Vbb4Arduino

‘Two Arduino’s’

Prototype Virtually, Make for Real
<table>
<thead>
<tr>
<th>Date</th>
<th>Description</th>
<th>VBB Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>22 Jan 2017</td>
<td><em>Initial Mashup and modernise</em></td>
<td></td>
</tr>
<tr>
<td>23 Jan 2017</td>
<td><em>Added initial entries for Arduino Emu, Arduino SIM, S2812, DS1307, 24LC256</em></td>
<td></td>
</tr>
<tr>
<td>24 Jan 2017</td>
<td><em>Added initial entries for Trimmer, POT, relay, solenoid, 555, 7 Segment</em></td>
<td>5.53a</td>
</tr>
<tr>
<td>27 Jan 2017</td>
<td><em>Updated Arduino SIM monitor</em></td>
<td></td>
</tr>
</tbody>
</table>
1 Contents
2 Introduction .................................................................................................................. 10
  2.1 Features ................................................................................................................. 10
  2.2 Limitations ............................................................................................................ 10
  2.3 Two Arduino’s the Arduino Java/Cpp Dual personality ......................................... 10
  2.4 Comparison with VBB ........................................................................................... 11
    2.4.1 Upgrade ............................................................................................................ 12
    2.4.2 Site License ....................................................................................................... 12
  2.5 Virtual Breadboard Circuit Modelling ................................................................... 12
3 Download and Installation .......................................................................................... 12
  3.1.1 Download Locations ......................................................................................... 12
  3.1.2 Installation VBB4Arduino .................................................................................. 13
  3.2 Buying VBB4Arduino ............................................................................................. 13
    3.2.1 How to buy ....................................................................................................... 13
    3.2.2 1. Purchase a license from PayPal ................................................................... 13
           3.2.3 2. Find the invoiceld ................................................................................. 13
  3.3 3. Download the Software ...................................................................................... 14
  3.4 4. Enter the activation code in the splash screen ................................................... 14
  3.5 TroubleShooting .................................................................................................... 15
  3.6 License Management .............................................................................................. 16
    3.6.1 Firewalls ........................................................................................................... 16
  3.7 Supporting Hardware .............................................................................................. 16
  3.8 Customised Shields and Boards[future] ................................................................. 16
4 VBB Application .......................................................................................................... 17
  4.1 Solutions ................................................................................................................ 17
  4.1 Projects Files ........................................................................................................ 17
  4.2 Design Sheets ........................................................................................................ 17
    4.2.1 Breadboard ..................................................................................................... 17
    4.2.2 Java Source Projects ...................................................................................... 17
    4.2.3 Sketch Source Projects .................................................................................. 17
    4.2.4 Logic Analyser ............................................................................................... 18
4.3 Project Select.............................................................................................................. 18
  4.3.1 TIPS .......................................................................................................................... 18
  4.3.2 New Tab .................................................................................................................... 19
  4.3.3 New Project Template .............................................................................................. 19
  4.3.4 New Arduino Emulator Project ............................................................................... 20
  4.3.5 New Arduino Simulator Project .............................................................................. 21
  4.3.6 Import Arduino Sketch Wizard ................................................................................ 22
  4.3.7 ‘Built-in’ template projects ...................................................................................... 25
4.4 Existing Tab .................................................................................................................. 27
4.5 Recent .......................................................................................................................... 29
4.6 Design Mode .................................................................................................................. 30
  4.6.1 App Window ............................................................................................................. 30
  4.6.2 App Menu ................................................................................................................ 31
  4.6.3 App Toolbar ............................................................................................................ 32
  4.6.4 Solution Explorer .................................................................................................... 33
  4.6.5 View Mode ............................................................................................................... 35
  4.6.6 View Panes .............................................................................................................. 35
  4.6.7 Design Sheets ......................................................................................................... 36
  4.6.8 DesignSheet Toolbox .............................................................................................. 36
4.7 The properties panel ..................................................................................................... 37
  4.7.1 The property description panel .............................................................................. 37
  4.7.2 Empty panel ............................................................................................................ 37
  4.7.3 Status Toolbar ........................................................................................................ 37
4.8 Runtime Virtualization ................................................................................................. 38
5 Design Sheets .................................................................................................................. 38
  5.1 Arduino Sketch Code Project ..................................................................................... 40
    5.1.1 Adding a Sketch Project to a VBB Project .............................................................. 40
    5.1.2 Adding Source to Sketch Project ........................................................................... 40
    5.1.3 Cloud Compiler : CodeBender .............................................................................. 41
    5.1.4 Custom Libraries .................................................................................................. 41
    5.1.5 Verifying Source .................................................................................................. 42
5.1.6  Supported Arduino Sketch Libraries ........................................ 43

5.2  Java Source Code Project Design Sheet ........................................ 44

5.2.1  Adding a Java Project to a VBB Project ................................... 44

5.2.2  Adding Source to Java Project ............................................... 44

5.3  Solution Tree Manager ............................................................. 46

5.4  Activating Java Source Project ................................................... 46

5.4.1  Context Sensitive Functions ................................................... 47

5.4.2  Delete .................................................................................. 47

5.4.3  Rename ................................................................................ 47

5.4.4  Cloud Compiler : Java Compiler ............................................. 48

5.4.5  Custom Libraries ................................................................... 48

5.5  Design Time Source Editor ......................................................... 50

5.5.1  Keyword Highlighting ............................................................. 51

5.5.2  Syntax Error Highlighting ....................................................... 51

5.5.3  Tab Suggestions .................................................................... 51

5.5.4  Parameter Suggestions .......................................................... 52

5.5.5  Edit Menu, Cut, Copy and Paste Undo .................................... 52

5.5.6  Code Verification ................................................................. 53

5.5.7  Java Source Console Tab ....................................................... 53

5.6  Logic Analyser .......................................................................... 55

5.6.1  Trace Log ( *.VLG ) ............................................................. 55

5.6.2  Adding a Logic Analyser ....................................................... 55

5.6.3  Drag the Design Sheet into a View ......................................... 55

5.6.4  3. Sampling Control Panel .................................................... 58

5.7  Breadboard DesignSheet ............................................................ 61

5.7.1  Toolbar : Breadboard ............................................................. 61

5.7.2  Note on Zoom factor: ........................................................... 62

5.7.3  Placing a component from the Toolbox .................................. 63

5.7.4  Component Editing ............................................................... 63

5.7.5  Placing a component from the Toolbox .................................. 63

5.7.6  Select a component .............................................................. 63
5.7.7 Select a group of components ......................................................... 64
5.7.8 Append a component to a selection ................................................. 64
5.7.9 Move a group of selected components .............................................. 65
5.7.10 Shrink and grow a selection ......................................................... 65
5.7.11 Wiring Essentials ............................................................................ 66
5.7.12 All Nets (--) .................................................................................... 72
5.7.13 Virtual Links .................................................................................. 72
5.7.14 Show Links .................................................................................... 73

5.8 Toolbox: Breadboard ........................................................................ 76

6 Breadboard Components ........................................................................ 77
6.1 Function Block .................................................................................. 92
6.1.1 The Function Block expression ....................................................... 93
6.2 Function Block Expression ................................................................. 94
6.3 Function Block Expression Grammar ............................................... 94
6.3.1 Function Block Editor Dialog ......................................................... 97
6.4 Function Block Usage ....................................................................... 97
6.4.1 Dynamic Voltage Divider ............................................................... 98
6.4.2 Multi Tap Voltage Divider .............................................................. 100

6.5 Generic Layout Components ............................................................. 77
6.5.1 Generic Layout Only ..................................................................... 77

6.6 Generic DIP ..................................................................................... 78

6.7 Parametric ....................................................................................... 79
6.7.1 Using the Library Component ......................................................... 80

6.8 Marking up a generic component with a function block .................... 83

6.9 Radial Component ............................................................................ 86
6.9.1 Parametric Model ........................................................................... 86
6.9.2 Properties ....................................................................................... 87

6.10 Axial Component ............................................................................. 87
6.10.1 Parametric Model .......................................................................... 87
10.3.2 Properties ................................................................. 127
10.4 Usage ........................................................................... 127
10.5 JUMPN ......................................................................... 128
  10.5.1 Pinout ..................................................................... 129
  10.5.2 Properties .............................................................. 129
  10.5.3 Usage ..................................................................... 129
10.6 NumericKeyPad ............................................................ 130
  10.6.1 Pinout ..................................................................... 130
  10.6.2 Properties .............................................................. 130
  10.6.3 Usage ..................................................................... 131
10.7 Seg7 ............................................................................. 132
  10.7.1 Pinout ..................................................................... 132
  10.7.2 Properties .............................................................. 133
  10.7.3 Usage ..................................................................... 133
10.8 DigitalPort .................................................................... 136
  10.8.1 Pinout ..................................................................... 136
10.9 Properties ...................................................................... 136
  10.9.1 Usage ..................................................................... 136
10.10 PushButton ................................................................. 137
  10.10.1 Pinout ..................................................................... 138
  10.10.2 Properties .............................................................. 138
10.11 Usage .......................................................................... 139
10.12 Switch .......................................................................... 140
  10.12.1 Pinout ..................................................................... 140
  10.12.2 Properties .............................................................. 140
  10.12.3 Usage ..................................................................... 141
10.13 PanelMeter ................................................................. 141
  10.13.1 Pinout ..................................................................... 141
  10.13.2 Properties .............................................................. 141
  10.13.3 Usage ..................................................................... 142
10.14 SlidePot ....................................................................... 143
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.14.1 Pinout</td>
<td>143</td>
</tr>
<tr>
<td>10.14.2 Properties</td>
<td>143</td>
</tr>
<tr>
<td>10.14.3 Usage</td>
<td>143</td>
</tr>
<tr>
<td>10.15 RotaryPot</td>
<td>144</td>
</tr>
<tr>
<td>10.15.1 Pinout</td>
<td>144</td>
</tr>
<tr>
<td>10.15.2 Properties</td>
<td>144</td>
</tr>
<tr>
<td>10.16 Usage</td>
<td>144</td>
</tr>
<tr>
<td>10.17 JoyStick</td>
<td>145</td>
</tr>
<tr>
<td>10.17.1 Pinout</td>
<td>145</td>
</tr>
<tr>
<td>10.17.2 Properties</td>
<td>145</td>
</tr>
<tr>
<td>10.17.3 Usage</td>
<td>146</td>
</tr>
<tr>
<td>10.18 MiniTerminal</td>
<td>146</td>
</tr>
<tr>
<td>10.18.1 Pinout</td>
<td>146</td>
</tr>
<tr>
<td>10.18.2 Properties</td>
<td>147</td>
</tr>
<tr>
<td>10.18.3 Usage</td>
<td>147</td>
</tr>
<tr>
<td>10.19 LiquidCrystal</td>
<td>148</td>
</tr>
<tr>
<td>10.19.1 PinOut</td>
<td>Error! Bookmark not defined.</td>
</tr>
<tr>
<td>10.19.2 Properties</td>
<td>Error! Bookmark not defined.</td>
</tr>
<tr>
<td>10.19.3 Usage</td>
<td>149</td>
</tr>
<tr>
<td>10.20 LDR – Light Dependent Resistor</td>
<td>149</td>
</tr>
<tr>
<td>10.20.1 Description</td>
<td>149</td>
</tr>
<tr>
<td>10.20.2 Properties</td>
<td>149</td>
</tr>
<tr>
<td>10.20.3 Usage</td>
<td>150</td>
</tr>
<tr>
<td>11 API ‘Smart’ Components</td>
<td>170</td>
</tr>
<tr>
<td>11.1 WS2812</td>
<td>170</td>
</tr>
<tr>
<td>11.2 API</td>
<td>171</td>
</tr>
<tr>
<td>11.3 DS1307</td>
<td>172</td>
</tr>
<tr>
<td>11.3.1 API</td>
<td>173</td>
</tr>
<tr>
<td>11.4 24LC256 EEPROM</td>
<td>175</td>
</tr>
<tr>
<td>11.4.1 API</td>
<td>175</td>
</tr>
</tbody>
</table>
2 Introduction

VBB is a software platform designing ‘Breadboard’ form-factor electronic circuits and developing the microcontroller firmware that drive them.

Vbb4Arduino is a version of Virtual Breadboard (VBB) dedicated to the Arduino learner. It is an intentionally limited standalone ‘sandbox’ version of VBB ideal for the taking the first steps learning about physical computing.

2.1 Features

- Integrated ‘C’ programming using VBB hosted CodeBender cloud compiler
- Integrated ‘Java’ programming using VBB hosted Java cloud compiler
- Rich set of components for learning most of the Arduino basics
- 77 Built in examples

2.2 Limitations

- No extensibility and no programming of real devices.
- No programming of real Arduino devices
- No analog circuit analysis. VBB is a digital environment with modelling of common analog circuits (All VBB Versions)

2.3 Two Arduino’s the Arduino Java/C++ Dual personality

Arduino is a curious mix of Java and C++. The Arduino ‘Sketch’ is derived from Processing which is a java development environment. Hence the Arduino Sketch looks at first glance like java and for many starter examples it is java. However even though a Sketch looks like java under the hood it is actually C++ ‘hacked’ to look like java. Java in C++ clothing! As examples become more advanced the C++ nature of Arduino starts to dominate.

So just like there are advantages for Java and C++ in Arduino there are advantages for both when virtualizing applications. VBB started off virtualizing only the Java side of Arduino but unfortunately after a while Arduino diverges from Java and so the C++ side of Arduino is needed.

The nice thing about thinking about Arduino as Java is that it’s easier to virtualize some of the more interesting aspects of embedded applications so just like with Arduino the idea is to start with Java and migrate to C++ as you advance.

<table>
<thead>
<tr>
<th></th>
<th>Java</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrated Development</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Code Suggestions</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Cloud Compiler</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>
### Use with Arduino IDE
- No
- Yes

### User Library
- Inner Class
- C++/H

### 3rd Party Libraries
- No
- Yes – with Arduino IDE

<table>
<thead>
<tr>
<th>Feature</th>
<th>VBB4Arduino</th>
<th>VBB with ArduinoToolkit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Arduino Examples *</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Integrated Editor with Smart Suggestions</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Arduino .INO importer</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Arduino .INO exporter</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Single Step Debugger</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>Program Arduino device</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

#### 2.4 Comparison with VBB
VBB4Arduino is a simplified version of the full VBB product. It is intended as a standalone getting started version with less options and modules which can confuse starters. It is best used as a sandbox to learn physical computing concepts. Advanced users might like to progress to using the full version of VBB with the ArduinoToolkit expansion module.

- **Run Arduino Examples**
  - VBB4Arduino: ✓
  - VBB with ArduinoToolkit: ✓
- **Integrated Editor with Smart Suggestions**
  - VBB4Arduino: ✓
  - VBB with ArduinoToolkit: ✓
- **Arduino .INO importer**
  - VBB4Arduino: ✓
  - VBB with ArduinoToolkit: ✓
- **Arduino .INO exporter**
  - VBB4Arduino: ✗
  - VBB with ArduinoToolkit: ✓
- **Single Step Debugger**
  - VBB4Arduino: ✗
  - VBB with ArduinoToolkit: ✓
- **Program Arduino device**
  - VBB4Arduino: ✗
  - VBB with ArduinoToolkit: ✓
2.4.1 Upgrade
[TODO] There is an option to upgrade to Arduino Developer Bundle

2.4.2 Site License
[TODO] There is a site license available for schools.

2.5 Virtual Breadboard Circuit Modelling

Virtual Breadboard can be used as a Circuit Emulator for some types of Circuits. In particular VBB is not a SPICE simulation and does not currently resolve circuit current so you cannot use it for circuit-analysis. However for a wide variety of ‘physical computing’ circuits VBB emulations work just fine.

What's the difference between a Simulation and an Emulation?

Emulation models behaviour whereas simulation emerges behaviour. A behaviour might be to turn on a LED connected via a resistor to a microcontroller pin. A simulation might compute the circuit resistance and equivalent resistance of the power driver of the pin along with the voltage curve and forward voltage of the diode to resolve the instantaneous current and then lookup the luminosity curve of the diode to render a faithful representation of the color and intensity of the LED the user might expect to see. An emulation on the other hand just draws a LED as on when the PIN is HIGH and off when the PIN is LOW. The resulting behaviour from the user and microcontroller are the same. A LED is on when the pin is HIGH and the LED is off when the pin is LOW. Naturally emulations are faster to calculate and easier to implement. VBB is a circuit emulator and SPICE is a circuit simulator.

It's important to understand this difference when using VBB because you cannot place capacitors and resistors and expect for example behaviours such as filters to emerge. There are however common work arounds to achieve common behaviours using properties of the discrete components.

3 Download and Installation

All Vbb4Arduino versions is free to download. This makes it easy to manage updates etc. As a minimum VBB can be used to layout Breadboard designs for free. This is already useful and there are other popular Breadboard layout design tools that are limited to design only.

In addition VBB has virtualization features which you can optionally pay to enable.

3.1.1 Download Locations
Vbb4Arduino is available for download from

3.1.2 Installation VBB4Arduino
VBB4Arduino ships as a standalone executable packaged in a ZIP file.

Unzip the executable to the location where you would like to run it from. For example the desktop.
Execute VBB4Arduino.exe to start the application. Note this is not an installer and will not create shortcuts. You will need to create your own shortcuts if required

Dependencies
Vbb4Arduino requires .NET 4.0 runtime to be installed

3.1.2.1 Administrator Permissions
When you first run VBB you may need to use the Administrator account to give VBB4Arduino permission to install its configuration files in the AppData.
VBB will install its examples etc in the
<User>AppData\Local\Virtual Breadboard\VBB4Arduino <version> directory

3.2 Buying Vbb4Arduino

Purchasing a license will enable the Vbb4Arduino runtime allowing models to virtualize

3.2.1 How to buy
The goal of VBB4Arduino is to make a simplified version of VBB that supports circuit animation with limited features at a much lower ‘App’ level pricing.
Follow these steps to download and activate VBB4Arduino.

3.2.2 1. Purchase a license from PayPAL

3.2.3 2. Find the invoiceId
The invoiceId is the activation code for VBB4Arduino. I place the licenseKey on PayPal invoice so you can always look it up in the future in your paypal account.
3.3 3. Download the Software

**Product O/S** Download VBB4Arduino Version 5.34 - 5 September 2015

- **VBB**
- **VBB4Arduino**  See VBB4Arduino for Notes on Activation

  - VBB4Arduino *does not need to be installed.* Simply unzip the VBB4Arduino.exe and run

3.4 4. Enter the activation code in the splash screen

When VBB4Arduino starts it will be begin with a splash screen
Enter the activation code into the textbox and the code will be checked using the internet and the software activated. You should see a big Green tick once activated.

Press Start to continue using the software

3.5 Troubleshooting

You need an internet connection that can reach the Google App-Engine Server.
3.6 License Management

Earlier versions of VBB were released without any protection except a paywall. These versions were simply copied and shared without so much as a thank-you as a matter of fact. As a response, internet-based protections have been integrated into VBB.

Some of the features of VBB are now hosted in the cloud in the Google AppEngine server and accessed by license keys that are generated when features are purchased. It all works pretty seamlessly as long as you have a quality full time internet connection.

Fortunately the internet has become more or less ubiquitous but there are some situations where it doesn’t work out.

3.6.1 Firewalls

Internet Access is made via standard Http WebServices on Port 80. This looks just like Browser Access and usually there are no problems with firewalls.

However rarely, maybe 1 in 100 users find their configurations are not compatible with VBB. The primary problem is firewalls. There are 2 firewall that can cause issues.

3.6.1.1 Chinese Firewall

For some reason China blocks access to the Google App Engine host url. It’s quite a clever block too because I have tried to route via a proxy but it didn’t work. So for now the only solution is a VPN or we just refund Chinese customers and say sorry.

3.6.1.2 Work

Sometimes there is a domain/password proxy that needs to be configured. The ‘Classic’ versions have a setting for this now in the Tool ➔ Options dialog. The Windows App libraries don’t offer this same setting and rely on the Browser settings however this seems a bit indirect so a source of potential issues.

3.6.1.3 Antivirus/Internet Security

Applications such as Norton Internet security block internet access preventing authentication and pay-for-feature usage. You might have to add VBB.exe to the exception list of your Internet monitor and/or Firewall in order for Activation to function correctly.

3.7 Supporting Hardware

- VbbIO
- Virtual Shield

3.8 Customised Shields and Boards [future]

- PCBDirect service for Shields and Boards
4 VBB Application

VBB is an Integrated Development Environment for the purpose of designing, simulating and building microcontroller based electronics hardware systems.

There are 3 modes, Project Select, Design Mode and runtime mode.

The fundamental concept is the Design Sheet. Design sheets are contained in projects and projects are contained Solutions.

4.1 Solutions
A solution The VSM ‘Virtual System Model’ Solution file is the root file for VBB. It contains references to a collection of Projects and contains global settings for the solution. Vbb4Arduino solutions have only 1 project.

4.1 Projects Files
A project is a collection of Design Sheets.

4.2 Design Sheets

4.2.1 Breadboard
The Breadboard is the primary Design Sheet in VBB and consists of a collection of components wired together to create circuits.

4.2.2 Java Source Projects
The Source Project Design Sheet contains a collection of Java source files.

4.2.3 Sketch Source Projects
The Source Project Design Sheet contains a collection of Sketch source files.
that can be included in Vbb

4.2.4 Logic Analyser
The Logic Analyser is a special function DesignSheet containing a Logic Analyser instrument for circuit analysis.

4.3 Project Select
The project select form helps you create a new project or open an existing project. Select a tab to perform the task

- **Tips** – The Tips tab contains a list of tips that can be useful when getting started
- **New** : Create a new project from scratch or with a helper
- **Existing** : Open an existing project from the file system
- **Recent** : Open an existing project from a list of projects recently worked on

4.3.1 TIPS
The Tips tab contains a list of tips that can be useful when getting started

- **Check Tip Items** – Check each tip once you have read as a note to yourself that it has been read
- **Hide Seen Tips** – When checked the tips you have already read and checked as read will not display in the list
- **Show Tips Tab On Startup** – When checked the TIPS Tab will be selected as the default Tab when the project select dialog is show. Unselect to default to NEW or Existing
4.3.2 New Tab

The New Tab is used for creating New projects from project templates. A project template is a complete solution based on an example or project type which you can use to quickly get started. The templates are organized into a tree of project types to help you locate a template that best suits your embedded application challenge. The Project Types Tree on the left hand side selects the project type and the Project Templates lists the available templates for the selected type.

4.3.3 New Project Template

Select/Open or DoubleClick the New project template to create a new Blank project. The blank project created will contain a project called ‘New Solution’ with a single project called ‘New Project’ with a single Breadboard called ‘Breadboard0.VBB’.

The empty project is created in scratch disk.
4.3.4 New Arduino Emulator Project

Select/Open or DoubleClick the to create a new empty Arduino Emulator Project with Arduino Java code project.

The first step is to give the new project a name as source code projects can’t be saved to scratch disk.
4.3.5 New Arduino Simulator Project

Select/Open or DoubleClick the **New Arduino Simulator Project** to create a new empty Arduino Simulator Project with Arduino Sketch code project.

The first step is to give the new project a name as source code projects can’t be saved to scratch disk.
4.3.6 Import Arduino Sketch Wizard

Select/Open or DoubleClick the wizard to import an existing Arduino sketch as a project.
4.3.6.1  Step 1: Select the sketch to import using the browser.
Locate the sketch to import. See chapter on Java conversion to explain which project can be successfully imported.

Step 1: Browse for Arduino .INO sketches to import
Step 2: Select the template to use when creating the Breadboard
Step 3: Choose to import as a Java Emulator or Sketch simulator project
Step 4: Click OK to create the project

The import wizard can create either Arduino Java or Arduino Sketch projects.
4.3.6.2  Step 2: Choose a template

There are different templates preconfigured hardware suitable for different types of projects

4.3.6.2.1  Breadboard LED on Pin 13
Default example hardware with Pin13 attached to a LED same as a standard Arduino Board

4.3.6.2.2  Breadboard LED on Pin 13 + PORT A0 + Serial Miniterminal
Hardware with a serial console output and a potentiometer on A0 suitable for experiments reading analog signals and making reports on those signals.

4.3.6.3  Step 3: Choose a project type
Select the option button to choose the project type to create

4.3.6.3.1  Emulator Project
Emulator projects are based on the Java Emulator Virtual Arduino and Arduino Java Source. The imported sketch is converted to Java and linked to the Virtual Arduino

4.3.6.3.2  Simulator Project
Simulator projects are based on the AVR 328 Instruction set simulator Virtual Arduino and Arduino Sketch source. The imported sketch is copied into the project unchanged and linked to the Virtual Arduino
4.3.7 ‘Built-in’ template projects

In addition to creating new projects or projects based on existing Arduino code Vbb4Arduino contains numerous ‘built-in’ reference examples.

The examples are organised in a tree of example ‘project types’

To open a reference example click the folder on the left to show the available examples in the group then Select/Open or double click the example in the right Project Templates panel to open the example.

Copied Sketch Source Code

Breadboard template

Arduino sketch code project created with name of the imported source

AVR Simulator Arduino

Breadboard template

Copied Sketch Source Code
See Appendix XX for Listing and explanation of the reference Examples.
4.4 Existing Tab

The Existing Tab allow projects that already exist to be located and opened using the inbuild file browser.

The Default Directory is the location that is first shown. By default this is `<User>My Vbb4Arduino Projects` but you can change the default location using the Browse Button to navigate to a new folder location.
1. Existing Tab
2. List of VBB Solutions in the current directory
3. Drop down navigator
4. Move up to the previous directory

The navigation dropdown list allows you to search the computer for directories containing an existing VBB solution to open.
### 4.5 Recent

The recent tab holds the most recently used project list for quick access to recently used projects.

![Recent Tab Image]

<table>
<thead>
<tr>
<th>Project</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>AnalogInputSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\AnalogInputSim.VSM</td>
</tr>
<tr>
<td>NewSimulator.VSM</td>
<td>C:\Users\James\My VBB4Arduino Projects\NewSimulator.VSM</td>
</tr>
<tr>
<td>NewEmulator.VSM</td>
<td>C:\Users\James\My VBB4Arduino Projects\NewEmulator.VSM</td>
</tr>
<tr>
<td>SimProject.VSM</td>
<td>C:\Users\James\My VBB4Arduino Projects\SimProject.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\My VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>ButtonSim.VSM</td>
<td>C:\Users\James\My VBB4Arduino Projects\ButtonSim.VSM</td>
</tr>
<tr>
<td>SerialEvent.VSM</td>
<td>C:\Users\James\My VBB4Arduino Projects\SerialEvent.VSM</td>
</tr>
<tr>
<td>StringTool.VSM</td>
<td>C:\Users\James\My VBB4Arduino Projects\StringTool.VSM</td>
</tr>
<tr>
<td>ArduinoLibraryVSM</td>
<td>C:\Users\James\My VBB4Arduino Projects\ArduinoLibraryVSM</td>
</tr>
<tr>
<td>Blink.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\Blink.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>BlinkSim.VSM</td>
<td>C:\Users\James\AppData\Local\Virtual Breadboard\VBB4Arduino Projects\BlinkSim.VSM</td>
</tr>
<tr>
<td>WebClientNews2.VSM</td>
<td>C:\Users\James\My VBB Projects - USERS\WebClientNews2.VSM</td>
</tr>
<tr>
<td>WebClientNews2.VSM</td>
<td>C:\Users\James\My VBB Projects - USERS\WebClientNews2.VSM</td>
</tr>
<tr>
<td>WebClientNews3.VSM</td>
<td>C:\Users\James\My VBB Projects - USERS\WebClientNews3.VSM</td>
</tr>
<tr>
<td>WebClientNews3.VSM</td>
<td>C:\Users\James\My VBB Projects - USERS\WebClientNews3.VSM</td>
</tr>
</tbody>
</table>

![Status Bar Image]
4.6 Design Mode

When the project opens the state of the VBB UI moves into DesignMode with the following environment features.

The organisation is based around the Solution Explorer which is a tree of projects containing design sheets as leaves.

DesignSheets are Viewed in ViewPanes and to move a design sheet to a view pane it is dragged and dropped from the solution explorer tree into the design sheet.

Properties of Design Sheets and components in designs are shown in the Properties Editor.

4.6.1 App Window

The VBB Application is a Multiple Document Interface (MIDI) style application where the document being managed is the DesignSheet.
4.6.2 **App Menu**

The App Menu

- File: Access functions for opening and saving files
- Edit: Access functions for editing
- Tools: Access tools
- Help: Access application information

4.6.2.1 **File**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Shortcut</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🟢</td>
<td>New</td>
<td>Ctrl + N</td>
<td>Opens the New Project Dialog with the New Tab selected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Only available when no solution is loaded.</td>
</tr>
<tr>
<td>📜</td>
<td>Open</td>
<td>Ctrl + O</td>
<td>Opens the New Project Dialog with the Existing Tab selected.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Only available when no solution is loaded.</td>
</tr>
<tr>
<td></td>
<td>Close</td>
<td></td>
<td>Closes the current solution and returns to the closed solution state. Only available when a solution is loaded.</td>
</tr>
<tr>
<td>🚫</td>
<td>Save</td>
<td>Ctrl + S</td>
<td>Saves the current solution</td>
</tr>
<tr>
<td></td>
<td>Save As</td>
<td></td>
<td>Opens the Save Dialog prompting for the new project name. Only available when a solution is loaded.</td>
</tr>
</tbody>
</table>

4.6.2.2 **Edit**

<table>
<thead>
<tr>
<th>Icon</th>
<th>Name</th>
<th>Shortcut</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>🕹</td>
<td>Undo</td>
<td>Ctrl + Z</td>
<td>Undo’s the previous action(s)</td>
</tr>
<tr>
<td>🕹</td>
<td>Redo</td>
<td>Ctrl + Y</td>
<td>Redo’s the previous action(s)</td>
</tr>
<tr>
<td>🕹</td>
<td>Cut</td>
<td>Ctrl + X</td>
<td>Copies and deletes the current selection to the clipboard and deletes the selection from the current DesignSheet. Only available when a DesignSheet is active and components or code</td>
</tr>
</tbody>
</table>
have been selected.

<table>
<thead>
<tr>
<th>Copy</th>
<th>Ctrl + C</th>
<th>Copies the current selection of the current DesignSheet to the clipboard. Only available when a DesignSheet is active and components have been selected.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paste</td>
<td>Ctrl + P</td>
<td>Pastes the components copied to the clipboard to the current DesignSheet. Only available when a DesignSheet is active and components have been copied to the clipboard.</td>
</tr>
<tr>
<td>Delete</td>
<td>Del</td>
<td>Deletes the current selection from the current DesignSheet. Only available when a DesignSheet is active and components have been selected.</td>
</tr>
</tbody>
</table>

### 4.6.2.3 Help Menu

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update ICEShield Firmware</td>
<td>Starts the ICEShield firmware update dialog. See ICEShield</td>
</tr>
<tr>
<td>About</td>
<td>Opens the About dialog to display version information</td>
</tr>
<tr>
<td>Virtual Breadboard HomePage</td>
<td>Launches a Browser and opens the Virtual Breadboard HomePage</td>
</tr>
</tbody>
</table>

### 4.6.3 App Toolbar

The VBB application Toolbar gives quick access to the core Vbb4Arduino features

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Save</td>
<td>Saves the current project. If the project has not yet been saved it will launch the SaveAs dialog</td>
</tr>
<tr>
<td>PowerUp</td>
<td>Powers Up virtualization of the <strong>active circuit model</strong>. Only available if you have purchased a license and from Design Mode</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Verify</td>
<td>Performs a code check on source code projects</td>
</tr>
<tr>
<td>PowerDown</td>
<td>Powers down the circuit virtualization and returns to Design Mode</td>
</tr>
</tbody>
</table>

### 4.6.3.1 Active DesignSheet

The Active design sheet is the currently selected sheet which is the design sheet highlighted darker green

![Diagram](image)

1. **You can always click into the TitleBar to select the DesignSheet as the current DesignSheet**
2. Some DesignSheets allow you click anywhere on the DesignSheet to select it

### 4.6.4 Solution Explorer

The solution Explorer contains a tree representing the DesignSheets in the solution.
The Solution Explorer is context sensitive.

**Right Click on a Project** to add Logic Analyser, Breadboard, Arduino Java Source or Arduino Sketch Source DesignSheet

**Right Click on a Java Source Project** to add a Java source file to a Source Code Project

**Right Click on a Arduino Source Project** to add a Sketch, .cpp or .h source file to the Source Code Project

Deleting : Right click on the element to delete and select the delete drop down menu

Rename : Right click on the element to rename and in-place edit the new name
4.6.5 View Mode

The design panel has up to three design panes. Each pane may contain a different arrangement of DesignSheets.

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>One Panel</td>
<td>Show the design panel as a single design pane. Any selected DesignSheet is shown in the single pane</td>
</tr>
<tr>
<td>Two Panel</td>
<td>Splits the design panel into two separate design panes</td>
</tr>
<tr>
<td>Three Panel</td>
<td>Splits the design panel into three separate design panes.</td>
</tr>
</tbody>
</table>

When you double click a DesignSheet in the navigation tree it is selected into one of the available panes according to its default viewing position.

You can override the default location by dragging and dropping the design from the navigation tree to the desired panel.

When switching between layouts the previous layout remembers which DesignSheets are in which panel.

4.6.6 View Panes

The individual view panes are MDI child panes.
4.6.7 Design Sheets

DesignSheets are the documents managed by Vbb4Arduino. They have a design area and an optional Toolbox. There are four design sheets in VBB4Arduino.

- Breadboard
- Source Code Java
- Source Code Sketch
- Logic Analyser

4.6.8 DesignSheet Toolbox

The toolbox contains the collections of components specific to the DesignSheet.
4.7  The properties panel

Many items, DesignSheets and Components have Properties

The properties editor is populated when components are selected with DesignSheets. The properties are specific to individual components.

When you select a component the properties box is populated with the component properties.

4.7.1  The property description panel

When a component is selected the description of the property is displayed in the property description panel

4.7.2  Empty panel

When a panel has no DesignSheet it appears blue. You can fill the panel with a DesignSheet by dragging and dropping a design into the panel

4.7.3  Status Toolbar

Shows status information including runtime timing and debug information
4.8 Runtime Virtualization

When a circuit is powered up it moves into Runtime mode where the circuit is virtualized and can be explored for testing or learning purposes. Clicking the PowerDown button stops virtualization and returns Vbb4Arduino to design mode.

If there is more than one Breadboard in the project the Active Breadboard is launched.

5 Design Sheets

DesignSheets are the main document managed by the VBB system. There are 4 DesignSheets available in Vbb4Arduino

<table>
<thead>
<tr>
<th>Java Source Project</th>
<th>.SRC</th>
<th>The source code project is a special type of DesignSheet which contains a collection of java source code files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Icon</td>
<td>Description</td>
<td>File Extension</td>
</tr>
<tr>
<td>------</td>
<td>-------------</td>
<td>----------------</td>
</tr>
<tr>
<td><img src="image" alt="Arduino Sketch Project" /></td>
<td>Arduino Sketch Project</td>
<td>.SRC</td>
</tr>
<tr>
<td><img src="image" alt="Logic Analyser" /></td>
<td>Logic Analyser</td>
<td>VLI</td>
</tr>
<tr>
<td><img src="image" alt="Breadboard" /></td>
<td>Breadboard</td>
<td>.VBB</td>
</tr>
</tbody>
</table>
5.1 Arduino Sketch Code Project

The Source Project DesignSheet is a container for collections of source code.

The integrated Arduino Sketch projects are compiled using the VBB Hosted instance of the CodeBender compiler and the binaries execute on the Arduino SIM development board.

**Arduino Sketch**

Sketch Projects can contains

- Sketch – Arduino Java/Cpp style code
- Cpp – C++ code for user libraries
- .h – C code for user libraries

5.1.1 Adding a Sketch Project to a VBB Project

To add an Arduino Sketch project right click on the project in the solution explorer then select Arduino ➔ Add Arduino Sketch Source Project

5.1.2 Adding Source to Sketch Project

Arduino Sketch projects support 3 types of source files.

- Sketch (.ino)
- Header (.h)
- C++ (.cpp)

Inserting Code into a
5.1.3 Cloud Compiler : CodeBender

The Arduino Sketch projects as verified and compiled binary using a CodeBender cloud service hosted by virtualbreadboard.io cloud services. CodeBender has frozen Arduino support at Arduino 1.0.5 which is sufficient for working with all the examples and components available in the Vbb4Arduino sandbox.

There is no support for adding additional 3rd party libraries beyond those included in Vbb4Arduino or for upgrading to Arduino 1.6. For advanced Arduino use you should use the Arduino SIM component either in Arduino IDE monitor mode or VbbIO bootloader mode. Refer to the Arduino AVR Component.

5.1.4 Custom Libraries

You can add your own libraries using the C++/.h files directly into the project. The Libraries reference example shows how this is done. It is taken from the Arduino explanation of how to create libraries.

1) Add a new .h file and enter the library header code
2) Add a new .cpp file and enter the Cpp code
3) Include the library. **TIP: Use "" not <> for the code location**

5.1.4.1 Morse.ino

```cpp
#include "Morse.h"
Morse morse(13);
void setup()
{
}
void loop()
{
    morse.dot(); morse.dot(); morse.dot();
    morse.dash(); morse.dash(); morse.dash();
    morse.dot(); morse.dot(); morse.dot();
    delay(3000);
}
```
5.1.4.2  Morse.h

```
#include "Morse.h"
Morse morse(13);
void setup(){}

void loop()
{
    morse.dot(); morse.dot(); morse.dot();
    morse.dash(); morse.dash(); morse.dash();
    morse.dot(); morse.dot(); morse.dot();
    delay(3000);
}
```

5.1.4.3  Morse.cpp

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

Morse::Morse(int pin)
{
    pinMode(pin, OUTPUT);
    _pin = pin;
}

void Morse::dot()
{
    digitalWrite(_pin, HIGH);
    delay(250);
    digitalWrite(_pin, LOW);
    delay(250);
}

void Morse::dash()
{
    digitalWrite(_pin, HIGH);
    delay(1000);
    digitalWrite(_pin, LOW);
    delay(250);
}
```

5.1.5  Verifying Source

Arduino Sketch’s can be verified for correctness by clicking the Verify button in the main toolbar.

Click the Verify button to verify Active Sketch code
5.1.6 Supported Arduino Sketch Libraries

Supported Arduino Standard Libraries

- LiquidCrystal
- SoftwareSerial
- Servo
- Stepper
- EEPROM
- KeyPad
5.2 Java Source Code Project Design Sheet

The Source Project Design Sheet is a container for collections of source code.

The integrated Arduino Java projects are compiled using the VBB Hosted instance of a Java compiler and the classfiles execute on the Arduino JVM development board.

Arduino Java Source

Java Projects can contain

- Java – Arduino Java

5.2.1 Adding a Java Project to a VBB Project

To add an Arduino Sketch project right click on the project in the solution explorer then select Arduino ➔ Add Arduino Java Source Project.

5.2.2 Adding Source to Java Project

Arduino Java projects support only the java source file. To add a new java source file right click the Java Source Project and select he Add New Java Source drop down menu to show the Add Java Source Dialog.
The Add New Java Source File menu allows you to add new source files to the project and pops up a dialog to allow you to name or select the Java source file to add.

5.2.2.1 1 Suggested Source File name

When the Add New Java Source File dialog appears it has a suggested Classname. To add you a new file you can accept this name or rename it to your preferred name. The name must be unique to the source code project which means there must not be another file of the same name in the solution directory.

5.2.2.2 2. Solution Files

The Java source files for the solution directory are shown in the Solution Files list. It can be that the source file is in the directory, for example if you have copied it directly into the directory, but not yet included in the project. In this case you can select the name from the solution files list and click OK to include it.

5.2.2.3 3. Import

You might have source files you want to include in other directories. In this case you can click the import button to open a file dialog to allow you to select a file to import into the solution directory. Once imported the file will appear in the Solution Files list where you can select it to include it in the project as per (2)
Once you have named a new file or selected an existing one click the OK button to add it to the source project. If the file name you have selected is not unique you will receive the ‘File Already in Project’ message and the source file will not be added.

Inserting Code into a

5.3 Solution Tree Manager

In the project manager view the Java Source Code Project appears with the list of the source files it contains as a sub-tree member

5.4 Activating Java Source Project

You can select the Java Source Project as the currently selected design sheet by clicking the title banner (the normal blue title bar is hidden) or double clicking the Java Source Project in the solution tree. Some functions such as copy/paste won’t work as expected if the currently selected design sheet is not the Java Source Project containing the source editor you are working with.
5.4.1 Context Sensitive Functions

By right clicking on a named Java Source Code Project in the Solution Tree the context sensitive functions are shown.

5.4.2 Delete

You can delete a java source file by right clicking on it in the Solution tree and clicking the delete menu option.

The Java Source file will be removed from the Solution and also deleted from the solution directory.

5.4.3 Rename

You can rename delete a java source file by right clicking on it in the Solution tree and clicking the rename menu option.
The in-tree editor will then become enabled where you can edit the new name of the file.

When you click enter the name of the Tab source class name is also renamed but the class name in source is not renamed so you need to manually rename the source name. The name of the source name and the class name should match up.

5.4.4 Cloud Compiler: Java Compiler

The Arduino Java projects are verified and compiled binary using a Java cloud service hosted by virtualbreadboard.io cloud services.

There is no support for adding additional 3rd party libraries beyond those included in Vbb4Arduino as the Java libraries must match up to the fixed Virtual Breadboard Runtime Libraries (VbbRT).

5.4.5 Custom Libraries

You can add your own libraries using java subclasses or adding java classes.

5.4.5.1 Custom library subclasses

The advantage of using subclasses is that the code stays all together in the one source file and also you can directly access the underlying Arduino macro function such as digitalRead() etc directly from the subclass code.
5.4.5.2 Morse.java

```java
import muvium.compatibility.arduino.*;

public class Morse extends Arduino{

class MorseCode{
    int _pin;
    public MorseCode(int pin){
        _pin = pin;
        pinMode(pin, OUTPUT);
    }
    public void dot(){
        digitalWrite(_pin, HIGH);
        delay(250);
        digitalWrite(_pin, LOW);
        delay(250);
    }
    public void dash(){
        digitalWrite(_pin, HIGH);
        delay(1000);
        digitalWrite(_pin, LOW);
        delay(250);
    }
}

MorseCode morse = new MorseCode(13);

public void setup(){
}

public void loop(){
    morse.dot();
    morse.dash();
    morse.dot();
    delay(3000);
}
}
```

5.4.5.1 Custom library Java Classes

The advantage of using Java Classes as separate files is it makes larger projects easier to manage and you can more easily share code between projects.

However separate Java classes cannot access the Arduino macros and you need to pass the Arduino base class to access the Arduino functions.

To use separate java classes add addition Java source files to Source Project
MorseCode.java

```java
import muvium.compatibility.arduino.*;
public class MorseCode{
    int _pin;
    Arduino _arduino;

    public MorseCode(Arduino arduino,int pin){
        _pin = pin;
        _arduino = arduino;
        arduino.pinMode(pin, Arduino.OUTPUT);
    }
    public void dot(){
        _arduino.digitalWrite(_pin, Arduino.HIGH);
        _arduino.delay(250);
        _arduino.digitalWrite(_pin, Arduino.LOW);
        _arduino.delay(250);
    }
    public void dash(){
        _arduino.digitalWrite(_pin, Arduino.HIGH);
        _arduino.delay(1000);
        _arduino.digitalWrite(_pin, Arduino.LOW);
        _arduino.delay(250);
    }
}
```

Morse.java

```java
import muvium.compatibility.arduino.*;
public class Morse extends Arduino{

    //Constructor passes base for Arduino IO access
    MorseCode morse = new MorseCode(this,13);

    public void setup(){
    }

    public void loop(){
        morse.dot();
        morse.dash();
        morse.dot();
        delay(3000);
    }
}
```

5.5 Design Time Source Editor

The Java Editor at Design Time has useful features typical of modern source editor
5.5.1 Keyword Highlighting

The editor understands the code is a java and highlights the java keywords such as import, public, class, extends etc. This make the code more readable.

5.5.2 Syntax Error Highlighting

The editor can also detect when a syntax error has occurred with the java language structure and will highlight the syntax error with a red squiggle underscore to assist in locating and correcting the syntax error.

5.5.3 Tab Suggestions

As you type can get completion suggestions by typing {CTRL}+(SPACE)

This will popup a context sensitive context box based on you're the text entered so far.
You can navigate the suggestions box using the mouse or the {UP} and {DOWN} arrow keys.

Pressing {TAB} or {ENTER} will choose the selected suggestion and enter the text. This is useful for getting the exact syntax of a function based on its partial name.

5.5.4 Parameter Suggestions

When you have entered a recognised function name when you type the ‘(‘ you will receive parameter suggestions to help complete the function.

5.5.5 Edit Menu, Cut, Copy and Paste Undo

You can cut, copy and paste code sections in the Source Editor.

TIP: If the Java Source Editor Project is not the currently selected design sheet the Cut/Copy/Paste functions won’t work with the Source Editor.
5.5.6 Code Verification

Arduino Java code can be verified for correctness by clicking the Verify button in the main toolbar. Errors are shown in the Errors Tab.

Clicking the error in the Errors tab then the error line will be located and highlighted in yellow.

5.5.7 Java Source Console Tab

During runtime the Output tab shows the output of the standard output from the emulated code.
5.6 Logic Analyser

5.6.1 Trace Log (*.VLG)

The Logic Analyser traces signal events for logic analysis of circuits useful for debugging. The Logic Analyser works with Logic Probes that are placed onto Breadboard sheets and linked to the Logic Analyser Instrument from the Instrument property of the LogicProbe.

![Logic Analyser Instrument](image)

Design Sheet – The design sheet is a form which graphs signal events

Toolbar – None

Toolbox – None

5.6.2 Adding a Logic Analyser

![Adding Logic Analyser](image)

The trace log is found in the instruments project context sub menu. Right click on the project and select the Add Logic Analyser from the drop down context menu

5.6.3 Drag the Design Sheet into a View

The Logic Analyser doesn’t have a default view location so won’t appear until you drag it from the Project tree into a view pane
5.6.3.1 1. View Window
The trace grid is where the signal chart will be drawn. The probes sample the signal from the Breadboard sheet and are drawn into the TraceGrid. Up to 10 probes can be sampled.

5.6.3.2 2. Probe List
The Probe list is initially empty when the circuit emulation is first run the probe list becomes populated with a probe list and trigger elements for each probe.

5.6.3.2.1 1 Probe name
The name of the probe is taken from the name property of the probe on the Breadboard sheet.
5.6.3.2.2 2. Trigger Element

The trigger element for each probe is combined to make a trigger event which is detected by the logic analyser to begin recording. This makes it possible to capture specific parts of a signal.

There are 5 possible trigger options

<table>
<thead>
<tr>
<th>Icon</th>
<th>Trigger</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Don’t-Care</td>
<td>Not used in the trigger detection</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Rising Edge</td>
<td>Trigger occurs when signal moves from low to high</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Falling Edge</td>
<td>Trigger occurs when signal moves from high to low</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>HIGH</td>
<td>Trigger occurs when signal is HIGH.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>LOW</td>
<td>Trigger occurs when signal is LOW</td>
</tr>
</tbody>
</table>

A trigger is formed by AND the trigger elements together. For example the above trigger is ISRISING(0) AND ISHIGH(0). The left hand edge of the view window is marked by the trigger.

3. Previous Filter Element

Click to switch to the previous filter element

4. Next Filter Element

Click to switch to the next filter element
5.6.4  3. Sampling Control Panel

The sampling control panel determines how signals are captured and viewed in the View Window.

1. **Sampling LED**
   The sampling LED flashes when a trigger event occurs and a sample is made.

2. **TimeBase**
   The Timebase is the displayed time per view per horizontal grid in the View Window from 0.1 microsecond through to 1 second.
5.6.4.1.1  3. Next Time Base
Decreases the timebase

5.6.4.1.2  4. Previous Time Base
Increases the timebase

5.6.4.1.3  5. Trace Mode Toggle Button
Click to enable trace mode. In trace mode the signal is sampled continuously and refreshes each time a trigger event occurs.

5.6.4.1.4  6. Capture Mode Toggle Button
Click to arm capture. In capture mode the first trigger event is captured and viewed. The capture mode button becomes disabled. You can then analyse the resulting signal trace without it changing. To sample another signal you need to arm the capture mode again.

5.6.4.1.5  7. View Window Offset

Adds an offset to the view window left or right from the trigger event so you can view different parts of the signal in and around the trigger event

Offset = 0
Offset = ~+50% Shifts the left offset along to view the signal before the trigger
5.7  Breadboard DesignSheet

The Breadboard DesignSheet is a circuit emulation enabled drag and drop design sheet.

- **DesignSheet** – the DesignSheet a design graphic containing the component models
- **Toolbar** – the Breadboard toolbar is populated with graphical manipulation tools to zoom pan and manipulate the graphical design
- **Toolbox** – The Breadboard toolbox is populated with electronic component models which can be dragged and dropped on to Breadboard design

5.7.1  Toolbar : Breadboard

The graphics toolbar allows the manipulation of the Breadboard graphics, components and links.

<table>
<thead>
<tr>
<th>Toolbar</th>
<th>Button</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select Mode</td>
<td><img src="image" alt="Select Mode" /></td>
<td>Enters Select Mode. The cursor becomes an arrow. In select mode click on components to select them. Selecting a component populates the property box with the component properties.</td>
</tr>
<tr>
<td>Rotate Left</td>
<td><img src="image" alt="Rotate Left" /></td>
<td>Rotate Left. Rotates a component 90 degrees left.</td>
</tr>
<tr>
<td>Rotate Right</td>
<td><img src="image" alt="Rotate Right" /></td>
<td>Rotate Right. Rotates a component 90 degrees right.</td>
</tr>
<tr>
<td>Move Mode</td>
<td><img src="image" alt="Move Mode" /></td>
<td>Enters move Mode. The cursor becomes a NSEW pointer. In move mode you can drag and drop component to new location.</td>
</tr>
<tr>
<td>Link Mode</td>
<td><img src="image" alt="Link Mode" /></td>
<td>Enters link Mode. The cursor becomes a cross-hair and you can draw links between component pins.</td>
</tr>
<tr>
<td>Junction</td>
<td><img src="image" alt="Junction" /></td>
<td>Merges two links by joining with a junction.</td>
</tr>
<tr>
<td>Net</td>
<td><img src="image" alt="Net" /></td>
<td>Merges links with the same net name</td>
</tr>
<tr>
<td>Zoom</td>
<td><img src="image" alt="Zoom" /></td>
<td>Zoom Mode.</td>
</tr>
</tbody>
</table>
Click the Zoom button to enter zoom mode. When over the Breadboard layout the cursor will change into a Zoom graphic. Click and hold the mouse down and then moving vertically up will zoom out and vertically down will zoom in. Stays in zoom mode until another graphic mode is selected.

<table>
<thead>
<tr>
<th><strong>Zoom In</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Zoom in by factor 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Zoom out</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Zoom out by factor ½</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Zoom region</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Zoom to a region.</td>
</tr>
<tr>
<td></td>
<td><em>Not functional @v0.1</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Zoom Extents</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Zoom to the extents.</td>
</tr>
<tr>
<td></td>
<td><em>Not functional @v0.1</em></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Restore Origin</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Restore the origin to offset 0,0 and zoom =1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Lock origin</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Lock the origin when zooming. When locked the offsets don’t change only the zoom factor. When not locked the offset changes to keep the center of the screen fixed</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Pan</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Pan Mode.</td>
</tr>
<tr>
<td></td>
<td>Click the Pan button to enter Pan mode. When over the Breadboard layout the cursor will change into a Hand Zoom graphic. Click and hold the mouse down and then drag the display to pan around. Stays in Pan mode until another graphic mode is selected</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Grow</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Grow the selected components by factor 2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Shrink</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shrink the selected components by factor ½</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Link Color</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the link color. Sets the currently selected links to the selected color. Future links will be created with the new color.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Link Weight</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Set the link width. Sets the currently selected links to the selected weight. Future links will be created with the new weight.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Show Nets</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Show the virtual nets between named nets in the circuit board.</td>
</tr>
</tbody>
</table>

5.7.2 **Note on Zoom factor:**
The snap to grid has a problem when the zoom factor is not a multiple of 2. You should not try to select links or draw links at arbitrary zoom factors. The zoom, zoom extents, zoom region are best used to inspect the design and select components but you should restore the origin and use the zoom in, zoom out when drawing links to ensure the zoom is a multiple of 2.

5.7.3 Placing a component from the Toolbox

Placing a component is an important skill – not quite drag and drop. You need to click on the Toolbox icon releasing the mouse button. This attaches the component to the mouse pointer when you move over the Breadboard design. You can then drag the component into position in the DesignSheet and click a second time to place the component.

5.7.4 Component Editing

There are several ways you can work with components.

- Placing a component
- Select a component
- Select a group of components
- Append a component to a selection
- Move a group of selected components
- Copy and paste selected components
- Delete selected components

5.7.6 Select a component

To select a component enter Select Mode and left click on the component. Selected components are bounded by a dashed blue line.
5.7.7  Select a group of components

Select a group by drawing a window around the components to select.

Draw a window with a Left Click to anchor the window, and holding down the mouse button drag out a window.

Release the mouse to select the components fully inside the window.

5.7.8  Append a component to a selection

Append a component to a selection by holding shift and left clicking the component.
5.7.9 Move a group of selected components

To move a selected group of components

Select Move Mode by clicking the move state button.

Shift + Left Button drag and drop or Right button drag and drop.

5.7.10 Shrink and grow a selection

To increase density of certain parts of a graphical design it can be useful to scale down a component or group of components. Equally it is useful to be able to restore the size of the components and scale components back up.
5.7.10.1 **To scale down a component selection**
- Click the scale down component toolbar button.

5.7.10.2 **To scale up a component selection**
- Click the scale up component toolbar button.

5.7.11 **Wiring Essentials**

Wires are links from component pin to component pin.
To begin wiring click the wiring button

The cursor will become a crosshair

To start a link move to the pin to create a link from and click the left button

The link will attach to the pin and to the mouse cursor.

Move the cursor to the first joint of the link and press the left button again

Now the link is anchored to the last joint and the cursor

Click again with the left mouse on the next joint location or the destination pin or Double Click to finish the pin
Now the link is anchored to the destination pin and the cursor.

To finish the link right mouse button or double click.

When a link is made between two pins the link becomes thicker to indicate the link is an active link.
5.7.11.1 Working with Junctions

Junctions are used to merge links together.

Note: Links overlayed over each other are not considered a single link.

To create a junction click the junction mode and then:

- Links touching or crossing are not merged together without a junction.
- Enter Junction Mode and click with the left mouse to place junctions.
- Links need to be merged with a junction to become active.
5.7.11.2  Working with Net Labels

Net Labels allow connections to be made between pins by using a name instead of a wire. This can be used to better organise the connection layout.

The Net Toolbar option is used to place nets and features some helper options to add nets

5.7.11.2.1  New

New is the standard option to add a new net. When you add a new net it is placed with the Name new. You can edit the name by clicking the net name and editing the name in the property box

Drop Down List of Named nets

You can also select a net from the drop-down list in the property box. The dropdown list contains the names of all the nets currently on the Breadboard.
5.7.11.2.2 AutoInc

The AutoInc dropdown option has a further sub dropdown menu which allows you to select a net using the common prefixes (R = Resistor, C = Capacitor, L = Inductor, U = Integrated Circuit).

When you select from the submenu the net will be named +1 to the current highest net name. So for example if you have R1, R2, R2 on your breadboard already and you select R* then the net will be named R3.

5.7.11.2.3 DX / AX

The V18’O, Arduino and others have a naming convention for digital pins of D0, D1, D2 .. and Analog Pin A0, A1, .. These nets are also automatically added when using the V18’O, Arduino components so these are very commonly used net names. The AX and DX submenus gives you a shortcut for selecting these names.
5.7.12 All Nets (--)  

All the current nets on the Breadboard are listed below the – toolbar separator giving a shortcut to select a matching net for a net to be placed.

5.7.13 Virtual Links  

Nets can be thought of as acting a virtual link between the pins that are connected by the Net

Nets can be
- Connected to the end of links or midway along links like junctions
- Placed over pins directly

5.7.14 Show Links

You can view the links between the nets using the ShowLinks button. When you click the button it will show the links. It is a state button and when you unselect the button it will hide the links again.

5.7.14.1.1 Showing specific Net Links

To show a specific net link you can select a member of the net and click to ShowLinks button
5.7.14.1.2 Show subgroups of links

If you select members of more than one net then the links belonging to all the selected nets will be shown.

5.7.14.1.3 Predefined Nets

Arduino footprint boards have predefined nets associated with the pins.
By default the nets for these on but you can switch off the default nets using the components Nets property

<table>
<thead>
<tr>
<th>Properties</th>
<th>Manual</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>Manual</td>
</tr>
<tr>
<td>eeprom</td>
<td>512</td>
</tr>
<tr>
<td>eeprom size</td>
<td>ON</td>
</tr>
<tr>
<td>Nets</td>
<td>OFF</td>
</tr>
</tbody>
</table>

5.7.14.2 Component Layering

Components are layered, you move components up and down the layering using the context menu.

Select the component and press right click to start the context menu.
5.7.14.3 Locking a Component

The context menu contains an option for locking components. Select the component, right click to bring up the context menu and click Lock Control to lock the component. A locked component cannot be accidentally moved.

5.8 Toolbox: Breadboard

The Breadboard Toolbox is the most important Toolbox to VBB as it contains the component models that constitute the circuits.
6  Breadboard Components

VBB components are models of real world electronic components.

**Important!** VBB does not support analog circuit simulation like SPICE. Sorry but VBB just doesn’t work that way. Instead VBB is a circuit modelling tool with a focus on the application code and uses simplified electronic models to virtualize the electronics. Nevertheless a huge number of circuits can be modelled this way especially if you take the time to learn a few VBB tricks.

There are two types of Components that can be placed in a Breadboard

- Component Models which virtualize
- Generic Layout which do not virtualize

These can be used in combination in a circuit but only the sections of the circuit that contain Component Models will virtualize. In addition, there is a special function block component which is a user programmable way to model the behaviour of many common circuit blocks.

The concept of the function block is the observation that in many cases electronic designs are simply copy and pasted again and again by designers who rely only on the behaviour of the circuit design and really don’t care about the circuit itself.

*Why resolve the entire analog behaviour of a circuit block which performs a known fixed function?*

Why not just take a picture of the circuit block and attach a FunctionBlock that models the same circuit behaviour. That is the philosophy of VBB and for many circuits relevant to Arduino users it’s a good first approximation. Of course you need to learn a bit about modelling circuit blocks instead of hoping VBB will figure it all out for you. There are some packages out there that do that but then you lose the advantages of circuit modelling such as performance and ease of design – do you really want to have to fully understand electronic design, select every resistor and capacitor value exactly before you even begin the fun stuff of virtualizing and programming? Didn’t think so ! ..

7  Generic Layout Components

Layout Components are generic graphics only components and will not emulate. However when used in combination with a function block they can appear to emulate. These components are parametric models of the common hole-through components. Layout components help you create a circuit board which can be fabricated by our PCB service or as a guide when assembling a real circuit board even where specific components models are not available.

7.1.1  Generic Layout Only

- Generic DIP
- Axial
7.2 Generic DIP

Breadboards are useful because they have pin sockets that match the spacing of the standard DIP packaging of electronic circuits. The Generic DIP component allows you to easily create a DIP component package.

DIPs components are typically placed across the center section.
7.3 Parametric

The DIP component is parametric so you can create any DIP by setting the component properties accordingly.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifier used in BOM</td>
</tr>
<tr>
<td>Part Label</td>
<td>The part name shown on the packaging</td>
</tr>
</tbody>
</table>
### Part Logo
An optional logo that is shown center of the package. Uses magenta for transparent color.

### PinCount
The number of pins (multiple of 2).

### Spacing
The size of the spacing between pins – in hundreds of mil – typically 3 or 5.

### Pin Names
A CSV (comma separated array) of names starting from pin 1 to pin \( \text{PinCount} \) of the pins labels. This is shown as an overlay as a form of schematic for the part.

### Library
The Library is a special property which activates as Custom Dialog to select a component from a library which then sets the properties.

#### 7.3.1 Using the Library Component

When you first place a Generic DIP the properties are blank. You can enter your own values or select from the library. To use the library:

1. Click on the Library property to show the custom editor button.
2. Click the custom editor button to show the Component Library Dialog

Select a component from the library
3. Click the component group

4. Select the component

5. Click OK

This sets the properties for the component creating the component.
7.4 Marking up a generic component with a function block

You can ‘add virtualization’ to a generic component footprint by marking it up with one or more function blocks.

For example you might want to virtualize a dual AND gate

1. Place a suitable generic component

   Select a 4081 Quad 2-input AND gate using the generic library

2. Place a Function block and give it 2 inputs
Referring to the 4081 datasheet or by inspection of the pins on the top of the DIP component we want to assign the function block the pin logic of the 4081

So Function Block Output 1Y = AND(1A,1B)

The easiest way to do this is to use the netlist. From the properties of the 4081 copy the pin names over to the netlist field to assign pin names to the known pin positions

<table>
<thead>
<tr>
<th>Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Part Label</td>
<td>4081</td>
</tr>
<tr>
<td>Part Logo</td>
<td></td>
</tr>
<tr>
<td>PinCount</td>
<td>14</td>
</tr>
<tr>
<td>Spacing</td>
<td>3</td>
</tr>
<tr>
<td>Pin Names</td>
<td>1A,1B,1Y,2Y,2A,2B,GND,3A,3B,3Y,4Y,4A,4B,VCC,</td>
</tr>
<tr>
<td>Library</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td></td>
</tr>
<tr>
<td>netlist</td>
<td>1A,1B,1Y,2Y,2A,2B,GND,3A,3B,3Y,4Y,4A,4B,VCC,</td>
</tr>
</tbody>
</table>

Now in the function block properties copy over the relevant net names to the match the input and outputs. Conveniently in this example they are aligned at pin 1,2,3 so it’s just a matter of copying over the first 3 net names from the 4081 to the function block
<table>
<thead>
<tr>
<th>Properties</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
<td>Function Block</td>
</tr>
<tr>
<td><strong>Inputs</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>Expression</strong></td>
<td>AND(P1,P2)</td>
</tr>
<tr>
<td><strong>ID</strong></td>
<td></td>
</tr>
<tr>
<td><strong>netlist</strong></td>
<td>1A,1B,1Y</td>
</tr>
</tbody>
</table>

Finally assign the Expression AND(P1,P2) to create an AND gate from pins 1 and 2 with the output on pin 3, which is how the 4081 is specified.

You can preview the virtual links using the show links and functionblocks options.

At runtime you can hide the links and function block to make it appear the 4081 generic component footprint is virtualized.
7.5 Radial Component

Generic Radial component can be used to model components with a 2 pin Radial configuration for which there is no matching visual component.

7.5.1 Parametric Model
### 7.5.2 Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifier used in BOM</td>
</tr>
<tr>
<td>Spacing</td>
<td>The number of grid spacings between pins</td>
</tr>
<tr>
<td>Diameter</td>
<td>The diameter of the radial component in grid spacings (0.1&quot;)</td>
</tr>
<tr>
<td>Color</td>
<td>The color of the Radial packaging</td>
</tr>
<tr>
<td>Value</td>
<td>The value of the part to appear in the BOM. Eg Capacitance or special part no.</td>
</tr>
</tbody>
</table>

### 7.6 Axial Component

Generic Axial component can be used to model components with a 2 pin Axial configuration for which there is no matching visual component.

### 7.6.1 Parametric Model
7.6.2 Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifier used in BOM</td>
</tr>
<tr>
<td>Spacing</td>
<td>The number of grid spacings between pins</td>
</tr>
<tr>
<td>Diameter</td>
<td>The diameter of the axial component in grid spacings (0.1”)</td>
</tr>
<tr>
<td>Length</td>
<td>The length of the axial component in grid spacings (0.1”)</td>
</tr>
<tr>
<td>Color</td>
<td>The color of the Radial packaging</td>
</tr>
<tr>
<td>Value</td>
<td>The value of the part to appear in the BOM. Eg Capacitance or special part no.</td>
</tr>
</tbody>
</table>

7.1 Generic Header Component

Generic Radial component can be used to model all different header configurations.
### Generic Male Header

Generic male header component can be used to model many different header configurations.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pins</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Rows</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Color</td>
<td>Black</td>
<td>Black</td>
<td>Blue</td>
</tr>
</tbody>
</table>

#### 7.1 Generic Male Header

- **Pins**: Enter the number of horizontal pins. Default: 2. Options: 2 | 3 | 4 | 8 | User Enter.
7.1 **Generic Female Header**

Generic male header component can be used to model many different header configurations.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pins</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Rows</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Color</td>
<td>Black</td>
<td>Black</td>
<td>Blue</td>
</tr>
</tbody>
</table>

7.1 **Screw Terminal**

Generic male header component can be used to model many different header configurations.
<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pins</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Seperation</td>
<td>.1</td>
<td>0.1</td>
<td>0.2</td>
</tr>
<tr>
<td>Color</td>
<td>Green</td>
<td>Names Color</td>
<td>Named Color of the packaging. See the Appendix.</td>
</tr>
</tbody>
</table>
8  Circuit Model Components

VBB has some special circuit model components which can be used to model or ‘mock’ many different types of circuits relevant to the Arduino user.

- **Function Blocks** are programmable logic blocks that can overlay circuits to mock circuit behaviour.
- **Resistors** are modelled either as a fuse or an open-circuit.
- **Capacitors** are modelled as an open-circuit.
- **Diodes** are modelled as switched pull-up or switched pull-down resistor.
- **NPN Transistors** are modelled as a switch which turns on when the base voltage is HIGH.
- **PNP Transistors** are modelled as a switch which turns on when the base voltage is LOW.

8.1  Function Block

The Function Block is a powerful circuit modelling block and can be used in Co-Virtualization of many different circuits.

When the function block is used in combination with generic layouts there is special support to hide the function block to make it looks like the circuit block is virtualizing. It’s a type of component development kit.

The purpose of the Function Block component is to model certain circuit behaviour not directly supported by VBB.

![Function Block Diagram]

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The Name of the Function Block for circuit documentation</td>
</tr>
<tr>
<td>Inputs</td>
<td>The number of inputs. When this value changes the Function Block diagram will automatically resize.</td>
</tr>
<tr>
<td>Expression</td>
<td>The FunctionBlock Expression.</td>
</tr>
<tr>
<td>ID</td>
<td>The device ID in the BOM list</td>
</tr>
</tbody>
</table>
8.1.1 The Function Block expression

The function block expression calculates an output voltage based on its input pins. It works in a similar way to an Excel function.

\[ V_{out} = f(P_1, P_2, \ldots, P_n) \]

For example

- ADD(P1,P2) 'Voltage Summer'
- MUL(SUB(P1,P2),2) 'Differential Voltage Multiplier'

When used the Function Block graphic will change to reflect the names for the input pins.

8.1.1.1.1 Summer Concept Example

A voltage summer is a circuit function that sums two or more voltage inputs.

\[ V_{sum} = V_1 + V_2 \]

This is a common circuit block often implemented with operational amplifiers.

However there are other ways to sum voltages such as by using digital sampling.

The focus of VBB is on equivalent circuits so it doesn't matter how the circuit implements a voltage adder VBB can model it using a function block.
To create a voltage summer you use the ADD function in the function block expression.

You could use a function block alongside Generic components in a VBB circuit and it would work the same and look the same as if it’s behaviour we evolved from the discrete circuit elements the way SPICE does it but the end result is the same.

8.2 Function Block Expression

8.3 Function Block Expression Grammar

```
Expression = Primitive | Function
Primitive ::= Numeric | Pn | PINn
Function ::= functionName( expression [, expression]* )
functionName ::= CONST | RISING | FALLING | ISHIGH | ISLOW | PULSIN | CINT | NAND | NOR | XOR | XNOR | MIN | MAX | ABSDIFF | CSIGN | CEN | SIGN | PWM | NEG | IF | MUL | ADD | SUB | RANGE | MAP | OR | NOT | GT | LT | VOLTS | DAC
```

Function Block Expressions
FunctionName ::= Pin Functions | Logic Functions | Math Functions | Special Functions

Pin Functions ::= ISHIGH | ISLOW | RISING | FALLING

Logic Functions ::= AND, OR, XOR, NOT, NAND, NOR, XNOR

Math Functions ::= ABSDIFF, ADD, CEN, CONST, CSIGN, DIV, GT, IF, LT, MAP, MIN, MUL, NEG, RANGE, SIGN, SUB

Special Functions ::= DAC, PULSIN, PWM, VOLTS

Expressions can often contain other functions as parameters. For example:

AND(OR(P1, P2), OR(P3, P4)) Complex Logic

ADD(P1, P2) 'Voltage Summer

MUL(SUB(P1, P2), 2)' Differential amplifier gain = 2

Pins

PN return a value 0 for logic level GND and 1 for Vdd which is nominally 5V

Pin Functions:

ISHIGH (PIN) : Returns 1 if the PIN is HIGH else returns 0
ISLOW( PIN ) : Returns 1 if the PIN is LOW else returns 0
RISING( PIN ) : Returns 1 if the PIN has a rising edge
FALLING( PINS ) : Returns 1 if the PIN has a falling edge

Logic Functions:

AND(F1, F2, ..., F3): Returns 1 if all FN return 1
OR(F1, F2, ..., FN): Returns 1 if any FN return 1
XOR(F1, F2, ..., FN): Returns 1 if equal numbers of FN are 1 and 0
NAND(F1, F2, ..., F3): Returns 1 if any FN is 0
NOR(F1, F2, ..., FN): Returns 1 if no FN is 1
XNOR(F1, F2, ..., FN): Returns 1 if odd numbers of FN are 1 and 0
NOT(F1): Return 1 if F1 is 0
Math Functions:

ABSDIFF(FA,FB): Evaluates a $|FB-FB|

ADD(FA,FB,...,FN): Evaluates $FA+FB+\cdots+FN$

CEN(FA): Evaluates $|FA-FB|-0.5$

CINT(FA): Converts floating FA to integer

CONST( string ): Evaluates string as numeric

CSIGN(FA): if FA > 0.5 returns 1 else returns 0

DIV(FA,FB,...,FN): Evaluates $FA\div FB\div\cdots\div FN$

GT( FA, FB ): If FA > FB returns 1 else returns 0

IF( FA, FB, FC ): If FA $\neq$ 0 returns RB else returns FC

LT( FA, FB ): If FA < FB returns 1 else returns 0

MAP( FA, FB, FC ): Evaluates $FA-FBFC-FB$ clipped to range 0 to 1

MIN(FA,FB,...,FN): returns the lowest value in the parameter list FA,...,FN

MAX(FA,FB,...,FN): returns the highest value in the parameter list FA,...,FN

MUL(FA,FB,...,FN): Evaluates $FA*FB*\cdots*FN$

NEG( FA ) : Evaluates $-FA$

RANGE( FA, FB, FC ) : Evaluates $FA+(FB-FA)*FC$

SIGN( FA, FB ) : if( FA > FB ) returns 1 else returns 0

SUB( FA,FB,...,FN ) : Evaluates $FA-FB-\cdots-FN$

Special Functions

DAC(FA,FB,...,FN): Digital To Analog converter using N bits to create analog range between 0 and 1

PULSIN( FA ): Measure the time in seconds of a pulse where the time is measured between the rising and falling edge
PWM (FA): Digital to Analog converter using the PWM pulse duty on FA to create an analog value between 0 and 1

PWM(FA,FB): Digital to Analog converter using the PWM pulse duty on Differential FA - FB to create an analog value between 0 and 1

VOLTS (FA): Converts the result of FA into a volts value by dividing by the nominal voltage value of 5.

8.3.1 Function Block Editor Dialog
The function block editor dialog is a

8.4 Function Block Usage
Pullup Resistor

A pullup resistor is a very common circuit block very often used to create voltage dividers or convert button presses into high/low values.

For example when a momentary button which is connected to ground is connect to a resistor is connected to
The first trick is to set the resistor is *open-circuit* mode. This means the resistor is just a picture and its value is not used in the resulting circuit behaviour.

The output of the momentary button is the input to the function block and it has two values. O/C *open-circuit* when the switch is off and 0.0 *ground* when the switch is on.

To model the O/C open-circuit needs to be converted to 1 = HIGH = 5V

**Pullup resistor := IF(ISLOW(P1),0,1) = NOT(ISLOW(P1))**

ISLOW(P1) returns 1 when the voltage at P1 is LOW = 0.0 and otherwise it returns 1. However the output of the function block needs to invert this which can be done with an NOT or IF.

### 8.4.1 Dynamic Voltage Divider

A voltage divider is a common circuit configuration. Consider a voltage divider which is tapped between two resistors and momentary switch connected to ground on the bottom of the ladder. The voltage divider is has the well known formula

\[ V_{out} = \frac{R_2}{R_1+R_2} \cdot V_{in} \]
So for resistors 2K and 950Ohm with 5V rail we can derive the output we expect when the button is pressed to be $V = \frac{950}{2K + 950} \times 5 \approx 1.6V$

**IF(ISLOW(P1), VOLTS(1.6),1)**

If the button is pressed P1 becomes 0.0 so ISLOW is true and becomes 1 which IF evaluates and returns the first term VOLTS(1.6). VOLTS is a function that converts 1.6 volts into the nominal range which is 5V by default so VOLTS(1.6) return $(1.6 / 5)$.

If the button is not pressed P1 is O/C is ISLOW is false and becomes 0 which IF evaluates as false returns the second term 1 which is nominally HIGH or 5V.
8.4.2 Multi Tap Voltage Divider

If your microcontroller doesn't have many spare pins it can be useful to decode multiple buttons using a single analog pin.

The way you can do this is to use the principle of Voltage Division to create different voltage levels depending on the button pressed. This voltage can then be read as an analog value and compared with calculated values to decode the button pressed.

The trick is to realise this is just a voltage divider

Voltage Divider R1 = 2K
Voltage Divider R2 = 330 + 620

\[ V_{ButtonDown} = \frac{R_2}{R_1+R_2} \cdot V_{in} = \frac{330+620}{330+620+2K} \cdot 5V = \frac{950}{2950} \cdot 5V = 1.61V \]

So we can see each button press gives a unique voltage level we can decode
- VRight = 0.0V
- VUp = 0.7V
- VDown = 1.6V
- VLeft = 2.45V
- VSelect = 3.7V
To model this circuit block what we end up with is a FunctionBlock that merges the 5 buttons.

It is intended that FunctionBlocks can be hidden behind the scenes and a good way to do that and keep the circuit models organised is to use Net Labels and netlists. You can see in the example the net labels on each button and these labels are matched by the netlist names of the FunctionBlock.

**TIP:** The hidden links are shown as the Jade coloured lines and the FunctionBlock is visible these can be toggled visible or hidden using the toolbar netlist toggle buttons.
The expression uses the **MIN** function to select the minimum voltage from the switched button dividers created by the 5 separate buttons which gives us the final expression.

```plaintext
MIN

( IF(ISLOW(P1),VOLTS(0),1)
  IF(ISLOW(P2),VOLTS(0.7),1)
  IF(ISLOW(P3),VOLTS(1.6),1)
  IF(ISLOW(P4),VOLTS(2.45),1)
  IF(ISLOW(P5),VOLTS(3.7),1)
)
```

One difference to the original schematic is the use of the protection resistor to the ADC output of the function block. This is used actually as a trick to separate the output of the topmost right button to the output of the function block. Without it VBB would signal a short circuit but it's a useful circuit addition that doesn't change the voltage output.

Virtualization
Practical Application
VBB is primarily about firmware development support so it's only important that the circuit model behaves like the real circuit does from the point of view of the microcontroller application code.

As you can see the microcontroller decoder code can't tell the difference and the decoder code works just the same as it does in real world.

So to sum up. Use VBB function blocks to model circuit behaviour such that the microcontroller code can't tell the difference between the virtual model or a real circuit. Then you can happily develop and test your application virtually before committing to hardware.
8.4.3 Resistors

Generic resistor component models the standard axial resistor with parametric spacing and resistance color bands.

8.4.3.1 Parametric Model

8.4.3.2 Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifier used in BOM</td>
</tr>
<tr>
<td>Resistance</td>
<td>The resistance of resistor – the color markings update according to resistance</td>
</tr>
<tr>
<td>Spacing</td>
<td>The number of grid spacings between pins</td>
</tr>
<tr>
<td>Function</td>
<td>Emulation Mode Fuse</td>
</tr>
</tbody>
</table>

8.4.4 Resistors

Resistors have three modes which can be selected from the Function property of the resistor.

- fuse (Default)
- wire
- Pullup
- Pulldown
- open-circuit (Document)
8.4.4.1 Fuse Mode
In fuse mode the resistor acts as a poly fuse which if there is a short circuit will break it’s connection else it will make a wired a connection. Fuse mode is the most default mode and the most versatile as it can represent a regular resistor, pullup and pulldown.

As a polyfuse when a resistor is open circuit on end of the resistor by the other end is 5V then the polyfuse is a wire and the other side of the resistor sees the 5V

When one end is 5V and other end is 0V, a short-circuit, the polyfuse breaks and an open-circuit is created

8.4.4.2 Pullup Mode
Pullup mode can be used for resistors that are dedicated as pullups. The main reason to use a resistor in pullup mode is when used as a basic voltage divider in conjunction with a pulldown resistor. Also there can be a minor performance improvement over using fuse mode

A Pullup resistor pulls a voltage HIGH if its not driven LOW.

One end of the pull-up resistor is assumed to be connected to Vdd even if its not actually wired that way.

A common example shown here is to attach a pullup to a momentary switch so that the output is normally HIGH and switched LOW when the button is pressed.
### 8.4.4.3 Pulldown Mode

A Pullup resistor pulls a voltage LOW if its not driven HIGH. The main reason to use a resistor in pullup mode is when used as a basic voltage divider in conjunction with a pulldown resistor. Also, there can be a minor performance improvement over using fuse mode.

*One end of the pull-down resistor is assumed to be connected to Gnd even if its not actually wired that way*

A common example shown here is to attach a pulldown to a momentary switch so that the output is normally LOW and switched HIGH when the button is pressed.

![Diagram showing pulldown resistor](attachment:image)

### 8.4.4.4 Voltage Divider

Pullup and Pulldown resistors can also be used in some Voltage Divider configurations.

Voltage dividers are used for creating reference voltages, or even digital to analog converters. In this example a 4-bit analog to digital converter is created by ladder. The resistors are switched together using a DIP switch array to create a network of resistor values. This circuit will emulate because the pullups are connected to VDD and the pulldowns are connected to GND. There are other Voltage divider networks which dynamically drive the resistors, these cannot be emulated directly.
8.4.4.5 **Wire Mode**

In normal mode a resistor function the same as a regular wire. This can be used instead of fuse mode as there are minor performance improvement compared to a fuse.

8.4.4.6 **Document Mode**

In document mode a resistor does not participate in the circuit.

### 8.5 **Diode**

Generic diode component models the standard axial diode with parametric spacing.

#### 8.5.1 **Parametric Model**

- **Pin Spacing**

#### 8.5.2 **Properties**
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifier used in BOM</td>
</tr>
<tr>
<td>Spacing</td>
<td>The number of grid spacings between pins</td>
</tr>
<tr>
<td>Function</td>
<td>Emulation Behaviour</td>
</tr>
<tr>
<td></td>
<td>Document</td>
</tr>
</tbody>
</table>

8.5.3 Diodes

Diodes are modelled as switched pull-up or switched pull-down resistor.

- document *No connection in circuit*
- VDD *Switched Pullup*
- GND *Switched Pulldown*

VDD and GND Mode

There are two common modes supported:
- In GND mode the diode acts as a pulldown resistor when connected to ground.
- In VDD mode the diode acts as a pullup resistor when connected to VDD.
8.5.3.1 Document mode

Diodes can also be used in document mode. A common example is a protection diode you often see in circuits across solenoid and motors. The protection diodes can be placed using document mode so they appear in the circuit but don't participate in the emulation.

There are lots of other circuit tricks that use diode non-linearity and other diode characteristics. These won't work in VBB and require the use of document mode and functionblocks.

8.6 PNP Transistor

PNP Transistors are modelled as a switch which turns on when the base voltage is LOW.

8.6.1 Properties:
ID, Netlist
8.6.2 Usage

Switch connects the PNP B base to ground (through the base resistor) causing it to activate its internal switch connecting (E)mitter to (C)ollector which connects the anode(+) of the LED to the 5V rail causing it to flash on.

Switch connects the PNP B base to Open Circuit (through the base resistor) causing it to disconnect its internal switch connecting (E)mitter to (C)ollector which disconnects the anode(+) of the LED from the 5V rail causing it to flash off.
### 8.7 NPN Transistor

**NPN Transistors** are modelled as a switch which turns on when the base voltage is HIGH.

#### 8.7.1 Properties:

**ID,Netlist**

#### 8.7.2 Usage

Switch connects the NPN B base to Vcc(5V) (through the base resistor) causing it to active its internal switch connecting (E)mitter to (C)ollector which connects the anode(+) of the LED to the 5V rail causing it to flash on.

Switch connects the NPN B base to Open Circuit (through the base resistor) causing it to disconnect its internal switch connecting (E)mitter to (C)ollector which disconnects the anode(+) of the LED from the 5V rail causing it to flash off.
8.8 **Ceramic / Tantalum / Electrolytic Capacitor**

Generic capacitor components

8.8.1 Properties

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Identifier used in BOM</td>
</tr>
<tr>
<td>Capacitance</td>
<td>The capacitance value of the capacitor to appear in BOM</td>
</tr>
</tbody>
</table>
9  Microcontroller Components

The microcontroller components are the Arduino Twins’ Em and SIM. Em runs the java code in a java virtual machine. SIM runs AtMega328 instruction set simulator and executes the compiled Arduino binary code.

9.1  Arduino EM

9.1.1  Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0-&gt;RX(In)</td>
<td>Digital 0 and Hardware UART RX pin</td>
</tr>
<tr>
<td>2</td>
<td>1&lt;-&gt;TX(In)</td>
<td>Digital 1 and Hardware UART TX pin</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Digital 2</td>
</tr>
<tr>
<td>4</td>
<td>3~</td>
<td>Digital 3 with PWM output</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>Digital 4</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>Digital 5 with PWM output</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>Digital 6 with PWM output</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>Digital 7</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>Digital 8</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>Digital 9 with PWM output</td>
</tr>
</tbody>
</table>
### Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Arduino</td>
<td></td>
<td>The device name used when selecting a target where more than one device shares a circuit</td>
</tr>
<tr>
<td>Application</td>
<td></td>
<td>Dynamic List of Source Code Project</td>
<td>The name of the source code project to pull the code from when virtualizing</td>
</tr>
<tr>
<td>EEPROM</td>
<td></td>
<td>Editor</td>
<td>EEPROM values saved and loaded to the eeprom at runtime</td>
</tr>
<tr>
<td>Eeprom size</td>
<td>512</td>
<td>512</td>
<td>1024</td>
</tr>
<tr>
<td>Nets</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
</tr>
</tbody>
</table>

**Application**

There can be more than one source code project. You might for example have different test configurations for a single hardware layout. You can choose the application to run by selecting it from the dropdown list.
9.1.3 EEPROM

The EEPROM is the internal device EEPROM. It is a persistent store which persists between runtime sessions. The size of the eeprom is set by the ‘eeprom size’ property which can be changed to align with the different physical devices implementing the Arduino EM device.

You can inspect the contents of the EEPROM, edit the contents, save and load the contents from the EEPROM editor.

Launch the editor by selecting the eeprom property and clicking the custom editor button

The edit can use a fix file as a ROM by using the [x] use file checkbox and browsing to a file. The file uses the microchip eeprom image MCH format which is just a file with one 2 digit hex byte entry per line.

You can export the EEPROM image to create a template

Import an EEPROM image to load known settings etc

The EEPROM values are saved as a CSV array in the property and the contents are updated at runtime so carry over between power on sessions

The s
The Arduino SIM is an AVR ATMega328 instruction set simulator. It can execute the binaries generated by the Arduino IDE.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>O-&gt;RX(In)</td>
<td>Digital 0 and Hardware UART RX pin</td>
</tr>
<tr>
<td>2</td>
<td>1&lt;-&gt;TX(In)</td>
<td>Digital 1 and Hardware UART TX pin</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>Digital 2</td>
</tr>
<tr>
<td>4</td>
<td>3~</td>
<td>Digital 3 with PWM output</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>Digital 4</td>
</tr>
<tr>
<td>6</td>
<td>5</td>
<td>Digital 5 with PWM output</td>
</tr>
<tr>
<td>7</td>
<td>6</td>
<td>Digital 6 with PWM output</td>
</tr>
<tr>
<td>8</td>
<td>7</td>
<td>Digital 7</td>
</tr>
<tr>
<td>9</td>
<td>8</td>
<td>Digital 8</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>Digital 9 with PWM output</td>
</tr>
<tr>
<td>11</td>
<td>10</td>
<td>Digital 10 with PWM output</td>
</tr>
<tr>
<td>12</td>
<td>11</td>
<td>Digital 11 with PWM output</td>
</tr>
</tbody>
</table>
9.2.1 Application Types

9.2.1.1 HEX File

The filename property points to a .hex binary that is to be loaded from the file at runtime.

9.2.1.2 Arduino IDE Monitor

To work with the Arduino IDE you should use the VBB project directory as the Arduino directory and you should name the main Arduino .ino file the same as the directory.
So for example if you have created a new project called BlinkSim then the Arduino file should also be called BlinkSim.ino.

When starting in Arduino IDE Monitor mode the Arduino IDE will launch a new session with this BlinkSim.ino file loaded and will start monitoring the project directory for new HEX files.

To load a new HEX file you should use the Arduino IDE Tools->Export Compiled Binary option to generate a new HEX file in the project directory.

9.2.1.3 Bootloader VbbIO
The Bootloader mode make it possible to program the virtual Arduino as a real Arduino via the VbbIO hardware dongle. Refer to the VbbIO User Manual.

9.2.1.4 Arduino Sketch Source Application
Select the source from the list of Arduino Sketch projects in the solution. There can be more than one application so select the code to execute from the list.
<table>
<thead>
<tr>
<th>Properties</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Arduino SIM</td>
</tr>
<tr>
<td>Application</td>
<td>Source0.SRC</td>
</tr>
<tr>
<td>Clock</td>
<td>HEX File</td>
</tr>
<tr>
<td>Build Dir</td>
<td>Arduino IDE Monitor</td>
</tr>
<tr>
<td>ID</td>
<td>BootloaderVbblO</td>
</tr>
<tr>
<td>layout</td>
<td>Rescan Bootloader Devices</td>
</tr>
<tr>
<td>netlist</td>
<td>Blink--</td>
</tr>
<tr>
<td></td>
<td>Source0.SRC</td>
</tr>
<tr>
<td></td>
<td>Source1.SRC</td>
</tr>
</tbody>
</table>
11 IO Components

UserIO components don’t match directly to physical devices but they provide common functions useful for quickly modelling and testing a circuit.

11.1 DIP1, DIP4, DIP8

The DIP component is an N array of DIP switches which are connected to 5V when Switch = ON and 0V (GND) when Switch = OFF. DIPS are an interactive component. Clicking on the DIP switch, the white square region, toggles the value of the switch. The SETTINGS property is an N Binary string representing the values of each of the DIP switches. Clicking the respective DIP switch will change the value of SETTINGS, alternatively the value can be directly edited in the SETTINGS property textbox.

<table>
<thead>
<tr>
<th>Name</th>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DIP1</td>
<td><img src="image" alt="DIP1 Graphic" /></td>
<td><img src="image" alt="DIP1 Equivalent Circuit" /></td>
</tr>
<tr>
<td>DIP4</td>
<td><img src="image" alt="DIP4 Graphic" /></td>
<td><img src="image" alt="DIP4 Equivalent Circuit" /></td>
</tr>
<tr>
<td>DIP8</td>
<td><img src="image" alt="DIP8 Graphic" /></td>
<td><img src="image" alt="DIP8 Equivalent Circuit" /></td>
</tr>
</tbody>
</table>
11.1.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>OUTPUT[N]</td>
<td>DIP Output. Nth Pin is 5V if the Nth switch is ON. Nth Pin is 0V – GND if the Nth switch is OFF</td>
</tr>
</tbody>
</table>

11.1.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OnState</td>
<td>HIGH</td>
<td>HIGH</td>
<td>LOW</td>
</tr>
<tr>
<td>DIP</td>
<td>0</td>
<td>&lt;User Entry&gt;</td>
<td>The decimal value of the DIP. DIP1 0 or 1, DIP4 0 to 15, DIP4 0 to 255. You can type the value into the setting box or click the dip switches at designtime or runtime to change the value</td>
</tr>
<tr>
<td>Pins</td>
<td>Bottom</td>
<td>Top</td>
<td>Bottom</td>
</tr>
</tbody>
</table>

11.1.3 Usage

![Diagram of DIP switch usage]
11.2 LEDN

The LED component is an N array of Light Emitting Diode (LED) indicators. The Nth LED is ON when the Nth input pin is driven by greater than 2.5V otherwise the LED is OFF. The color of the LED is determined by the COLOR property. When the LED is ON the LED color is lighter than when OFF giving a visual clue to voltage level at the LED input pin. LEDs are a fundamental indicator of circuit status.

<table>
<thead>
<tr>
<th>Name</th>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>LED1</td>
<td><img src="image1" alt="LED1 Graphic" /></td>
<td><img src="image2" alt="LED1 Equivalent Circuit" /></td>
</tr>
<tr>
<td>LED4</td>
<td><img src="image3" alt="LED4 Graphic" /></td>
<td><img src="image4" alt="LED4 Equivalent Circuit" /></td>
</tr>
<tr>
<td>LED8</td>
<td><img src="image5" alt="LED8 Graphic" /></td>
<td><img src="image6" alt="LED8 Equivalent Circuit" /></td>
</tr>
</tbody>
</table>
11.2.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>INPUT[2N]</td>
<td>LED Input pins. Nth LED is ON when the Nth INPUT pin is &gt; 2.5V else is OFF</td>
</tr>
</tbody>
</table>

11.2.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>RED</td>
<td>RED</td>
<td>GREEN</td>
</tr>
<tr>
<td>Cathode</td>
<td>Ground</td>
<td>Ground</td>
<td>Pin</td>
</tr>
<tr>
<td>PinCount</td>
<td>1</td>
<td>N</td>
<td></td>
</tr>
</tbody>
</table>

11.2.3 Usage
11.3 DotMatrixLED8x8

The DotMatrixLED8x8 component is a Dot Matrix LED array. The array consists of 8 Rows and 8 Columns. Each Row Anode pin drives 8 LEDs each attached to one of the cathode column pins.

http://sigma.octopart.com/140413/datasheet/Lumex-LDM-24488NI.pdf

<table>
<thead>
<tr>
<th>Name</th>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DotMatrixLED8x8</td>
<td><img src="image" alt="VBB Graphic" /></td>
<td><img src="image" alt="Equivalent Circuit" /></td>
</tr>
</tbody>
</table>

By switching quickly between each row and using the persistence of vision effect each dot in the matrix can be individually addressed. To emulate the persistence of vision effect VBB only changes the state of the LED when COL is LOW. In this way the COL pin becomes like a latch operation which latches the value of LED.

<table>
<thead>
<tr>
<th>COL</th>
<th>ROW</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOW</td>
<td>HIGH</td>
<td>LED ON.</td>
</tr>
<tr>
<td>LOW</td>
<td>LOW</td>
<td>LED OFF</td>
</tr>
<tr>
<td>HIGH</td>
<td>Don’t Care</td>
<td>No Change.</td>
</tr>
</tbody>
</table>

11.3.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
</table>
### 11.3.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>COLOR</td>
<td>RED</td>
<td>RED</td>
<td>GREEN</td>
</tr>
</tbody>
</table>

### 11.4 Usage

![Usage Diagram](image-url)
### 11.5 JUMPN

The JUMPER component is an N array of switches which form a short circuit between the Nth input/output pin pair when the Nth Switch = ON and open-circuit O/C when Nth Switch = OFF. JUMPERS are an interactive component. Clicking on the JUMPER switch, the white square region, toggles the value of the JUMPER switch. The SETTINGS property is an N Binary string representing the values of each of the JUMPER switches. Clicking the respective JUMPER switch will change the value of SETTINGS. Alternatively the value can be directly edited in the SETTINGS property textbox.

<table>
<thead>
<tr>
<th>Name</th>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>JUMP1</td>
<td><img src="image1" alt="JUMP1 Diagram" /></td>
<td><img src="image2" alt="JUMP1 Equivalent Circuit" /></td>
</tr>
<tr>
<td>JUMP4</td>
<td><img src="image3" alt="JUMP4 Diagram" /></td>
<td><img src="image4" alt="JUMP4 Equivalent Circuit" /></td>
</tr>
<tr>
<td>JUMP8</td>
<td><img src="image5" alt="JUMP8 Diagram" /></td>
<td><img src="image6" alt="JUMP8 Equivalent Circuit" /></td>
</tr>
</tbody>
</table>
11.5.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td>IO[2N]</td>
<td>JUMPER IO Switch pain. Nth IO pair is short-circuit where Nth switch is ON, and open-circuit (O/C) when the Nth switch is OFF</td>
</tr>
</tbody>
</table>

11.5.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SETTINGS</td>
<td>[JUMP1] 1</td>
<td>&lt;User Entry&gt;</td>
<td>N Bit binary string holding JUMPER switches status. Is modified by clicking the component in-place or direct editing in the SETTINGs property textbox.</td>
</tr>
<tr>
<td>N</td>
<td>[JUMP4] 1111</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[JUMP8] 11111111</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11.5.3 Usage

![Diagram of Vbb4Arduino User Manual](image-url)
11.6 NumericKeyPad

The KEYPAD4x4 component consists of a Col x Row array of touch switches. The KEYPAD has column connect pins (C1,C2...Cols-1) and row pins (R1,R2...Rows-1). Each switch is connected to one column and one row connection pin such that each (row, column) combination is unique for each switch. The appearance of the KEYPAD component is determined by the KEYMASK property which contains a string of the key characters used on the key faces. The currently activated switch is determined by the KEYON property where KEYON = 0 when no switch is connected and KEYON = 1 to (Cols*Rows) when a valid switch is activated. KEYPAD is a user interactive component. Clicking on individual keys will activate the switch for that key. When active the background of the key becomes light green in color giving a visual clue to the state of the key. Only one switch can be active at any one time. Clicking on the currently active switch will toggle the switch to off resulting in no key being selected and KEYON = 0. When a key is on its switch becomes active creating a short-circuit across the unique row, column connection pin combination for that key.

<table>
<thead>
<tr>
<th>Name</th>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>NumericKeyPad</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Rows=4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cols = 4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

11.6.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1, C2,..CN</td>
<td>Column Connections</td>
<td></td>
</tr>
<tr>
<td>R1,R2..RN</td>
<td>Row Connections</td>
<td></td>
</tr>
</tbody>
</table>

11.6.2 Properties
### 11.6.3 Usage

The switch is connected by pressing a key joining a column and a row. Typically the user scans all columns and row combinations to detect the keypress.

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rows</td>
<td>4</td>
<td>&lt;User Entry&gt;</td>
<td>The number of Rows of the KeyPad</td>
</tr>
<tr>
<td>Cols</td>
<td>4</td>
<td>&lt;User Entry&gt;</td>
<td>The number of columns of the KeyPad</td>
</tr>
<tr>
<td>KeyMask</td>
<td>123F456E789D A0BC</td>
<td>&lt;User Entry&gt;</td>
<td>Col * Row character string containing the key-face character for each KEY</td>
</tr>
</tbody>
</table>
11.7 Seg7

The SEG7 component is a 7-Segment Display module, which consists of a versatile array of LEDs which can be sequenced to form numbers and a limited form of alphanumeric character. 8 LEDs A, B, C, D, E, F, G, PT are physically arranged on the 7-Segment device. A LED Segment is ON when 5V HIGH is applied to its corresponding input pin and is OFF when 0V LOW is applied. When the appropriate patterns are applied to the inputs, 7-Segment display modules are able to display numbers and ASCII characters.

<table>
<thead>
<tr>
<th>Name</th>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><img src="image" alt="VBB Graphic" /></td>
<td><img src="image" alt="Equivalent Circuit" /></td>
</tr>
</tbody>
</table>

11.7.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>A LED Segment Input. HIGH = ON, LOW = OFF</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>B LED Segment Input. HIGH = ON, LOW = OFF</td>
</tr>
<tr>
<td>3</td>
<td>C</td>
<td>C LED Segment Input. HIGH = ON, LOW = OFF</td>
</tr>
<tr>
<td>4</td>
<td>D</td>
<td>D LED Segment Input. HIGH = ON, LOW = OFF</td>
</tr>
<tr>
<td>5</td>
<td>E</td>
<td>E LED Segment Input. HIGH = ON, LOW = OFF</td>
</tr>
<tr>
<td>6</td>
<td>F</td>
<td>F LED Segment Input. HIGH = ON, LOW = OFF</td>
</tr>
<tr>
<td>7</td>
<td>G</td>
<td>G LED Segment Input. HIGH = ON, LOW = OFF</td>
</tr>
</tbody>
</table>
11.7.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>red</td>
<td>red</td>
<td>green</td>
</tr>
</tbody>
</table>

11.7.3 Usage

![LED Segment Diagrams]

![LED Segment Diagrams]
<table>
<thead>
<tr>
<th>Char</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>PT</th>
<th>HEX</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>66</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>B6</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>BE</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>E4</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>FE</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F6</td>
</tr>
<tr>
<td>A</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>EE</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3E</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9C</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>7A</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>9E</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>8E</td>
</tr>
<tr>
<td>G</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>BC</td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>6E</td>
</tr>
<tr>
<td>I</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>60</td>
</tr>
<tr>
<td>J</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>78</td>
</tr>
<tr>
<td>K</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>5E</td>
</tr>
<tr>
<td>L</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>06</td>
</tr>
<tr>
<td>M</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>M</td>
</tr>
<tr>
<td>N</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>N</td>
</tr>
<tr>
<td>O</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>O</td>
</tr>
<tr>
<td>P</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>P</td>
</tr>
<tr>
<td>Q</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>Q</td>
</tr>
<tr>
<td>R</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>S</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>S</td>
</tr>
<tr>
<td>T</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>8C</td>
</tr>
<tr>
<td>U</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>7C</td>
</tr>
<tr>
<td>V</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>7E</td>
</tr>
<tr>
<td>W</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>D8</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>07</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>----</td>
</tr>
<tr>
<td>Y</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>76</td>
</tr>
<tr>
<td>Z</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>92</td>
</tr>
<tr>
<td>.</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>01</td>
</tr>
<tr>
<td>+</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>62</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>92</td>
</tr>
<tr>
<td>(</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1A</td>
</tr>
<tr>
<td>)</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>32</td>
</tr>
<tr>
<td>_</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
</tr>
<tr>
<td>SPACE</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>00</td>
</tr>
</tbody>
</table>
11.8 DigitalPort

The DigitalPort Component is an instrument which displays the decimal value of the binary value represented by its input pins.

<table>
<thead>
<tr>
<th>Name</th>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>DigitalPort</td>
<td>![DigitalPort Diagram]</td>
<td></td>
</tr>
</tbody>
</table>

11.8.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.. PinCount</td>
<td>INPUT[PinCount]</td>
<td>Port input pins</td>
</tr>
</tbody>
</table>

11.9 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Endian</td>
<td>Big</td>
<td>Big</td>
<td>Little</td>
</tr>
<tr>
<td>PinCount</td>
<td>8</td>
<td>4</td>
<td>8</td>
</tr>
<tr>
<td>Pins</td>
<td>Bottom</td>
<td>Bottom</td>
<td>top</td>
</tr>
</tbody>
</table>

11.9.1 Usage
The Button component is an interactive switch. Clicking the color panel of the button toggles the value of the POSITION property. The contact type determines the circuit made when the POSITION is ON or OFF. The color of the button is determined by the COLOR property. When POSITION = ON the color is set by ColorOn property and when POSITION = OFF the color is set by the ColorOff property. When these colors are not the same the color can be used as a visual clue to the state of the button.

### 11.10 PushButton

<table>
<thead>
<tr>
<th>Name</th>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
11.10.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>B1</td>
<td>The contact pin for the button</td>
</tr>
</tbody>
</table>

11.10.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact Type</td>
<td>1. On=VDD, Off=GND</td>
<td>1. On=VDD, Off=GND</td>
<td>The contact Type</td>
</tr>
<tr>
<td>--------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>------------------</td>
</tr>
<tr>
<td></td>
<td>2. On=GND, Off=VDD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. Make VDD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. Make GND</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. Break VDD</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. Break GND</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Button</th>
<th>1. circle</th>
<th>1. circle</th>
<th>The graphic of the button.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2. square</td>
<td>2. square</td>
<td></td>
</tr>
<tr>
<td></td>
<td>3. left arrow</td>
<td>3. left arrow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4. right arrow</td>
<td>4. right arrow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5. down arrow</td>
<td>5. down arrow</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6. up arrow</td>
<td>6. up arrow</td>
<td></td>
</tr>
</tbody>
</table>

| ColorOn      | blue      | Red|green|blue|yellow|NamedColor | Sets the graphic color of the On button. Can be a named color in addition to the option colors |
|--------------|-----------|-----------------------|-----------------------------|
|              |           |                       |                             |

| ColorOff     | black     | Red|green|blue|yellow|NamedColor | Sets the graphic color of the On button. Can be a named color in addition to the option colors |
|--------------|-----------|-----------------------|-----------------------------|
|              |           |                       |                             |

### 11.11 Usage

Not only can you change the size of the buttons using the Scale Up 📊.
### 11.12 Switch

The Switch component is an interactive switch. Clicking the switch image toggles the value of the POSITION property. When POSITION = ON a short circuit is present across the IO pins. When POSITION = OFF there is an open-circuit across the IO pins. To give a visual clue to the state of the switch, when POSITION = ON the switch lever in the ON Position or the Bitmap On is shown and when POSITION = OFF the switch lever in the OFF position or Bitmap Off is shown.

<table>
<thead>
<tr>
<th>Name</th>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ToggleSwitch</td>
<td><img src="image" alt="Switch Diagram" /></td>
<td><img src="image" alt="Toggle Switch Diagram" /></td>
</tr>
</tbody>
</table>

#### 11.12.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IO</td>
<td>Switch input/output pin</td>
</tr>
<tr>
<td>2</td>
<td>IO</td>
<td>Switch input/output pin</td>
</tr>
</tbody>
</table>

#### 11.12.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Switch</td>
<td>1. Off</td>
<td>1. Off</td>
<td>2. On</td>
</tr>
<tr>
<td>Bitmap On</td>
<td></td>
<td>FileDialog for Bitmap .png</td>
<td>.bmp</td>
</tr>
<tr>
<td>Bitmap Off</td>
<td></td>
<td>FileDialog for Bitmap .png</td>
<td>.bmp</td>
</tr>
</tbody>
</table>
11.13 PanelMeter

The PanelMeter is a Voltage Meter Instrument with a dial representation of the voltage on its input pin between Vss (left) and Vdd (right). The Panel Meter has the equivalent of a LOW PASS filter on the input pin so can display voltage levels for both absolute voltage levels and PWM equivalent analog voltages.

<table>
<thead>
<tr>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="VBB Graphic" /></td>
<td><img src="image2.png" alt="Equivalent Circuit" /></td>
</tr>
</tbody>
</table>

11.13.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>The voltage sense input pin</td>
</tr>
</tbody>
</table>

11.13.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shape</td>
<td>Square</td>
<td>Square</td>
<td>Round</td>
</tr>
</tbody>
</table>
11.13.3 Usage
11.14 SlidePot

The SlidePOT (potentiometer) is a variable voltage divider and consists of two inputs and a single output. The OUTPUT voltage is determined by the input voltages and the value of the POSITION property where OUTPUT = (POSITION x VINA) + ((1 – POSITION) x VINB). Where Property POSITION has a valid range from range from [0..1]. POT is an interactive component. Clicking on POT will set the POSITION property. Clicking on the far left of the POT component corresponds to position 0. Clicking on the far right of the POT component corresponds to position 1. Clicking in-between the left and right will set the POT to a position linearly between 0,1. The POSITION property can also be set using the property editor. You can also drag the POT Knob with the mouse to continuously change the value.

<table>
<thead>
<tr>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="VBB Graphic" /></td>
<td><img src="image" alt="Equivalent Circuit" /></td>
</tr>
</tbody>
</table>

11.14.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>INA</td>
<td>Input voltage reference</td>
</tr>
<tr>
<td>2</td>
<td>INB</td>
<td>Input voltage reference</td>
</tr>
<tr>
<td>3</td>
<td>OUTPUT</td>
<td>Variable output voltage</td>
</tr>
</tbody>
</table>

11.14.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pot</td>
<td>0</td>
<td>0</td>
<td>25</td>
</tr>
</tbody>
</table>

11.14.3 Usage

![Usage Image]
11.15 RotaryPot

The RotaryPot (potentiometer) is a variable voltage divider connected between Vdd and Vss with a single output. The OUTPUT voltage is determined by the rotation position where OUTPUT = Vdd x (POSITION / Positions). To rotate the POT you use the mouse to drag the rotation dial.

<table>
<thead>
<tr>
<th>VBB Graphic</th>
<th>Equivalent Circuit</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="VBB Graphic" /></td>
<td>POT</td>
</tr>
</tbody>
</table>

11.15.1 Pinout

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OUTPUT</td>
<td>Variable output voltage</td>
</tr>
</tbody>
</table>

11.15.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Positions</td>
<td>12</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>Position</td>
<td></td>
<td></td>
<td>Potentiometer setting between 0,Positions-1 which determines the OUTPUT = Vdd x (POSITION / Positions)</td>
</tr>
</tbody>
</table>

11.16 Usage
### 11.17 JoyStick

The JoyStick component is a model of a regular PC JoyStick with 2 potentiometers on the X and Y axis and button with a pullup resistor. Move the mouse over the keypad to move the JoyStick in X, Y and click the mouse to press the button pulling the button output to ground.

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>X</td>
<td>X-Axis output Vcc (left) through Vdd (right)</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>Pulled up button. House mouse down to pull down</td>
</tr>
<tr>
<td>3</td>
<td>Y</td>
<td>Y-Axis output Vcc (top) through Vdd (bottom)</td>
</tr>
</tbody>
</table>

#### 11.17.1 Pinout

**None**

#### 11.17.2 Properties

**None**
11.17.3 Usage

The mini terminal is a RS232 terminal which can send a receive asynchronous UART communications at TTL levels ie Vdd to Vcc at BAUD. Click on the Green screen segment and type to send messages. When pressing ENTER the characters set in the Enter property are sent. Communications is standard 8,N,1

11.18 MiniTerminal

The mini terminal is a RS232 terminal which can send a receive asynchronous UART communications at TTL levels ie Vdd to Vcc at BAUD. Click on the Green screen segment and type to send messages. When pressing ENTER the characters set in the Enter property are sent. Communications is standard 8,N,1

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RX(In)</td>
<td>The input into which the communications is received</td>
</tr>
</tbody>
</table>
## 11.18.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAUD</td>
<td>9600</td>
<td>2400</td>
<td>9600</td>
</tr>
<tr>
<td>Enter</td>
<td>LF(13)</td>
<td>CR(10)</td>
<td>LF(13)</td>
</tr>
</tbody>
</table>

## 11.18.3 Usage

![Diagram of TX(Out)](image)

The output from which communications is sent.
12 Breadboardable Components

Breadboardable components are components that have both virtualization functionality and match up with real components so when you layout a circuit with Breadboardable components it becomes a makeable circuit.

12.1 LiquidCrystal

The LiquidCrystal component is a model of a 2 line HD44780 Liquid Crystal Display. Supports HD44780 emulation in 4-bit mode.

Fonts: If available the HD44780 font is used else the default is used.

Limitations

- 4 Bit mode only supported
- Custom Bitmaps not supported
- R/W read not support

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Enable</td>
<td>Variable output voltage</td>
</tr>
<tr>
<td>2</td>
<td>Register Select Pin</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Read/Write Pin – Not functional</td>
<td></td>
</tr>
<tr>
<td>4-11</td>
<td>D0-7</td>
<td>Data Pins. Only 4 bit mode using D4-D7 is used</td>
</tr>
</tbody>
</table>

12.1.1 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Display Cols</td>
<td>16</td>
<td>8</td>
<td>10</td>
</tr>
</tbody>
</table>
12.1.1 Usage

12.2 LDR – Light Dependent Resistor

12.2.1 Description
The Light Dependent Resistor, (LDR), is a useful photonic sensor that converts light into a difference in resistance. Of course in a virtual environment you can’t actually change the light so a way is needed to vary the resistance. This is done by the LDR slider widget which you can slide from left to right to model the way light is varied.

Normally a LDR forms a voltage divider and the variable resistance causes the voltage at the voltage divider tap to vary. The LDR models this by being a variable voltage source in the range MINVOLT to MAXVOLT as set by the properties.

12.2.2 Properties

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Options</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MINVOLT</td>
<td>1</td>
<td>&lt;User Entry&gt;</td>
<td>Sets the value of the minimum voltage when the slider is all the way to the left</td>
</tr>
</tbody>
</table>
### 12.2.3 Usage

A voltage divider is formed with the resistor. By dragging the LDR widget different light conditions are simulated resulting in different output voltage levels.

| MAXVOLT | 5 | <User Entry> 1 to 5 | Sets the value of the maximum voltage when the slider is all the way to the right |

---

#### 12.1 Relay

The relay is a common configuration for a solenoid relay. A relay uses a smaller powered solenoid to mechanically switch a high current circuit. Relays are useful for driving higher voltage and current systems. When the solenoid circuit is powered the switch changes position connecting the alternate path.
12.1.1  Pins

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Switched A</td>
<td>Switched A connects to common when solenoid circuit is NOT powered</td>
</tr>
<tr>
<td>2</td>
<td>Switched B</td>
<td>Switched A connects to common when solenoid circuit IS powered</td>
</tr>
<tr>
<td>3</td>
<td>Common</td>
<td>The common pin</td>
</tr>
<tr>
<td>4</td>
<td>Solenoid A</td>
<td>Solenoid is powered when a current flows between Solenoid A and Solenoid B</td>
</tr>
<tr>
<td>5</td>
<td>Solenoid B</td>
<td>Solenoid is powered when a current flows between Solenoid A and Solenoid B</td>
</tr>
</tbody>
</table>

12.1.2  Properties
Common Properties only

Power Off

Power On
12.1.3 Usage
Relays are used to drive larger loads, typically mechanical loads such as larger solenoids. In this example you can see two solenoids are driven by a 12V supply to ground via the Relay. You can also visualise the action of the relay with the small switch internal to the relay.

12.2 Solenoid
The solenoid is a mechanical actuator that when a sufficient current passes from PWR to GND pint the centre rod actuates. Solenoids are useful for many different mechanisms such as locking.
12.2.1 Pins

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Ground</td>
<td>Switched A connects to common when solenoid circuit is NOT powered</td>
</tr>
<tr>
<td>2</td>
<td>Power</td>
<td>Switched A connects to common when solenoid circuit IS powered</td>
</tr>
</tbody>
</table>

Properties
Common Properties only

12.2.2 Usage
The solenoid actives when it is driven otherwise it is in the retracted position
12.1 Trimmer

The trimmer is a voltage divider potentiometer in the common 3 leg ‘tripod’ configuration with an adjustable centre. Dragging the trimmer dial changes the position of the center tap and hence changes the value of the resistor proportions of the voltage divider changing the output voltage proportionally to the rotation position.

The position of the dial is persisted when saved and restored on open and run.
### 12.1.1 Pins

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Variable resistor pin</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>Variable resistor pin</td>
</tr>
<tr>
<td>3</td>
<td>Tap</td>
<td>The voltage at TAP is $V_1 = (V_a - V_b) \cdot \text{position} / \text{MaxPosition}$</td>
</tr>
</tbody>
</table>

### 12.1.2 Usage

The potentiometer is a voltage divider potentiometer in the common 3 inline pins configuration with an adjustable centre. Dragging the dial changes the position of the centre tap and hence changes the value of the resistor proportions of the voltage divider changing the output voltage proportionally to the rotation position.

The position of the dial is persisted when saved and restored on open and run.

### 12.2 Potentiometer

The potentiometer is a voltage divider potentiometer in the common 3 inline pins configuration with an adjustable centre. Dragging the dial changes the position of the centre tap and hence changes the value of the resistor proportions of the voltage divider changing the output voltage proportionally to the rotation position.

The position of the dial is persisted when saved and restored on open and run.
12.2.1 Pins

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>Variable resistor pin</td>
</tr>
<tr>
<td>2</td>
<td>B</td>
<td>Variable resistor pin</td>
</tr>
<tr>
<td>3</td>
<td>Tap</td>
<td>The voltage at TAP is $V_1 = (V_a - V_b) \times \text{position} / \text{MaxPosition}$</td>
</tr>
</tbody>
</table>

12.2.2 Usage

![Diagram showing usage of pins]

Drag Dial to rotate
12.3 Segment7

The Segment7 is a Breadboardable version of the Segment7 from the user IO segment 7

12.3.1 Pins

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>e</td>
<td>Variable resistor pin</td>
</tr>
<tr>
<td>2</td>
<td>d</td>
<td>Variable resistor pin</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>a,d,c,pt common cathode</td>
</tr>
<tr>
<td>4</td>
<td>c</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>pt</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>B</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>a</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td>a,d,c,pt common cathode</td>
</tr>
<tr>
<td>9</td>
<td>F</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>g</td>
<td></td>
</tr>
</tbody>
</table>
12.3.2 Usage

Setting the common cathode pins to ground you can create different numbers by powering the individual segment pins.

12.4 Buzzer

The Buzzer component is an audio component that converts frequencies into audio. Commonly used with tone() or a 555 square wave source.

12.4.1 Pins

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
12.4.2 Usage
When used with a frequency source changing the frequency will change the audio tone heard for feedback.

12.5 Sharp Proximity Sensor
The sharp proximity sensor is a popular robotics sensor for detecting how close an object is. It is easy to use as it generates a voltage which changes according to how close the sensor is.

12.5.1 Pins

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>+</td>
<td>Power pin, connect to frequency source</td>
</tr>
<tr>
<td>2</td>
<td>GND</td>
<td>ground</td>
</tr>
</tbody>
</table>
12.5.2 Usage

By dragging the widget the sensor simulates proximity which can be visualized by the changing size of the tree which represents how close or far the sensor is from the tree.

Plotted over time you can generate signals that a robot might see as it approaches and moves away from a wall.
555 Timer IC

The 555 is a versatile chip used in a huge variety of timing, pulse generating and oscillator applications. Although modern microcontrollers have provided alternative ways to implement some of these type of circuits the 555 remains an invaluable part of any tinkerers toolbox.

There are two basic circuit configurations used by the 555 to create distinct functions

**AStable** - in this mode the 555 operates as an oscillator. Uses include LED and lamp flashers, pulse generation, logic clocks, tone generation and so on.

**Monostable** - in this mode the 555 operates as a triggered pulse generator. Uses include include timers, pulse detection, debouncing, pulse generation and so on.
12.6 Astable

In astable mode, the 555 timer puts out a continuous stream of rectangular pulses having a specified frequency.

The frequency depends on the values of R1, R2 and C in the above schematic.

There are plenty of articles covering the theory of the AStable 555 operation ([WIKIPEDIA](https://en.wikipedia.org/wiki/555_timer) is a good place to start) so I am not going to reproduce that here. Instead I am going to show you how to use VirtualBreadboard (VBB) to calculate these values for you and show you how to view the resulting output signal.

If you zoom into the VBB 555 component you will see the pins are labelled with the schematic labels. So the Breadboard is a hybrid schematic and Breadboard layout. This makes it easier to wire up a schematic as a Breadboard layout.
TIP: You can use 'Import from Clipboard' to Load this project directly into VBB. See 'ASTABLE PROJECT' in the Source Code section below.

12.7 ASTABLE Frequency Calculator

The VBB 555 component has several properties which can be used to determine the emulation behavior of the component:

- C
- R1-STABLE
- R2-ASTABLE
- C-MONOSTABLE
- Calculator

To make the 555 generate emulate a astable frequency generator you need to set the values of C,R1,R2 to match the component ID values of the corresponding circuit elements. This will set the value of the calculate property to ASTABLE(..).
12.8 Calculating the values for AStable

At runtime VBB uses the values for C,R1,R2 are used to calculate the frequency and duty of the 555. You can set the values manually yourself for example if are wiring up and existing design. Alternatively you can use the Calculator properties custom editor to calculate and the set the values for you.

To use the calculator

- Click the calculate property and then click the edit button to open the custom calculator dialog
- Enter the target frequency or period and click calculate
- Select your preferred C,R1,R2 values and click OK
12.9 Viewing the AStable Output

VBB emulations are 'real time' so when you 'Power Up' the circuit you can get a rough idea of the output frequency of the 555 just by looking at a flashing L.E.D if you attach one to the output pin.

This visual check is only useful to about 10 Hz after which the easiest way is to attach a logic probe and view the output waveform in the Logic Analyser. You can change the timebase to the period of the waveform best fits the grid lines making it a simple calculation to find the period.
12.10  Monostable

In monostable mode, the 555 timer puts out a triggered 'one shot' pulse.

The duration of the pulse depends on the value of R and C in the above schematic.
**TIP**: You can use 'Import from Clipboard' to Load this project directly into VBB. See 'MONOSTABLE PROJECT' in the Source Code section below.

### 12.11 Calculating the values for Monostable

To make the 555 generate emulate a monostable pulse generator you need to set the values of C and R to match the component ID values of the corresponding circuit elements. This will set the value of the Calculate property as MONO(·)

To use the calculator

- Click the calculate property and then click the edit button to open the custom calculator dialog
- Enter the target pulse period and click calculate
- Select your preferred C and R values and click OK.
12.12 Viewing the Monostable Output

You can interact with the virtual circuit. To generate the trigger input click on the DIP switch to create a **falling edge** at the 555 trigger input pin. You should see the L.E.D attached to the output power on for a fixed period. You can also capture the pulse length using the Logic-analyser to get a more accurate view of the pulse width.
13 API ‘Smart’ Components

There are two types of components. Signal level components and API components. A signal level component such as Liquid Crystal is modelled based on the signals it receives at its pins. This requires implementing the underlying timing specification of the component.

API components are components that only virtualize when driven via the Java API. The wiring of the component only links the component to the microcontroller. The actual communication to the device is done in an Abstract communications channel. This makes it easier to implement the component virtualization but requires the use of the API to make it work.

13.1 WS2812

The WS2812 is a ‘smart’ 3 color LED. It can be ‘daisy chained’ by connecting the output of a module to the input of the next module.
### 13.2 API

**JavaDoc Online**

---

#### NEO PIXEL Constants

```java
public const int NEO_RGB = 0x00; // Wired for RGB data order
public const int NEO_GRB = 0x01; // Wired for GRB data order
public const int NEO_BRG = 0x04;

public const int NEO_COLMASK = 0x01;
public const int NEO_KHZ800 = 0x02; // 800 KHz datastream
public const int NEO_SPDMASK = 0x02;
```

---

#### Usage

1. Place a WS2812
2. Switch to Module footprint
3. Wire the input pin to an output pin, 12
4. Include the API:
5. Instantiate and call
The DS1307 serial real-time clock (RTC) is a lowpower, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially through an I2C, bidirectional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12hour format with AM/PM indicator. The DS1307 has a built-in power-sense circuit that detects power failures and automatically switches to the backup supply. Timekeeping operation continues while the part operates from the backup supply.

13.3 DS1307
64 x 8, Serial, I2C Real-Time Clock
13.3.1 API

Wire

In VBB Wire is an API. The I2C signals are not transmitted over the actual wire but are handled abstractly.

```java
import mvium.compatibility.arduino.*;
import mvium.compatibility.arduino.Wire;
import mvium.compatibility.arduino.StringUtils;

class TestDS1307 extends Arduino{

    private static final int DS1307_ADDRESS = 0x68;
    private static final int POS_DAY = 0;
    private static final int POS_MONTH = 3;
    private static final int POS_YEAR = 8;
    private static final int POS_HOUR = 11;
    private static final int POS_MINUTE = 14;
    private static final int POS_SECOND = 17;

    char timeFormat[] = StringUtils.getChars("dd/mm/yyyy
    hh:MM:ss\r\n");

    // The setup() method runs once, when the sketch starts
    public void setup(){
        Wire.begin(); // initialise the connection
        Serial.begin(9600);

        Serial.write(StringUtils.castChars(timeFormat),
        sizeof(timeFormat));

        timeFormat[POS_YEAR - 2] = '2';
        timeFormat[POS_YEAR - 1] = '0';

        TimeNow();
        Serial.write(StringUtils.castChars(timeFormat),
        sizeof(timeFormat));
        UpdateTime();
        TimeNow();
    }
```
private void TimeNow(){
    Wire.beginTransmission(DS1307_ADDRESS);
    Wire.write(0);
    Wire.endTransmission();

    Wire.requestFrom(DS1307_ADDRESS, 7);
    writeBCD( Wire.read() & 0x7F, POS_SECOND);
    writeBCD( Wire.read(), POS_MINUTE);
    writeBCD( Wire.read(), POS_HOUR);
    Wire.read();
    writeBCD( Wire.read(), POS_DAY);
    writeBCD( Wire.read(), POS_MONTH);
    writeBCD( Wire.read(), POS_YEAR);
}

private void UpdateTime(){
    Wire.beginTransmission(DS1307_ADDRESS);
    Wire.write(0);
    Wire.write(0); // Seconds
    Wire.write(1); // Minute
    Wire.write(2); // Hour
    Wire.write(0); // Day of Month
    Wire.write(3); // Day
    Wire.write(4); // Month
    Wire.write(5); // Year (2005)
    Wire.write(0); // Control..
    Wire.endTransmission();
}

private void writeBCD(int val, int pos){
    timeFormat[pos++] = (char) ( '0' + ( val >> 4 ));
    timeFormat[pos++] = (char) ( '0' + ( val & 0xF ) );
}

// the loop() method runs over and over again,
// as long as the Arduino has power
public void loop(){
    TimeNow();
    Serial.write(StringUtils.castChars(timeFormat), sizeof(timeFormat));
    delay(2000);
}
}
13.4 24LC256 EEPROM

The 24LC256 is a 256Kb (32K x 8) Serial Electrically Erasable PROM (EEPROM),

13.4.1 API

Wire

In VBB Wire is an API. The I2C signals are not transmitted over the actual wire but are handled abstractly.

```java
import muvium.compatibility.arduino.*;
import muvium.compatibility.arduino.Wire;
import muvium.compatibility.arduino.StringUtils;

public class TestEEPROM extends Arduino{

    void i2c_eeprom_write_byte( int deviceaddress, int eeaddress, int data ) {
        Wire.beginTransmission(deviceaddress);
        Wire.write((int) (eeaddress >> 8)); // MSB
        Wire.write((int) (eeaddress & 0xFF)); // LSB
        Wire.write(data);
        Wire.endTransmission();
    }

    // WARNING: address is a page address, 6-bit end will wrap around
    // also, data can be maximum of about 30 bytes, because the Wire library has a buffer of 32 bytes
    void i2c_eeprom_write_page( int deviceaddress, int eeaddresspage, byte[] data, int length ) {
        Wire.beginTransmission(deviceaddress);
        Wire.write((int) (eeaddresspage >> 8)); // MSB
        Wire.write((int) (eeaddresspage & 0xFF)); // LSB

        for (int c = 0; c < length; c++)
            Wire.write(data[c]);
        Wire.endTransmission();
    }

}
byte i2c_eeprom_read_byte(int deviceaddress, int eeaddress) {
    byte rdata = (byte) 0xFF;
    Wire.beginTransmission(deviceaddress);
    Wire.write((int) (eeaddress >> 8)); // MSB
    Wire.write((int) (eeaddress & 0xFF)); // LSB
    Wire.endTransmission();
    Wire.requestFrom(deviceaddress, 1);
    if (Wire.available() != 0) rdata = (byte) Wire.read();
    return rdata;
}

// maybe let's not read more than 30 or 32 bytes at a time!
void i2c_eeprom_read_buffer(int deviceaddress, int eeaddress, byte[] buffer, int length) {
    Wire.beginTransmission(deviceaddress);
    Wire.write((int) (eeaddress >> 8)); // MSB
    Wire.write((int) (eeaddress & 0xFF)); // LSB
    Wire.endTransmission();
    Wire.requestFrom(deviceaddress, length);
    int c = 0;
    for (c = 0; c < length; c++)
        if (Wire.available() != 0) buffer[c] = (byte) Wire.read();
}

// The setup() method runs once, when the sketch starts
public void setup() {

    // Your setup code goes here
    char eeprom1[] = StringUtils.getChars("this is data from the first eeprom"); // data to write
    char eeprom2[] = StringUtils.getChars("this is data from the second eeprom"); // data to write

    Wire.begin(); // initialise the connection
    Serial.begin(9600);
    i2c_eeprom_write_page(0x50, 0, StringUtils.castChars(eeprom1), sizeof(eeprom1)); // write to EEPROM
    i2c_eeprom_write_page(0x51, 0, StringUtils.castChars(eeprom2), sizeof(eeprom2)); // write to EEPROM

    delay(10); //add a small delay
    Serial.println("Memory written");

}

private void readOutEEPROM(int address) {
    int addr = 0; //first address
    byte b = i2c_eeprom_read_byte(address, 0); // access the first address from the memory

    while (b != 0) {
        Serial.print((char) b); //print content to serial port
        addr++; //increase address
    }
}
b = i2c_eeprom_read_byte(address, addr); // access an address from the memory
}

Serial.println(" ");

// the loop() method runs over and over again, // as long as the Arduino has power
public void loop()
{

readOutEEPROM(0x50);
readOutEEPROM(0x51);

delay(2000);

}

14 Appendix Named Colors
You can enter named colors into any property that requires color.

"background" RGB(204,204,204)
"vbb" RGB(164,198,57)
"darkvbb" RGB(120,154,13)
"comment" RGB(255,255,128)
"arduino" RGB(0,152,158)
"darkarduino" RGB(0,120,125)
"aliceblue" RGB(240,248,255)
"antiquewhite" RGB(250,235,215)
"aqua" RGB(0,255,255)
"aquamarine" RGB(127,255,212)
"azure" RGB(240,255,255)
"beige" RGB(245,245,220)
"bisque" RGB(255,228,196)
"black" RGB(0,0,0)
"blanchedalmond" RGB(255,235,205)
"blue" RGB(0,0,255)
"blueviolet" RGB(138,43,226)
"brown" RGB(165,42,42)
"burlywood" RGB(222,184,135)
"cadetblue" RGB(95,158,160)
"chartreuse" RGB(127,255,0)
"chocolate" RGB(210,105,30)
"coral" RGB(255,127,80)
"cornflowerblue" RGB(100,149,237)
"cornsilk" RGB(255,248,220)
"crimson" RGB(220,20,60)
"cyan" RGB(0,255,255)
"darkblue" RGB(0,0,139)
"darkcyan" RGB(0,139,139)
"darkgoldenrod" RGB(184,134,11)
"darkgray" RGB(169,169,169)
"darkgreen" RGB(0,100,0)
"darkgrey" RGB(169,169,169)
"darkkhaki" RGB(189,183,107)
"darkmagenta" RGB(139,0,139)
"darkolivegreen" RGB(85,107,47)
"darkerorange" RGB(255,140,0)
"darkorchid" RGB(153,50,204)
"darkred" RGB(96,0,0) getSVGColor=RGB(139,0,0)
"darksalmon" RGB(233,150,122)
"lightpink" RGB(255,182,193)
"lightsalmon" RGB(255,160,122)
"lightseagreen" RGB(32,178,170)
"lightskyblue" RGB(135,206,250)
"lightslategray" RGB(119,136,133)
"lightslategrey" RGB(119,136,133)
"lightsteelblue" RGB(176,196,222)
"lightyellow" RGB(255,255,224)
"lime" RGB(0,255,0)
"limegreen" RGB(50,205,50)
"linen" RGB(250,240,230)
"magenta" RGB(255,0,255)
"maroon" RGB(128,0,0)
"mediumaquamarine" RGB(102,205,170)
"mediumblue" RGB(0,0,205)
"mediumorchid" RGB(186,85,211)
"mediumpurple" RGB(147,112,219)
"mediumseagreen" RGB(60,179,113)
"mediumslateblue" RGB(123,104,238)
"mediumspringgreen" RGB(0,250,154)
"mediumturquoise" RGB(72,209,204)
"mediumvioletred" RGB(199,21,133)
"midnightblue" RGB(25,25,112)
"mintcream" RGB(245,255,250)
"mistyrose" RGB(255,228,225)
"moccasin" RGB(255,228,181)
"navajowhite" RGB(255,222,173)
"navy" RGB(0,0,128)
"oldlace" RGB(253,245,230)
"olive" RGB(128,128,0)
"olivedrab" RGB(107,142,35)
"orange" RGB(255,165,0)
"orangered" RGB(255,69,0)
"lightcoral" RGB(240,128,128)
"lightcyan" RGB(224,255,255)
"lightgoldenrodyellow" RGB(250,250,210)
"lightgray" RGB(211,211,211)
"lightgreen" RGB(144,238,144)
"lightgrey" RGB(211,211,211)
"violet" RGB(238,130,238)
"wheat" RGB(245,222,179)
"white" RGB(255,255,255)
"whitesmoke" RGB(245,245,245)
"yellow" RGB(255,255,0)
"yellowgreen" RGB(154,205,50)
"darkseagreen" RGB(143,188,143)
"darkslateblue" RGB(72,61,139)
"darkslategray" RGB(47,79,79)
"darkturquoise" RGB(0,206,209)
"darkviolet" RGB(148,0,211)
"deeppink" RGB(255,20,147)
"deepskyblue" RGB(0,191,255)
"dimgray" RGB(105,105,105)
"dimgrey" RGB(105,105,105)
"dodgerblue" RGB(30,144,255)
"firebrick" RGB(178,34,34)
"floralwhite" RGB(255,250,240)
"forestgreen" RGB(34,139,34)
"fuchsia" RGB(255,0,255)
"gainsboro" RGB(220,220,220)
"ghostwhite" RGB(248,248,255)
"gold" RGB(255,215,0)
"goldenrod" RGB(218,165,32)
"gray" RGB(128,128,128)
"grey" RGB(128,128,128)
"green" RGB(0,128,0)
"greenyellow" RGB(173,255,47)
"honeydew" RGB(240,255,240)
"hotpink" RGB(255,105,180)
"indianred" RGB(205,92,92)
"indigo" RGB(75,0,130)
"ivory" RGB(255,255,240)
"khaki" RGB(240,230,140)
"lavender" RGB(230,230,250)
"lavenderblush" RGB(255,240,245)
"lawngreen" RGB(124,252,0)
"lemonchiffon" RGB(255,250,205)
"lightblue" RGB(173,216,230)
"orchid" RGB(218,112,214)
"palegoldenrod" RGB(238,232,170)
"palegreen" RGB(152,251,152)
"paleturquoise" RGB(175,238,238)
"palevioletred" RGB(219,112,147)
"papayawhip" RGB(255,239,213)
"peachpuff" RGB(255,218,185)
"peru" RGB(205,133,63)
"pink" RGB(255,192,203)
"plum" RGB(221,160,221)
"powderblue" RGB(176,224,230)
"purple" RGB(128,0,128)
"red" RGB(255,0,0)
"rosybrown" RGB(188,143,143)
"royalblue" RGB(65,105,225)
"saddlebrown" RGB(139,69,19)
"salmon" RGB(250,128,114)
"sandybrown" RGB(244,164,96)
"seagreen" RGB(46,139,87)
"seashell" RGB(255,245,238)
"sienna" RGB(160,82,45)
"silver" RGB(192,192,192)
"skyblue" RGB(135,206,235)
"slateblue" RGB(106,90,205)
"slategray" RGB(112,128,144)
"slategrey" RGB(112,128,144)
"snow" RGB(255,250,250)
"springgreen" RGB(0,255,127)
"steelblue" RGB(70,130,180)
"Tan" RGB(210,180,140)
"teal" RGB(0,128,128)
"thistle" RGB(216,191,216)
"tomato" RGB(255,99,71)
"turquoise" RGB(64,224,208)