Breezy4Pi Examples

Revision 0.80

Introduction

This document shall give a few easy examples of how to run your first macros on your Raspberry Pi using Breezy4Pi.

Please refer to the, "Setup and Installation of Breezy4Pi on your Raspberry Pi", and to the "User's Manual" when working through these examples.

This document assumes that you have correctly deployed Breezy4Pi onto the Tomcat webserver and that the application boots up and runs and that you can connect to it with your web browser.

Setup

Unzip the accompanying Examples.zip file and copy the entire directory structure to root folder on your Raspberry Pi. Your file structure should look like:

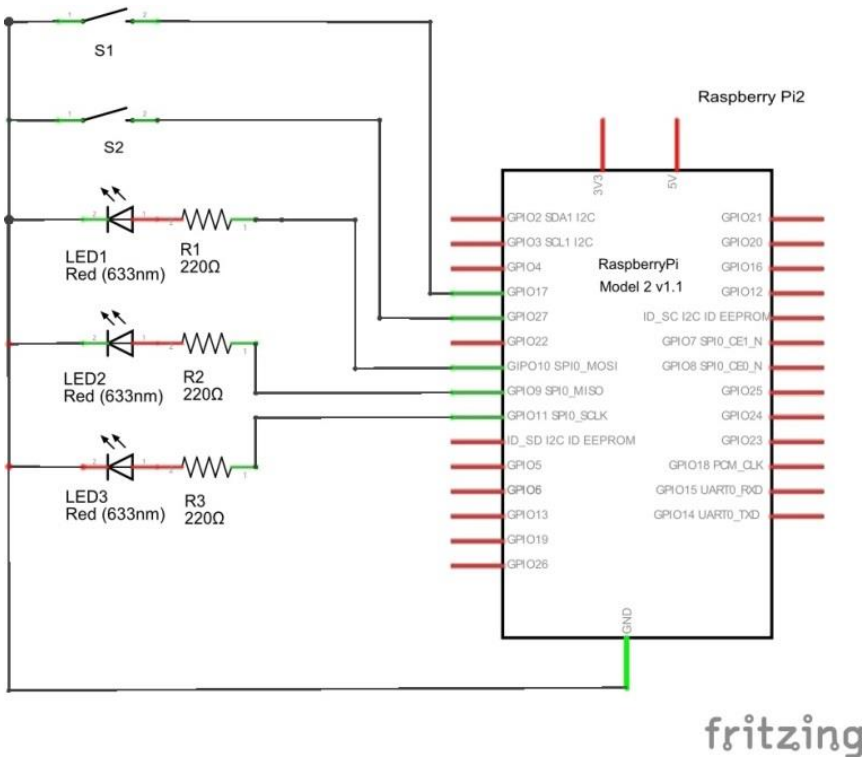
- root
  - Breezy
    - Logs
    - Resources
      - Boards
        - Files of *.board and *.boardtemplate extensions
      - Events
        - Files of *.events extensions
      - Macros
        - Files of *.macro extensions

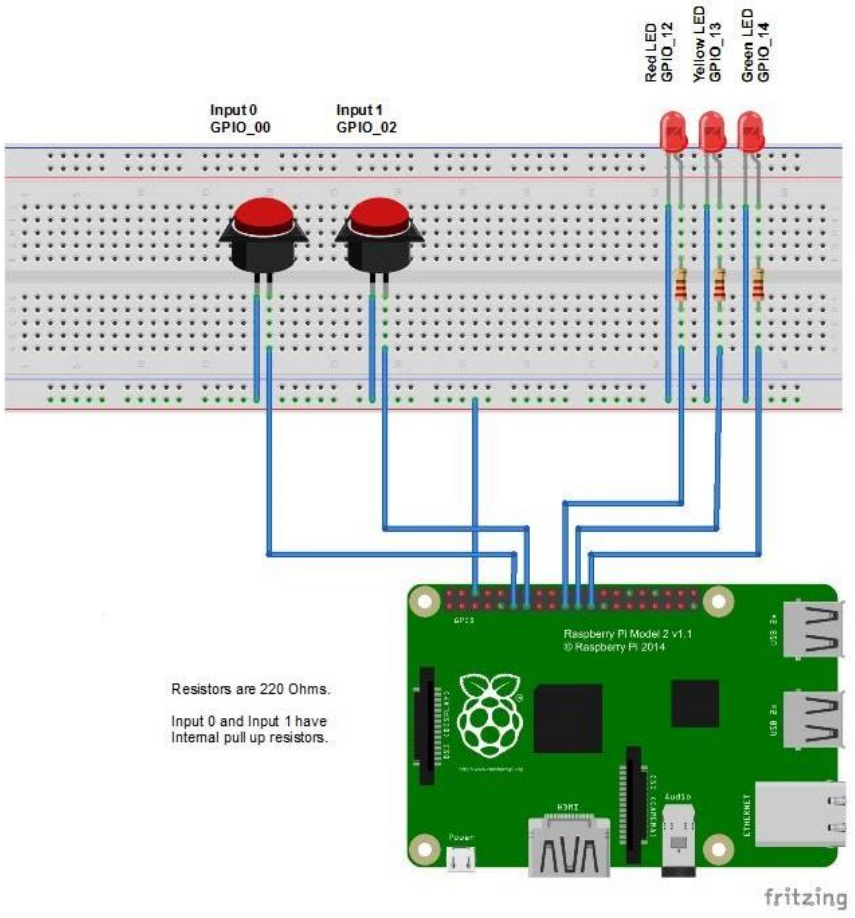
All filenames are UUIDs (universally unique identifiers) and are the primary keys for objects in Breezy4Pi. This is why, in the various editors, you will see something like, "7bf7744e-99fd-4476-9f34-992fda3ded8e.macro". This is provided in case you ever wish to examine the contents of a particular file.

These examples require the following:

1. Three LEDs, preferably one each of green, yellow, and red.
2. Three 220 ohm resistors.
3. Two momentary push buttons.

Please refer to the below schematic and board drawing for wiring instructions. The schematic uses a different GPIO numbering scheme than the board drawing. The schematic uses the pinout found on the Raspberry Pi website while the drawing pinout is the one used by Wiring Pi and Pi4J. Breezy4Pi uses the Wiring Pi and Pi4J pinouts.

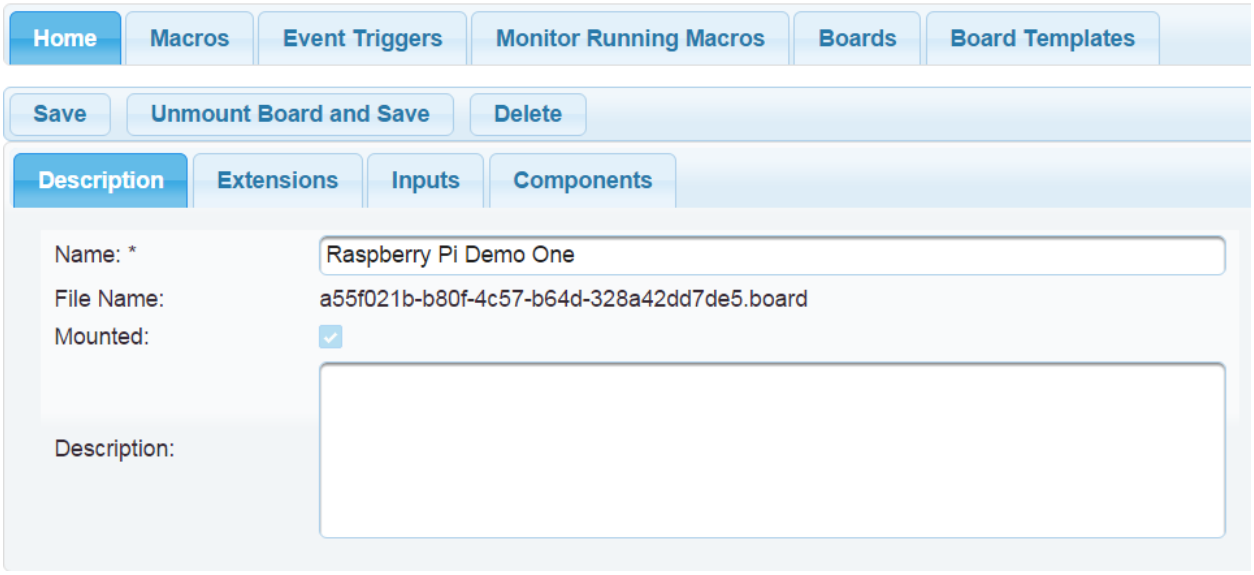

S1
S2
LED1
Red (633nm)
R1
220Ω
LED2
Red (633nm)
R2
220Ω
LED3
Red (633nm)
R3
220Ω
Raspberry Pi2
3V3
5V
GPIO2 SDA1 I2C
GPIO3 SCL1 I2C
GPIO4
GPIO17
GPIO27
GPIO22
GIPO10 SPI0_MOSI
GPIO9 SPI0_MISO
GPIO11 SPI0_SCLK
ID_SD I2C ID EEPROM
GPIO5
GPIO6
GPIO13
GPIO19
GPIO26
RaspberryPi
Model 2 v1.1
GPIO21
GPIO20
GPIO16
GPIO12
ID_SC I2C ID EEPROM
GPIO7 SPI0_CE1_N
GPIO8 SPI0_CE0_N
GPIO25
GPIO24
GPIO23
GPIO18 PCM_CLK
GPIO15 UART0_RXD
GPIO14 UART0_TXD
GND
fritzing


Red LED
GPIO_12
Yellow LED
GPIO_13
Green LED
GPIO_14
Input 0
GPIO_00
Input 1
GPIO_02
Raspberry Pi Model 2 v1.1
© Raspberry Pi 2014
Resistors are 220 Ohms.
Input 0 and Input 1 have
Internal pull up resistors.
fritzing

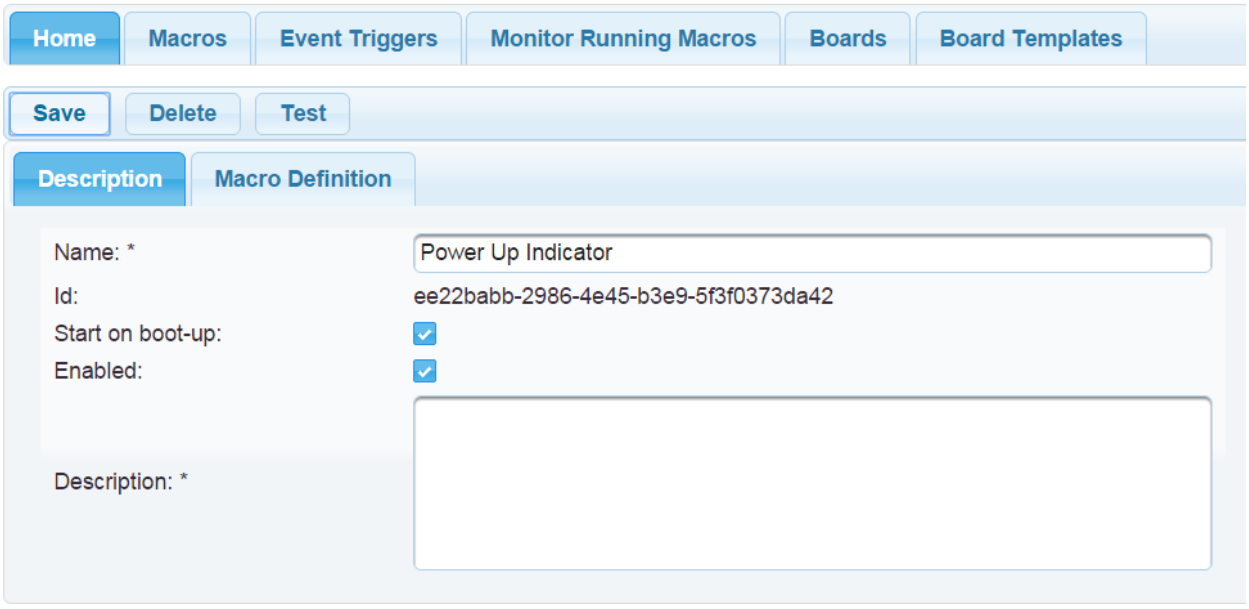
Example One – Power on Blinking Led

In this example, we will configure Breezy4Pi and program a macro to start the green LED to blink for five seconds when the Breezy4Pi application first boots up.

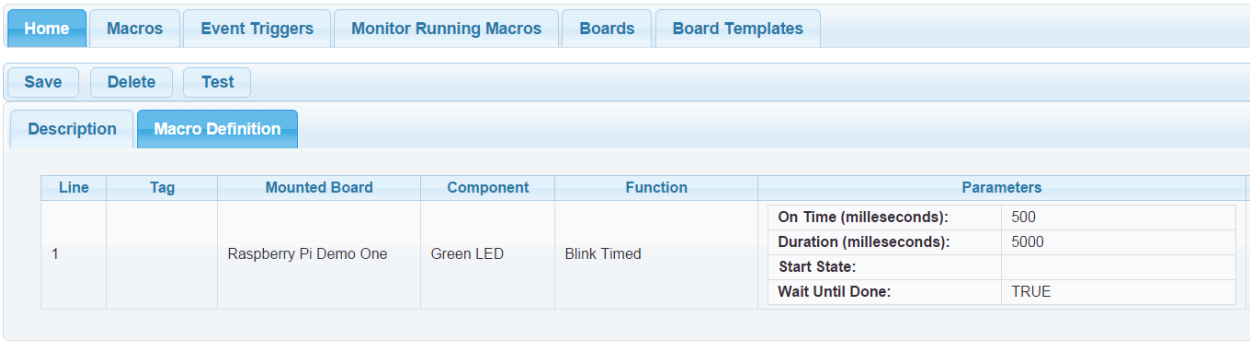
First, make sure the 'Raspberry Pi Demo One' board is mounted. If it is not, then click on 'Mount Board and Save'. You will need to reboot the Tomcat webserver and possibly reboot and/or power cycle the Raspberry Pi. This ensures that the underlying libraries upon which Breezy4Pi depend are properly re-initialized. When booting up, Breezy4Pi will properly configure the GPIOs according to the mounted board definitions.



Navigate to the Macros tab and select the 'Power Up Indicator' macro. Make sure that the macro is enabled which allows it to run and that it can start on boot-up.



If you click the 'Test' button, the macro will execute and you should see the green led blink for five seconds with a on time of one half a second. This is what the macro looks like:



Go ahead and reboot the Tomcat webserver and when the server starts up, you should see the green led blink for five seconds.

When you feel complete with this exercise, go ahead and disable the macro. You can also uncheck the 'Start on boot-up' checkbox. Even if the 'Start on boot-up' check box is still checked, the macro will not run.

Now it's time to try something a bit more challenging…

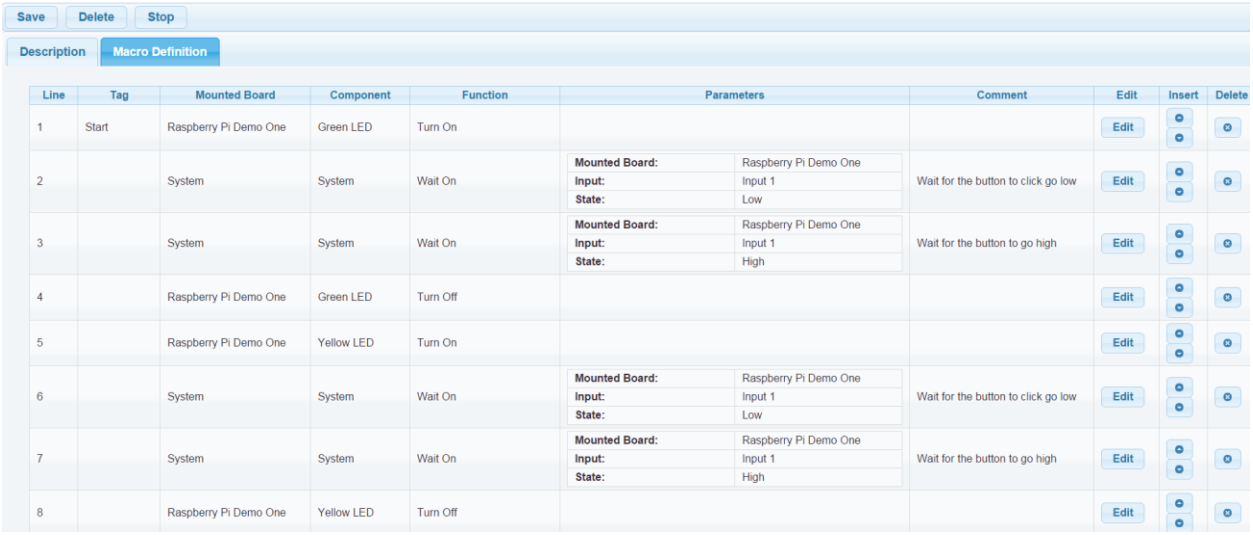
Example Two – Playing Traffic Cop

This example is a bit of an extension of the first one in that we continue to use the same mounted board, 'Raspberry Pi Demo One'.

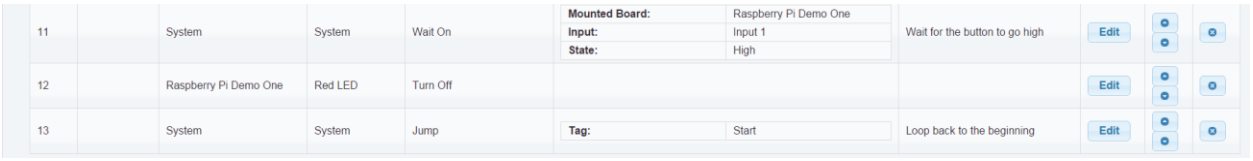
Now, navigate to the Macros tab and select 'Raspberry Pi Example One'. Make sure that it is enabled so that it can run. Leave the 'Start on boot-up' unchecked.

If you go to the 'Macro Definition' tab, you will see that things start to get interesting. First we turn on the green led and wait until the 'Input 1' momentary switch goes low. Then it waits until the switch returns to its high state when it is released. There, you will see the green led turn off and the yellow led turn on. This sequence continues for each of the colored LEDs until it reaches the last line where by it loops back to the beginning of the macro.

Here's an example of what this looks like:



Then, at the end, we loop back to the beginning:



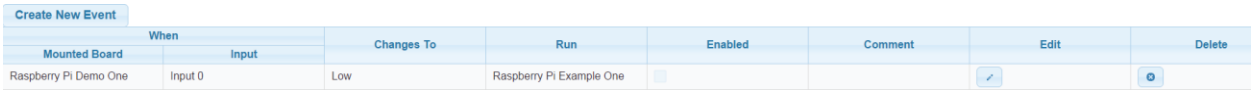
Click on the 'Test' button to start the macro. Each press of the momentary switch, 'Input 1' will turn on and off each LED in sequence. To end the macro, click on the 'Stop' button. Alternatively, you can end the macro from the 'Monitor Running Macros' tab.

If it appears that the LEDs are advancing erratically, try playing around with the debounce value for Input 1. To do this, stop the macro, go to the board editor for this board and navigate to the Inputs tab. Unmount the board to allow editing of Input 1. Change the Debounce value, then save the edit and Mount the Board and Save. Then reboot the Tomcat webserver.

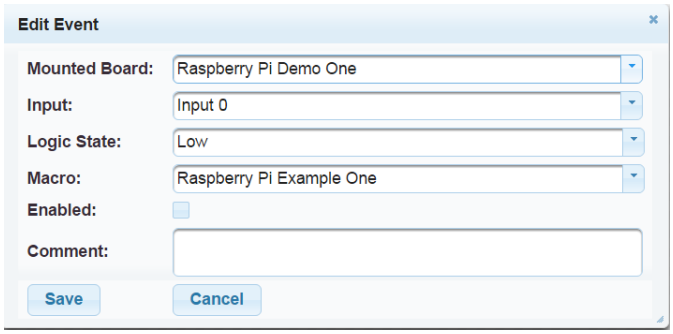
Example Two with a Twist – Playing Traffic Cop Using an Event

Once you feel complete with the above exercise, we can now add an event trigger to kick off a macro. First, make sure the macro we were playing with above is terminated. Once again, either click on the 'Stop' button, or click on 'End Macro' from the 'Monitoring Running Macros' tab.

Now, navigate to the 'Events Trigger' tab. You should see something like this:

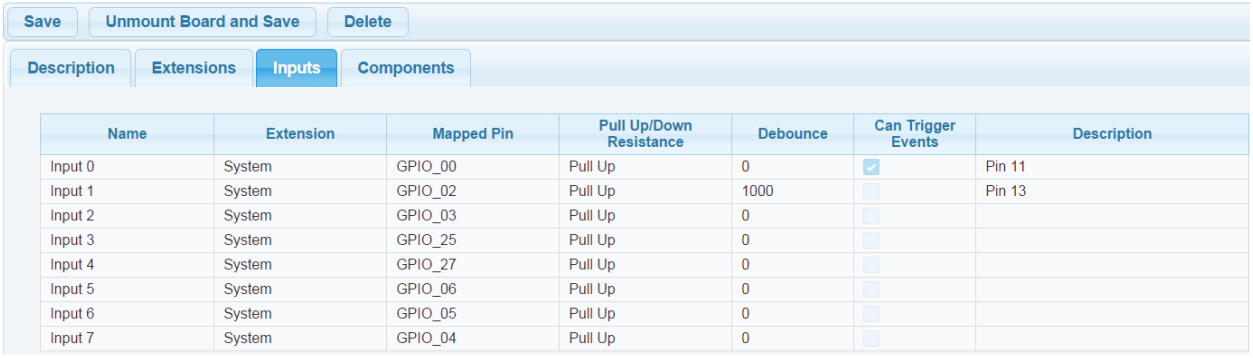


You will notice that this event trigger is not enabled. What this event trigger will do is start the macro, 'Raspberry Pi Example One' when the momentary switch connected to 'Input 0' is clicked causing it to go low. Click on the pencil icon in the Edit column to open the event trigger editor then click the 'Enabled' checkbox. Click 'Save' when done:



Now, push the 'Input 0' momentary push button to start the macro. The green LED should light up and pushing the 'Input 1' push button should advance the color LEDs.

If the event trigger is not working, make sure the macro is enabled and that the 'Can Trigger Events' checkbox for 'Input 0' is checked in the editor for the 'Raspberry Pi Demo One' board.

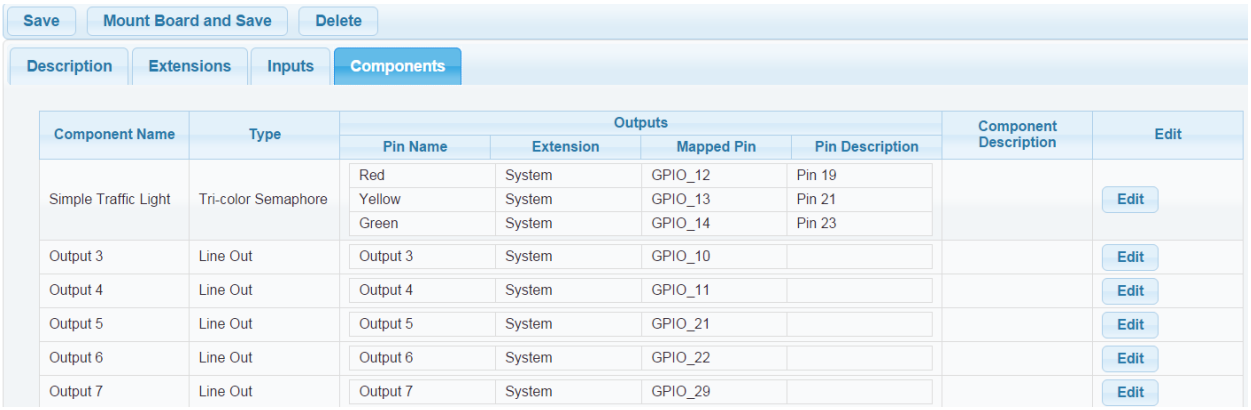


Name	Extension	Mapped Pin	Pull Up/Down Resistance	Debounce	Can Trigger Events	Description
Input 0	System	GPIO_00	Pull Up	0	☑	Pin 11
Input 1	System	GPIO_02	Pull Up	1000	☐	Pin 13
Input 2	System	GPIO_03	Pull Up	0	☐	
Input 3	System	GPIO_25	Pull Up	0	☐	
Input 4	System	GPIO_27	Pull Up	0	☐	
Input 5	System	GPIO_06	Pull Up	0	☐	
Input 6	System	GPIO_05	Pull Up	0	☐	
Input 7	System	GPIO_04	Pull Up	0	☐	

Example Three – Simple Light Chasing

This example requires that the 'Raspberry Pi Demo One' board is unmounted and that the 'Raspberry Pi Demo Two' board be mounted. Also, disable the event trigger and the 'Raspberry Pi Example One' macro. You will need to reboot the webserver to ensure that the board you want to be mounted is properly mounted and that the other board is unmounted.

The 'Raspberry Pi Example Two' gives an example of how to work with a component. A component is an aggregate of output pins and has native functions that can be applied as a whole. For example, with the Tri-Color Semaphore, if you turn on one LED, the component ensures that the other two LEDs are turned off. Here is what the board definition looks like:

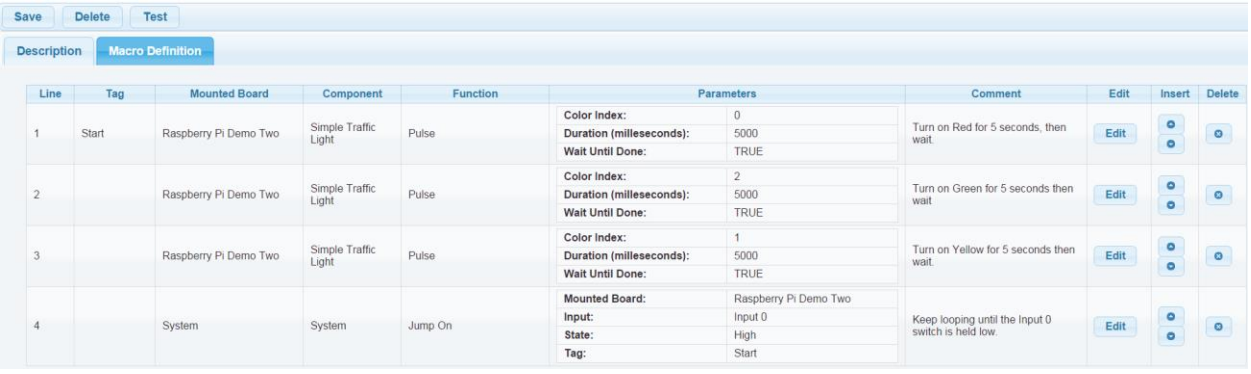


Component Name	Type	Outputs				Component Description	Edit
		Pin Name	Extension	Mapped Pin	Pin Description		
Simple Traffic Light	Tri-color Semaphore	Red	System	GPIO_12	Pin 19		Edit
		Yellow	System	GPIO_13	Pin 21		
		Green	System	GPIO_14	Pin 23		
Output 3	Line Out	Output 3	System	GPIO_10			Edit
Output 4	Line Out	Output 4	System	GPIO_11			Edit
Output 5	Line Out	Output 5	System	GPIO_21			Edit
Output 6	Line Out	Output 6	System	GPIO_22			Edit
Output 7	Line Out	Output 7	System	GPIO_29			Edit

The component named 'Simple Traffic Light' will be referenced in the example macro, 'Raspberry Pi Example Two'.

Once this board is mounted, you will see the 'Edit' column be replaced by 'Test'. Clicking on the 'Test' button will execute a test routine for the component. In this case all three LEDs will blink once.

Now navigate to the Macro tab and select the 'Raspberry Pi Example Two'. Click 'Test' to run the macro. You should see something like this:



Line	Tag	Mounted Board	Component	Function	Parameters	Comment	Edit	Insert	Delete
1	Start	Raspberry Pi Demo Two	Simple Traffic Light	Pulse	Color Index: 0 Duration (milleseconds): 5000 Wait Until Done: TRUE	Turn on Red for 5 seconds, then wait.	Edit		
2		Raspberry Pi Demo Two	Simple Traffic Light	Pulse	Color Index: 2 Duration (milleseconds): 5000 Wait Until Done: TRUE	Turn on Green for 5 seconds then wait	Edit		
3		Raspberry Pi Demo Two	Simple Traffic Light	Pulse	Color Index: 1 Duration (milleseconds): 5000 Wait Until Done: TRUE	Turn on Yellow for 5 seconds then wait.	Edit		
4		System	System	Jump On	Mounted Board: Raspberry Pi Demo Two Input: Input 0 State: High Tag: Start	Keep looping until the Input 0 switch is held low.	Edit		

What this macro does is pulse each LED in turn for a total of five seconds. Notice the 'Wait Until Done' entry. If this entry is 'FALSE' then the Pulse instruction will start the pulse then immediately go onto the

next instruction. If it is set to 'TRUE' then the Pulse instruction waits until it has completed the pulse before going onto the next instruction.

So, in this macro, you will see each LED turn on for five seconds then turn off before the next one comes on. This macro will cycle forever until you either 'Stop' it manually or you push the momentary switch connected to 'Input 0' causing it go low and thus failing the 'Jump On' condition.

For added fun, you can create an event trigger and associate it with this macro if you'd like to kick it off with a button press. Please know, however, that there can only be one event trigger associated with an Input. This may change in the future.