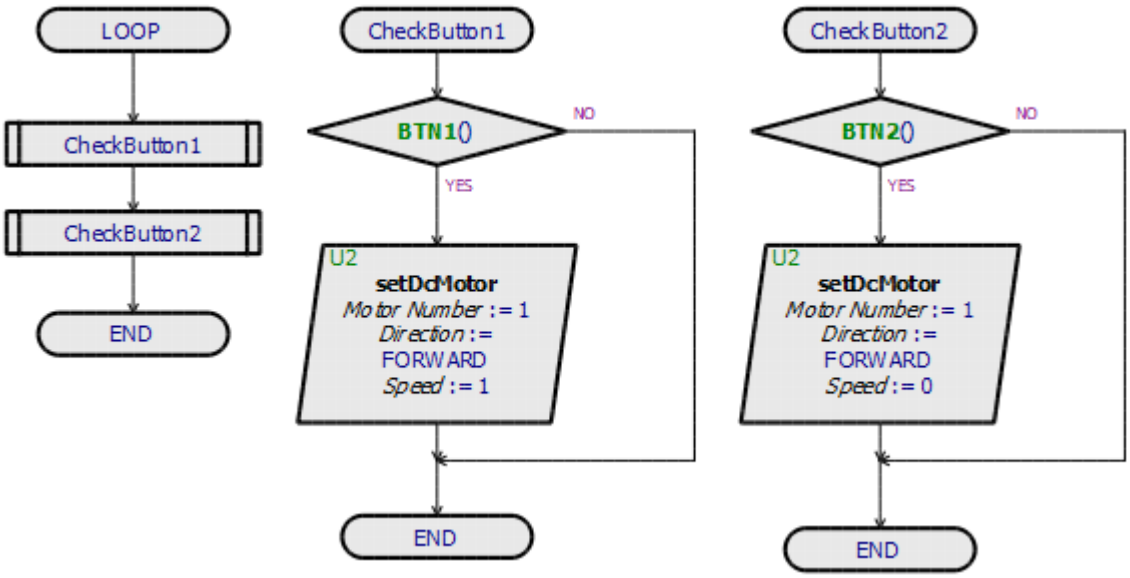


Stepping and Routines

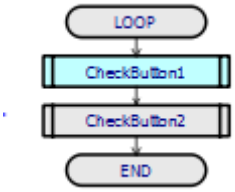
Let's look now at how the step into and step out debugging commands work when we have sub-routines in our program. These basically navigate us in and out of the program stack.



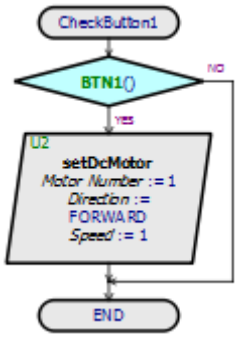
To help explain, we have modified our program to include a couple of sub-routine calls.



In this new program you can step through however it will only highlight the event call and not enter the event block. It will jump between CheckButton1, CheckButton2 and End without entering the two events. If you do want to enter an sub-routine wait until the event is highlighted.



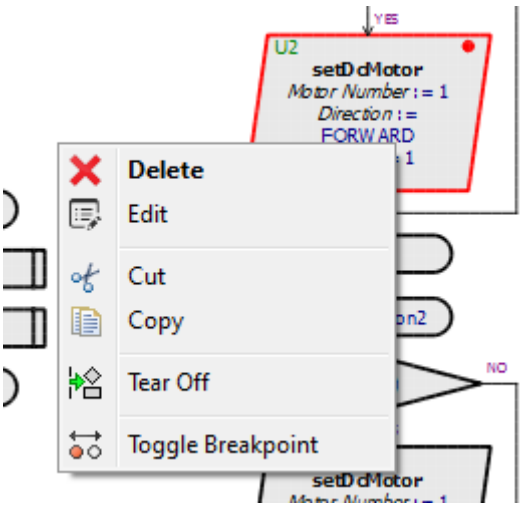
Then press step into to enter the routine and highlight the first code block in the routine.



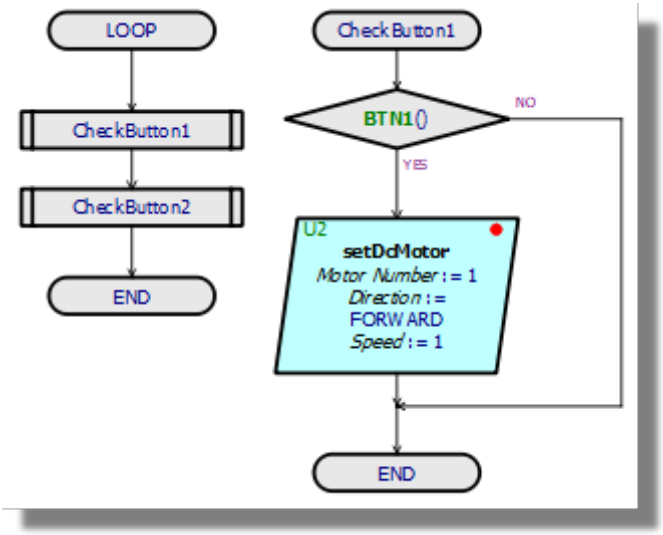
We could continue to step through the routine with our normal step commands or use the step out button to exit the routine and pause at the next code block following the return from the routine.

Breakpoints

Breakpoints allow you to stop the code when it reaches a specific block of code. To toggle breakpoints, enter Visual designer, right click on the block you wish to place the breakpoint and toggle breakpoint.



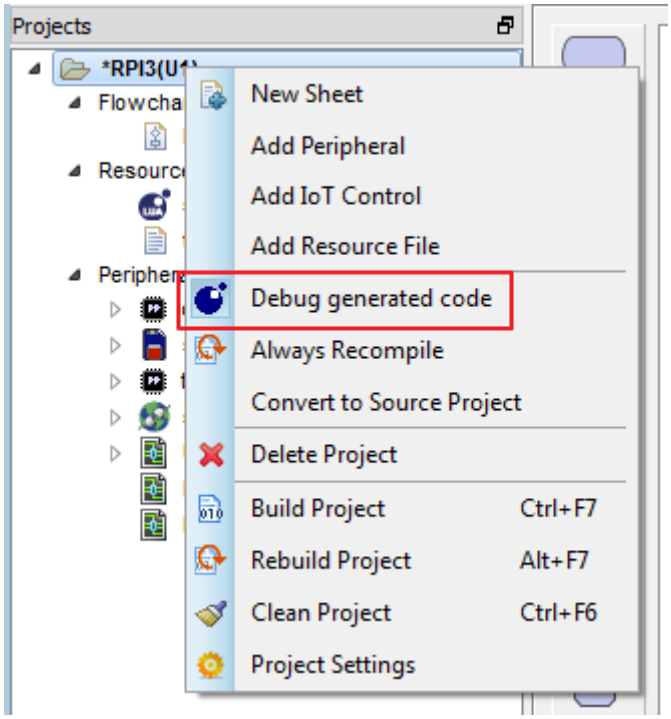
Now run the program, assuming button 1 is up nothing will happen. The code will run normally until you press button 1. Once button 1 has been pressed the code will pause on that block.



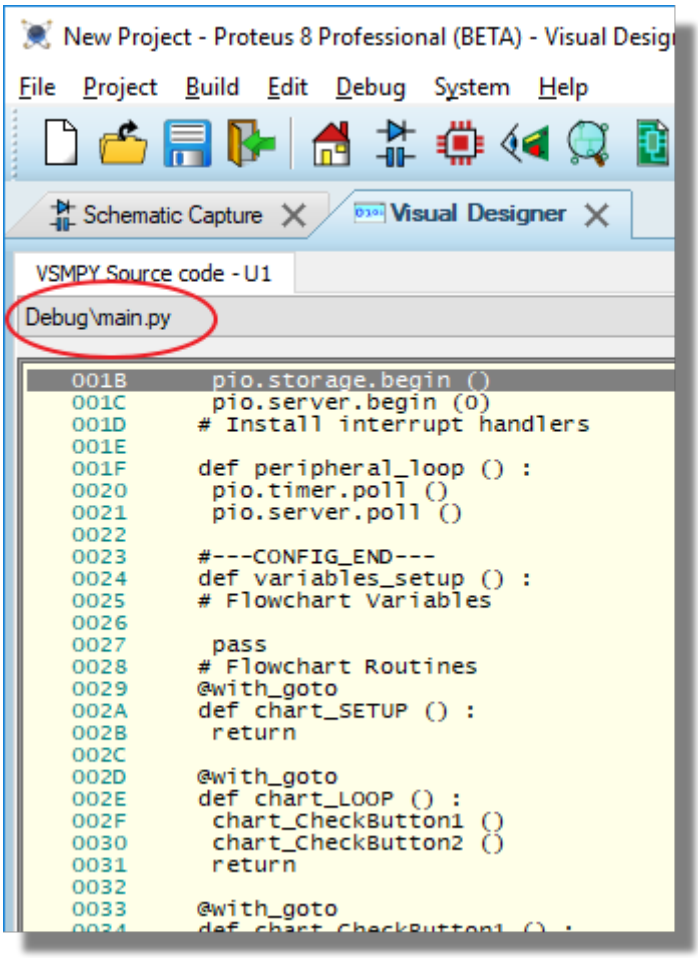
From here you can step through the code to debug further.

Debugging Generated Code

If you want to look deeper into what is happening in the code you can look directly at the code and not the blocks. Right click on the project menu and select debug generated code.



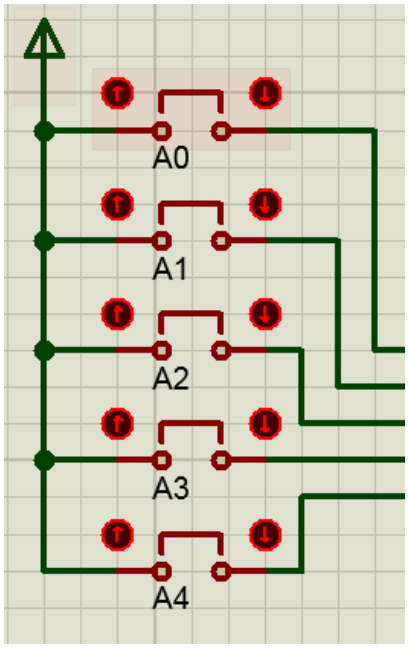
Now when you run the project the flowchart won't be present. Instead there is a blank screen. By pressing the step button once it will load up the python script. Now every time you step it will step line by line opposed to block by block.



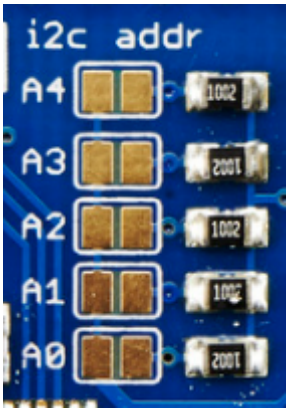
You should now be able to navigate and follow the code using the same debugging and stepping tools we discussed earlier. To navigate through different routines, use the dropdown box above the code.

Note about Addresses

Some hats are stackable such as this motor hat. When using multiple hats together you may need to change the I2C addresses. You can do this by using the jumpers. The Hat will have a base address, for the motor hat the base address is 0x60. The Jumpers work in binary so by connecting A0 the address will go to 0X61, and connecting A1 the address will go to 0x62 and so on. The motor hat has a possibility of 32 different addresses including address 0x60.





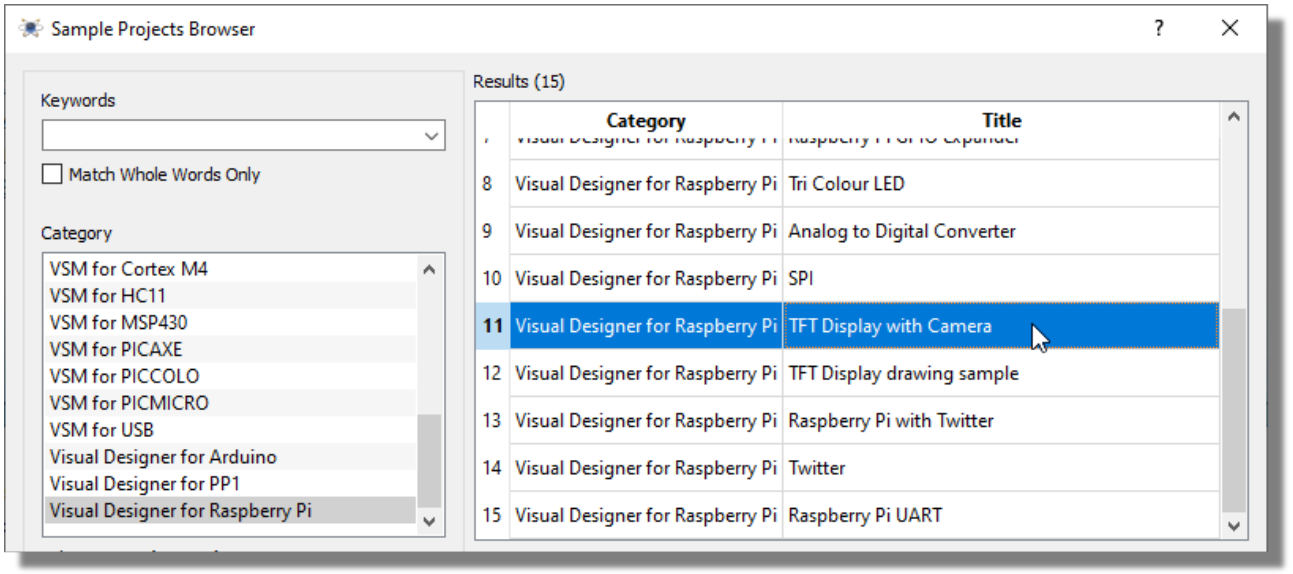


Tutorial 4 : Using the Camera

This tutorial is written with the understanding that you have previously followed the Visual Designer tutorials 1 - 3. If you have not we would highly recommend that you do so first so you have an understanding of your way around Visual Designer and the Terminology that is used below.

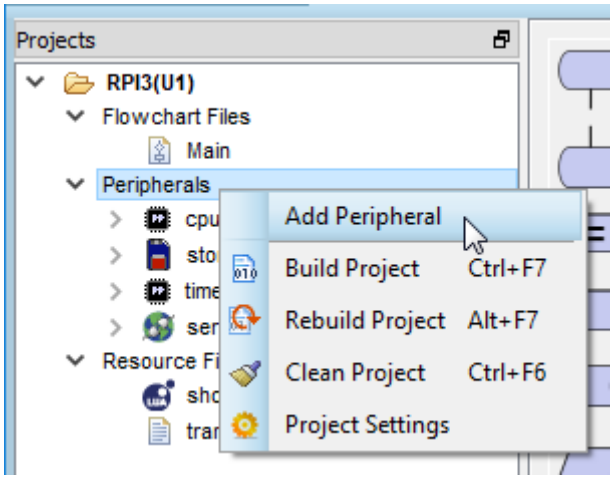
In this tutorial we will show you how to take a photo from your Web camera and show it on a TFT Display. Please insure that your camera device is connected and working before starting.

It may also be beneficiary to take a look at the Raspberry Pi TFT Sample project.

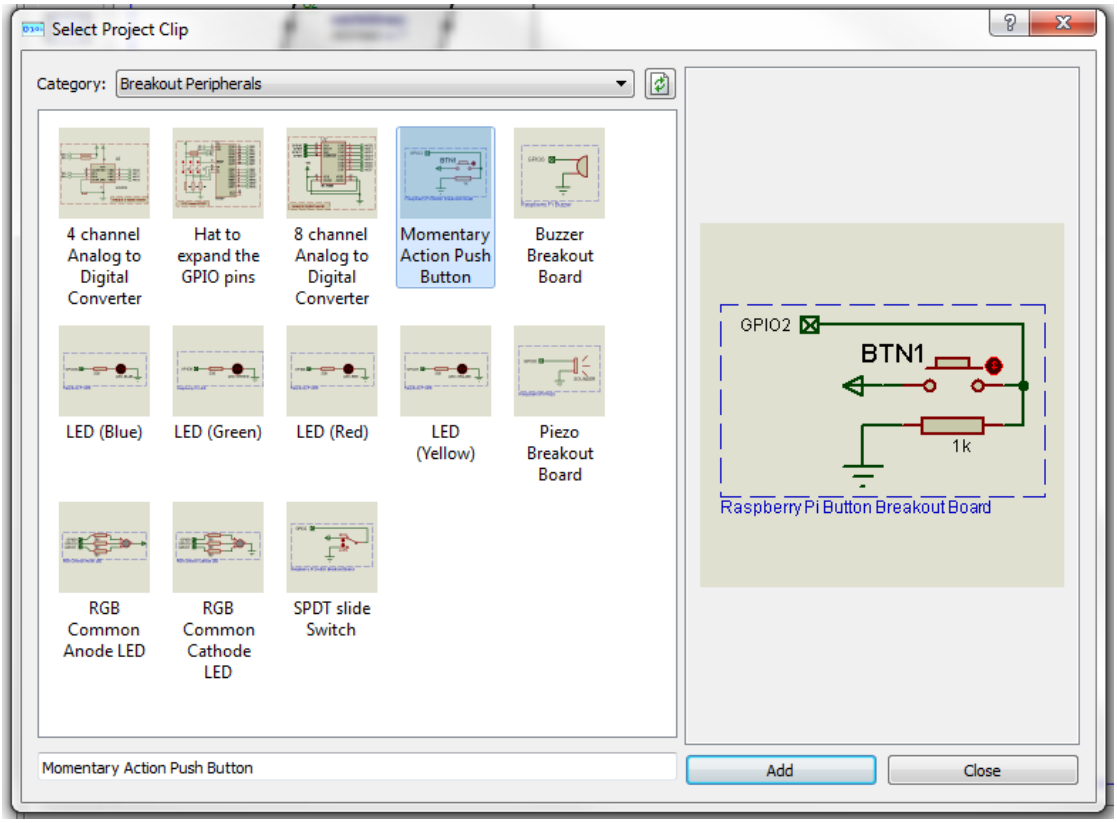


Adding the Peripherals

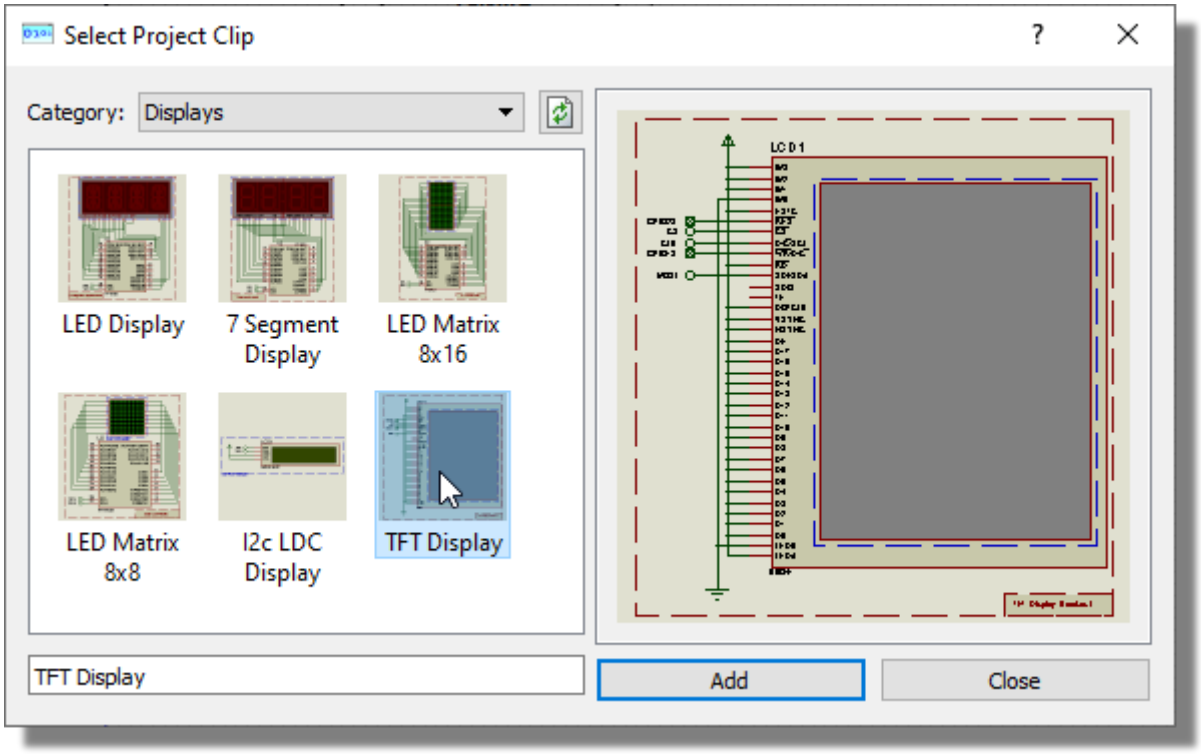
We need to add a TFT display to show the image, a Button to take the picture and lastly not least, the Camera peripheral. Right click on the Peripherals list on the left hand side and select Add peripherals:



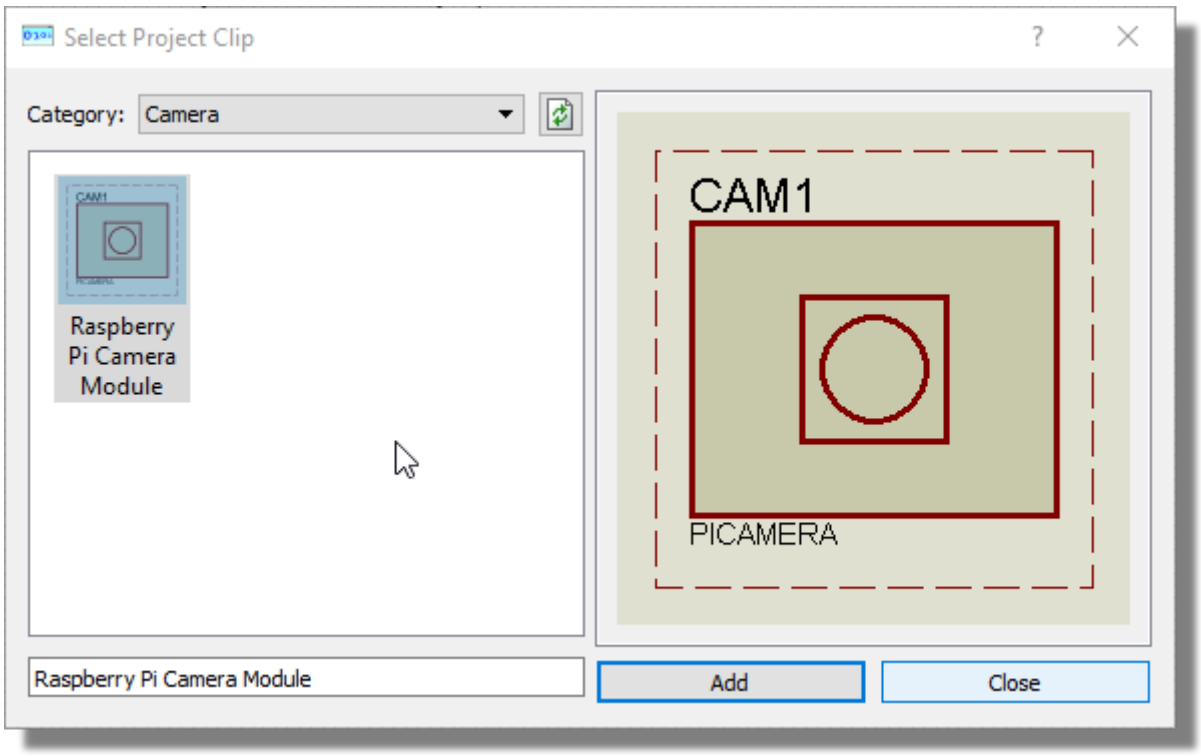
The Button is as in previous tutorials Breakout Boards > Momentary Action Push Button.



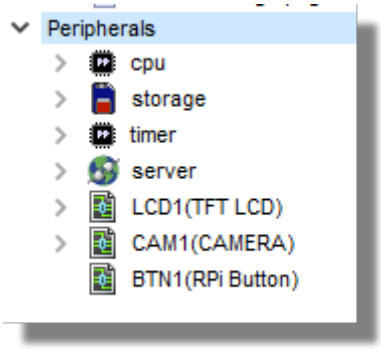
The TFT Display is found under the Displays drop down list in the Add Peripherals Dialog form:



The Camera is found under Camera list.

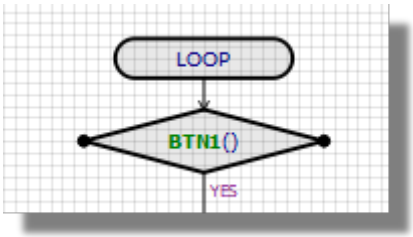


These will now appear in the list under Peripherals like so:

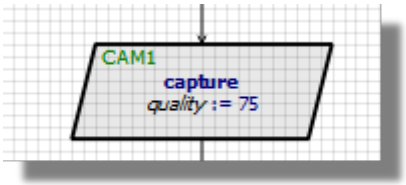


Writing the Program

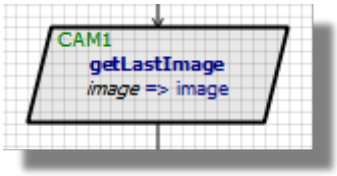
All Methods and Peripherals are be added to the LOOP Routine. There are four things that we want to do: 1. Push the button, 2. capture the image, 3. get the image and 4. display the image. Start by dragging the Button Peripheral in to the LOOP.



Next we need to capture the image. Under the Camera peripheral, drag the Capture Method in to the LOOP routine below the BTN1() peripheral.

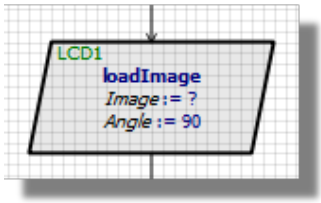


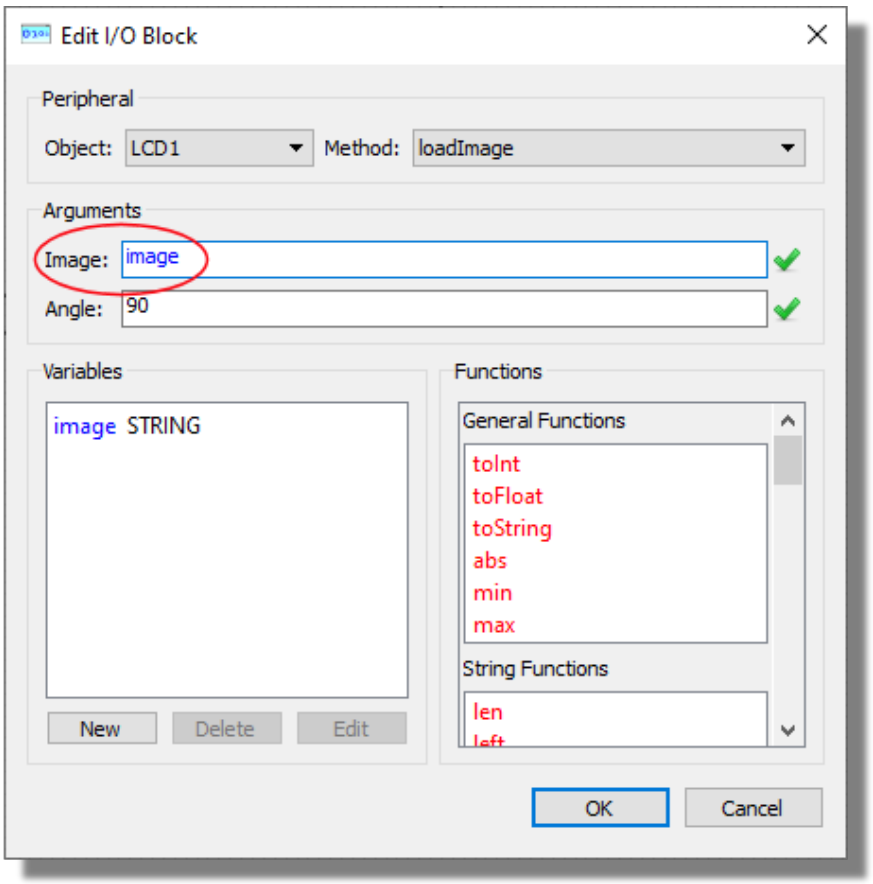
Now we need to 'get the image', and as we would like to get the image that was last taken, we need to add the *getLastImage* method. Drag this below the Capture method.



Note that this also adds a variable for the image name. In this case the default name is 'image' - physical image => image name.

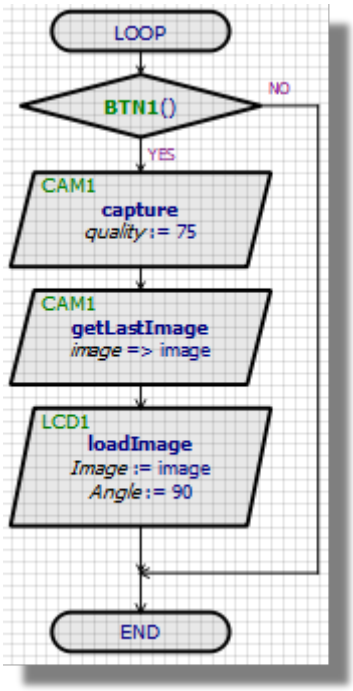
All that's left to do is load the image in to the TFT display. This is simply done by dragging the TFT 'loadImage' method from under the LCD1(TFT LCD) peripheral below the getLastImage method. You will need to edit the block and insert the image name. In this case as the default is *image* we type that in as seen below:



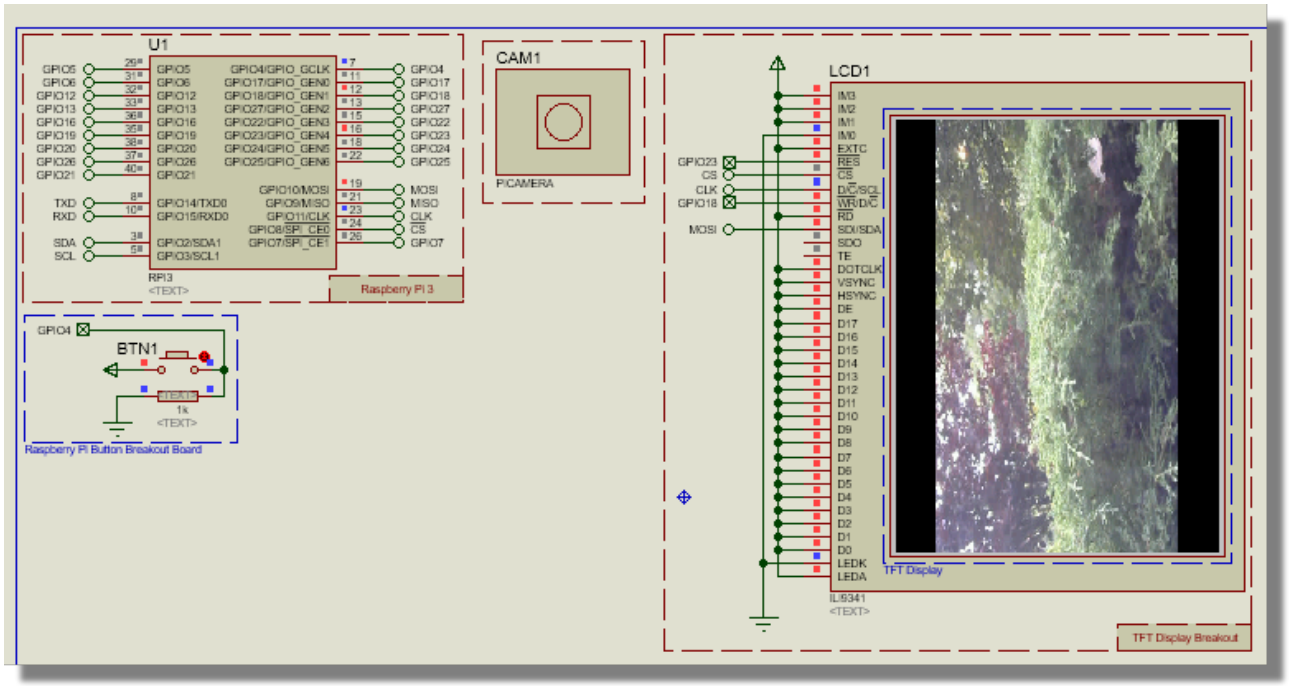


Draw a line from the right hand side of the BTN1 block to above the END block so that if the button is NOT pressed the loop will continue.

You project should now look something like this:



Press the Play button and take some Photos!



Tutorial 5 : Using Twitter

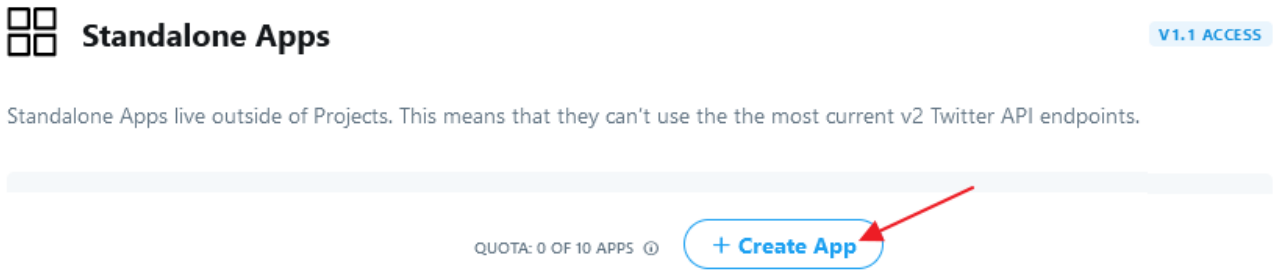
This tutorial is written with the understanding that you have previously followed the Visual Designer tutorials 1 - 4. If you have not we would highly recommend that you do so first so you have an understanding of your way around Visual Designer and the Terminology that is used below.

You will need a Twitter Developers account for this exercise. You can request one here: https://developer.twitter.com/en/apply-for-access.html Please be aware that it may take a few days before Twitter grant you your Developer account! (Ours was a matter of minutes...).

⚠ Please note that all Twitter Keys in this document are not real and have been randomly generated for illustration purposes only!

Creating an App in on Twitter

Make a new app. This should provide the required 4 keys. Click on Overview and then Create App. You do not need to make a Project.



Make a name for your App (My_Proteus_App) and fill in any other required forms. Most of the entries can just be anything you can think of, it's just to get you through the form to get the App created!

Once you have created the App, you will be given some 'keys':

API Key

AgRLLVMqqDbXrtdyHEdYnmuE2

API Secret Key

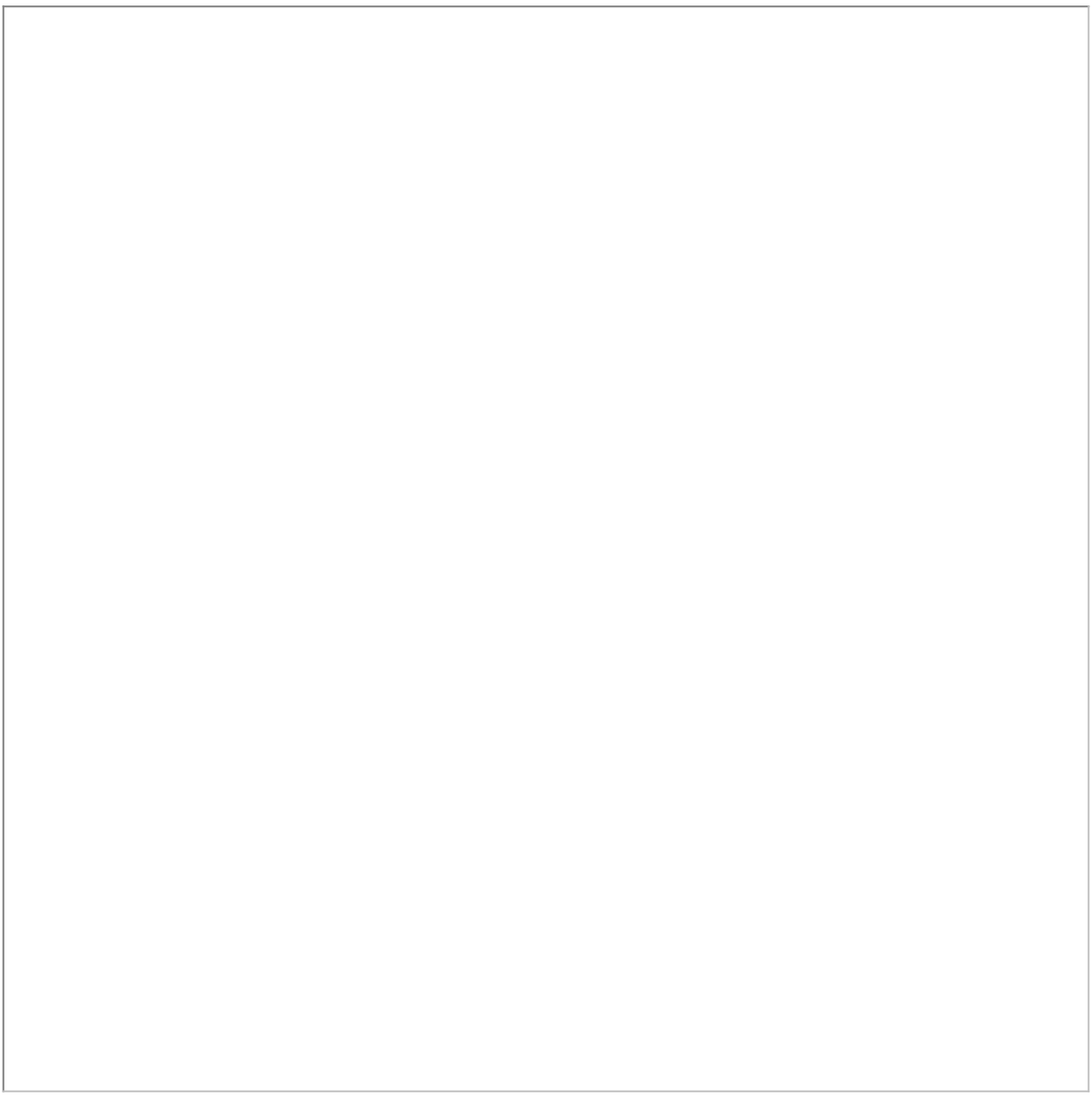
Wt9x5m2ReOoStAuXZ3Zcyjia8OE4Yy7JKCxNynBHaY9xsdGAeF

Bearer Token

AAAAAAAAAAAAAAAAAAAAAOnsRQEAAAAA%2FImcV7R7QmfkVN2CW1TldPT5izY%3D
7bSCH8c9M2Uw5meHdaOEYhLdBKiyPdeiNg3py2gcwlDriJPLWc

Copy the API Key and API Secret keys to a new file / somewhere safe.

Next we need to get 2 Access Tokens. Click on Get user Tokens:

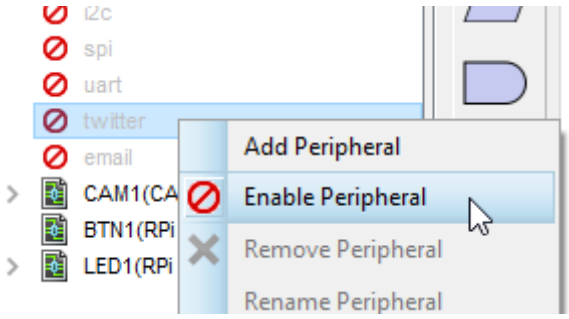


Next to Access Token and Secret, click on Generate:

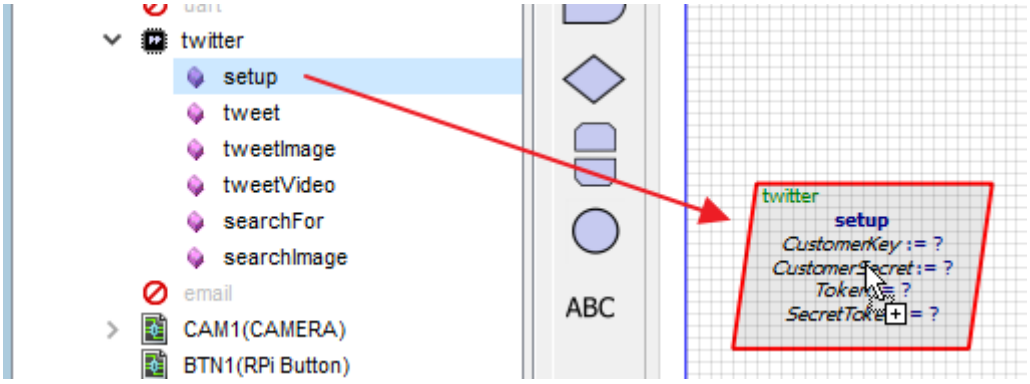
Now copy the two keys 'Access Token' & 'Access Token Secret' to the same place you saved the other API keys / safe location.

Setting up the Program

Having obtained the keys, you will need to write your program to tell Proteus what and how to tweet. Firstly you will need to enable the Twitter Peripheral on the left hand side:



Next, Under the Setup routine, drag a Setup object under the Twitter peripheral on to the sheet:



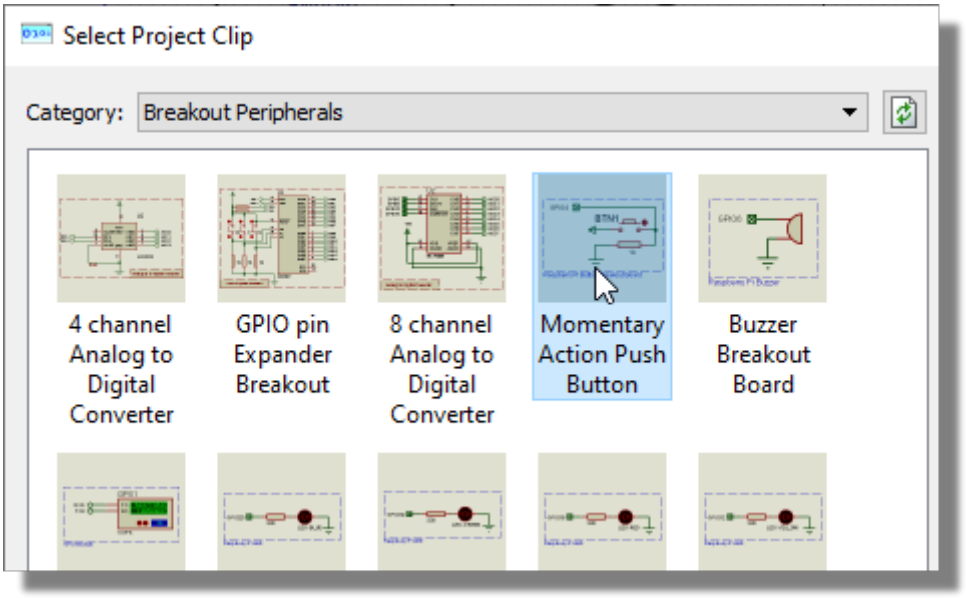
Now enter in the required keys from your Developers Twitter account as shown above.

- Customer Key = API Key.
- Customer Secret = API Secret Key.
- Token = Access Token.
- Secret Token = Secret Token.

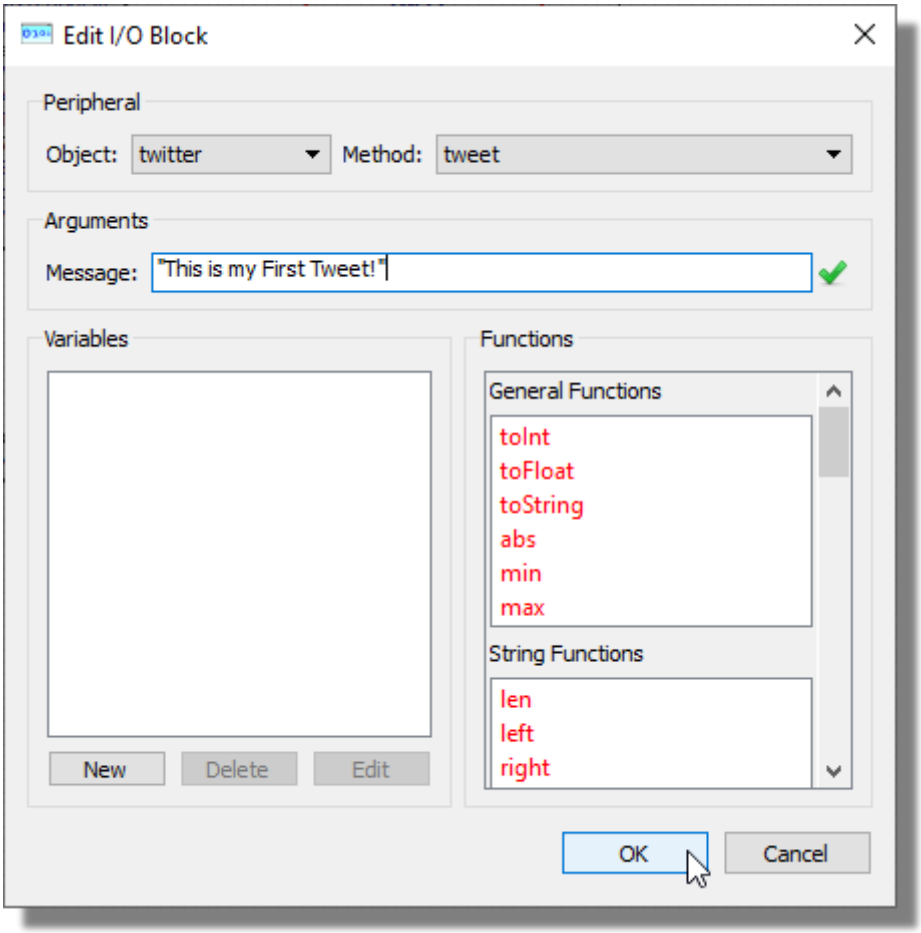
Now we nee to draw the rest of the flowchart. Under the LOOP routine.

What we want to do is send a simple Tweet when a button is pressed.

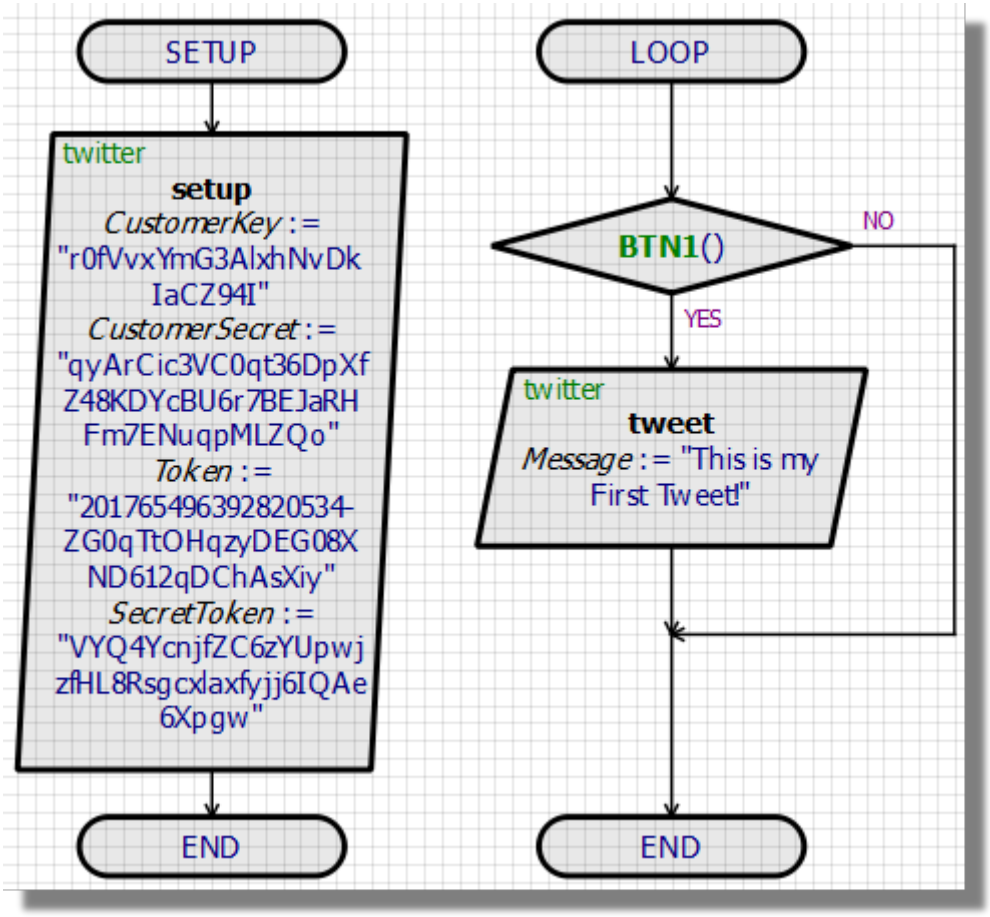
Add a Button from the Peripherals Library:



Add a Tweet Method. Edit the Tweet Method and enter in your message.



Your final flow chart should look something like this:



(Not real keys)

Compile and press the button. Then login to your Twitter account and you will see your message. It will be displayed as if you have sent it directly from twitter yourself.

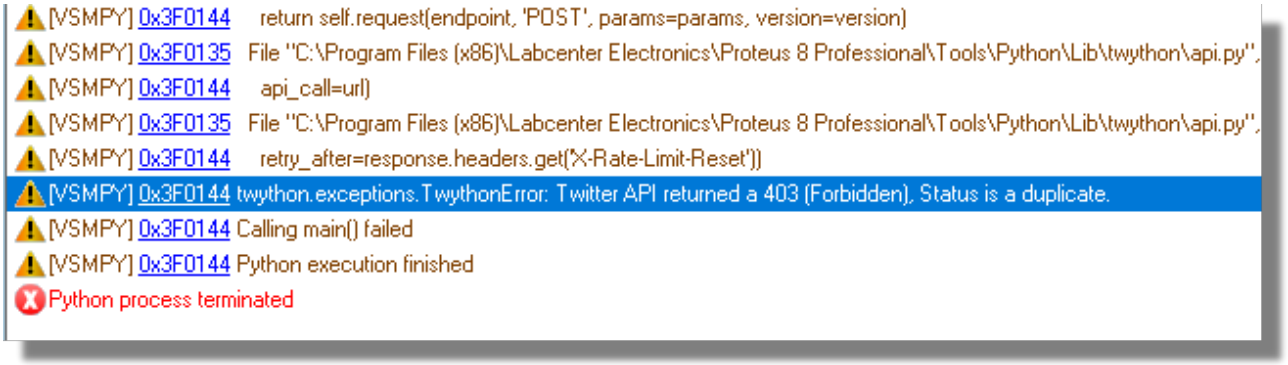
Sending an image

> This is for advanced users that will need to have followed the other tutorials first.

While sending a tweet is great, Proteus can also send images using the camera peripheral as well as search for tweets. Please take a look at our sample design by typing in 'Twitter' in the keywords field, or looking under Category 'Visual Designer for Raspberry Pi' and selecting the sample entitled Twitter.

Troubleshooting

If you get an error message that looks like:



It will be because you have tried to send the same tweet twice, and twitter does not like this.