Quick Start Guide

Introduction

The Raspberry Pi Compatible Compiler for LabVIEW is a product based on LabVIEW (Laboratory Virtual Instrument Engineering Workbench) by National Instruments. LabVIEW is a graphical programming language that uses icons instead of lines of text to create applications. In contrast to text-based programming languages that use instructions to determine the order of program execution, LabVIEW uses dataflow programming. In data flow programming, the flow of data through the nodes on the block diagram determines the execution order of the VIs and functions. VIs, or virtual instruments, are LabVIEW programs that imitate physical instruments. This document assumes basic familiarity with LabVIEW. For more information about LabVIEW, visit www.ni.com/labVIEW.

The Raspberry Pi Compatible Compiler for LabVIEW is a true compiler product that allows one to compile, deploy and run stand alone, full fledge LabVIEW applications embedded in Raspberry Pi targets. All flavors of official Raspberry Pi (or RasPi for short) single board computer boards released by the Raspberry Pi organization are supported. However, performance differences may be noticed based on which version of Pi is selected and the complexity of the LabVIEW VI downloaded. The Raspberry Pi is a single board computer; therefore, the same performance considerations are valid in the selection of regular computers to run LabVIEW applications. The Raspberry Pi boards that have higher horse power CPUs and more RAM memory will outperform their counterparts with lower end CPUs and less RAM memory. As an example, the Raspberry Pi 2 Model B will outperform its older brother, the Raspberry Pi 1 Model B+. The former includes a 900MHz quad-core Arm Cortex-A7 CPU and 1GB of RAM memory while the latter is powered by a 700MHz single-core ARM11 CPU and 512M of RAM memory.

Another point that is fundamental to highlight is that the Raspberry Pi Compatible Compiler for LabVIEW requires the RasPi target to have an Ethernet port and an IP address for communication with the host development computer for initial download of the compiled VI. Once the download is complete, the RasPi can operate standalone and disconnected from the Ethernet network. Furthermore, the setup of the RasPi, as it will be seen in a future section, requires the RasPi to have Internet access, so all the correct packages can be installed upon execution of the setup script.

This requirement doesn't constraint the use of the compiler with RasPi boards that include an Ethernet port as part of the board. The user is free to elect using any Raspberry Pi, as long as it has Ethernet connectivity. As an example, the user can setup a Raspberry Pi Zero with an Ethernet to USB dongle.

The Raspberry Pi Compatible Compiler for LabVIEW product works in combination with any of the available editions of LabVIEW for Windows; Base, Full, Professional and Home. The oldest version of LabVIEW supported will be LabVIEW 2014. One doesn't need to have the LabVIEW Application Builder installed on the machine, which allows the use of both the LabVIEW Base and Home editions. The compiler also allows the compilation of full GUIs or Console Application. Refer to the section named Supported Compilation Types for more information on the differences between the two compilation types.

You will be able to run the Raspberry Pi Compatible Compiler for LabVIEW Beta package for a period of thirty days.

This document also includes a FAQ section that you may find useful in debugging issues as they occur.

What you Need to Get Started

The below shows the list of items you will need in order to get started:

- A Raspberry Pi board with Ethernet connectivity resources
- A SD Card (minimum 8GB) loaded with the Raspbian Jesse Image (refer to the section named How to Setup the SD Card for more information about this step)
- The power supply required by your specific RasPi board. A 5V/2A power supply is a safe bet for all flavors of RasPi
- Either a SD Card USB Adapter or some other way to download the image to the SD Card
- A Windows development computer running any edition of LabVIEW
- HDMI cable to connect your RasPi to a monitor. One of the cable connectors must be HDMI to be connected to the RasPi, the other connector end can be of a matching type of the monitor input connector.
- A regular computer monitor with input connector matching the side of the HDMI cable that is not to be connected to the RasPi
- USB keyboard and USB Mouse

How to Setup the SD Card

The first step in getting your RasPi target ready for the compiler is to get the correct Operating System to the SD Card. This section describes the steps you will need to follow.

1) Go to the web link https://www.raspberrypi.org/downloads/raspbian/ and download the Zip file for the Raspbian Jessie (not the Raspbian Jessie Lite). This is the direct download link: https://downloads.raspberrypi.org/raspbian_latest

2) Unzip the file to a directory in your computer. You will see a XXX-raspbian-jessie.img file; where the XXX corresponds to the date of the build published to the site

3) The next step is to install the Operating System Image to the SD Card. Follow the instructions shown on this link: https://www.raspberrypi.org/documentation/installation/installing-images/windows.md

4) Once the Win32DiskImager utility completes its process, the SD card is ready to be connected to the RasPi board

Setting Up The RasPi

Now that the SD Card contains the Operating System, it is time to connect the RasPi. The below shows the steps to be followed to make sure the RasPi target is setup.

1) Connect the SD Card to the RasPi board' SD Card slot

2) Connect the RasPi to the monitor via the HDMI cable

3) Connect USB keyboard and USB mouse to two USB ports in your RasPi

4) Plugin the Power Supply to the RasPi connector and to the electrical outlet

5) Turn on the monitor

6) Once the RasPi finishes booting, you will be directed to the X Window application. The first thing to make sure is that an IP address was assigned to your RasPi. The easiest way to do that is by hovering over (not clicking) the network connection icon as pointed out by the following figure. The IP address assigned to your RasPi will popup. Write down the IP address as you will need it to configure your RasPi target on the compiler application
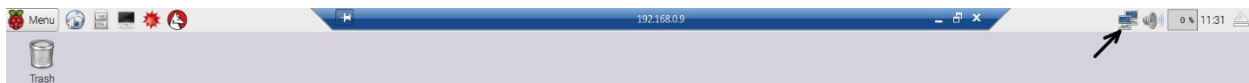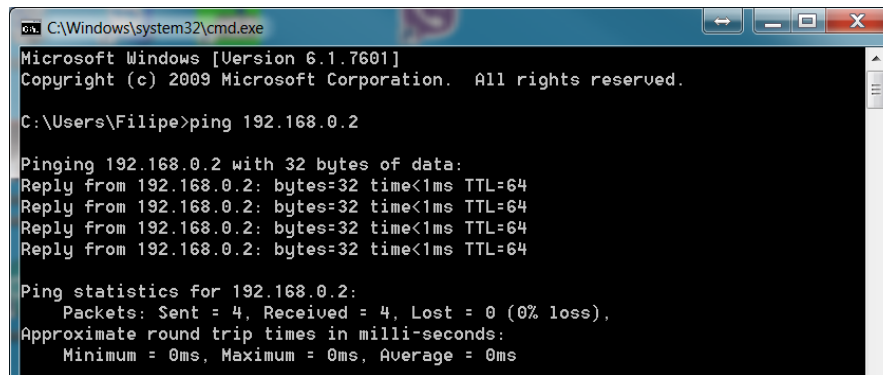
7) If you are using a SD card with capacity above 4GB, you will need to extend the File System on your RasPi so it used the full storage space of the SD card. To do that; follow the instruction on the picture below.
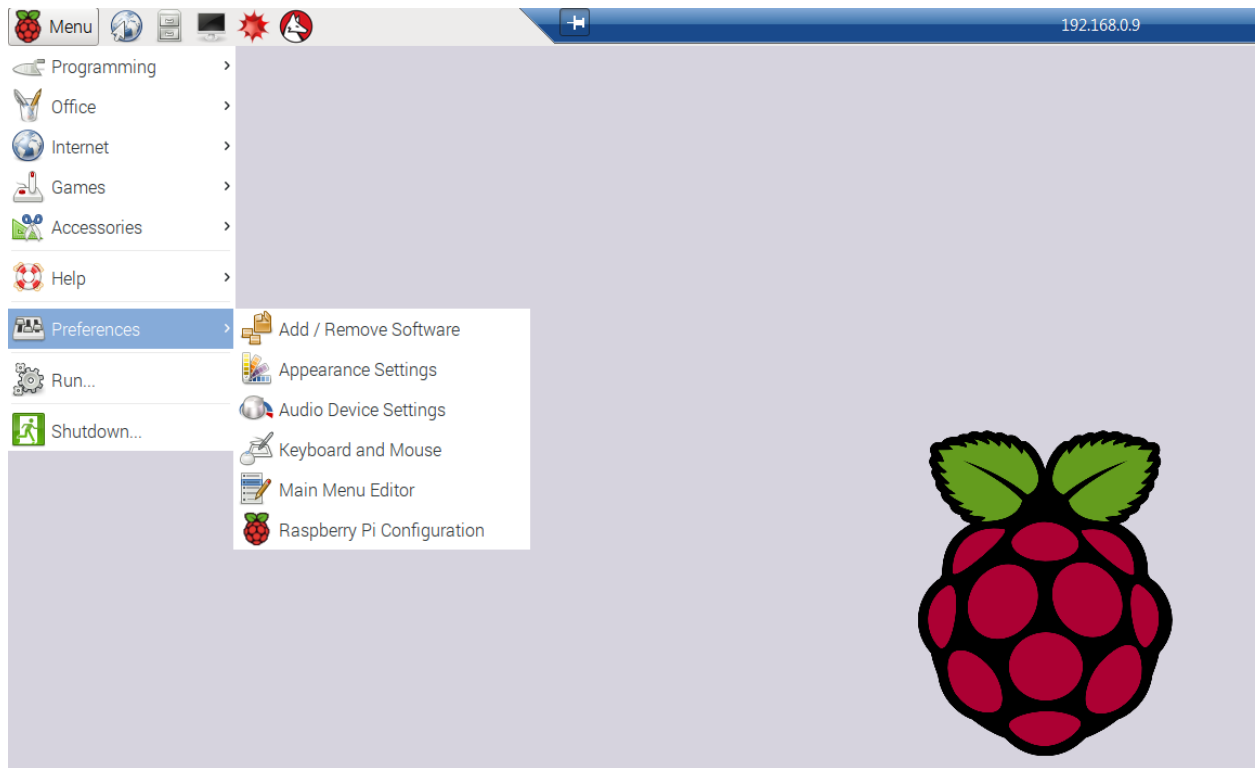
8) Click on the Expand File System button as highlighted below. The RasPi will prompt you to reboot for the changes to take effect. Go ahead and reboot it
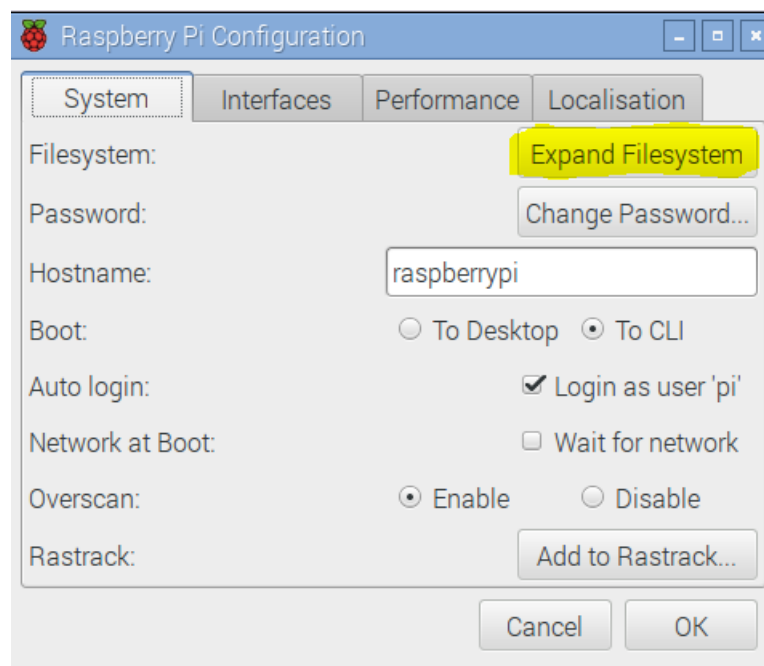
9) From your windows machine, ping the IP address you have written down to make sure it can see the RasPi target. To do that, click on the Start button on the Start button on the low left corner of your screen and type **cmd**. This will bring the command prompt screen. Type in the command: **ping <IP Address>**, replacing the IP Address portion of the command with the corresponding IP address of your Raspberry Pi target

If your development computer can communicate with the Raspberry Pi target, you will get replies in the format of the illustration above. If the ping operation times out, your Raspberry Pi can't be seen by the development computer for some reason. This needs to be fixed before you proceed to the next steps

10) Make sure your RasPi can access the Internet, as that will be a requirement for the installation of the needed extra packages as part of next step

If your Raspberry Pi can't access the internet, you need to stop now and make sure you fix the problem before proceeding.

11) Bring the command prompt according to the following figure.

12) In the command prompt window; type in the following command: **sudo apt-get update**
This will prompt the Raspbian Operating System running on your Raspberry Pi to update itself to its latest released packages as shown below. The lines showing on your Pi may vary and most likely will not match the lines shown by the illustration.


13) Once the Operating System is updated, the next step is to make sure SVN is installed on your RasPi. We will need SVN in order to fetch the external packages we created and hosted on a public Github repository. In the same command prompt, type the following command: **sudo apt-get install subversion**

14) Once the svn installation is complete, on the home directory of your Raspberry Pi, create a directory you wish to host the script files; according to the following illustration.
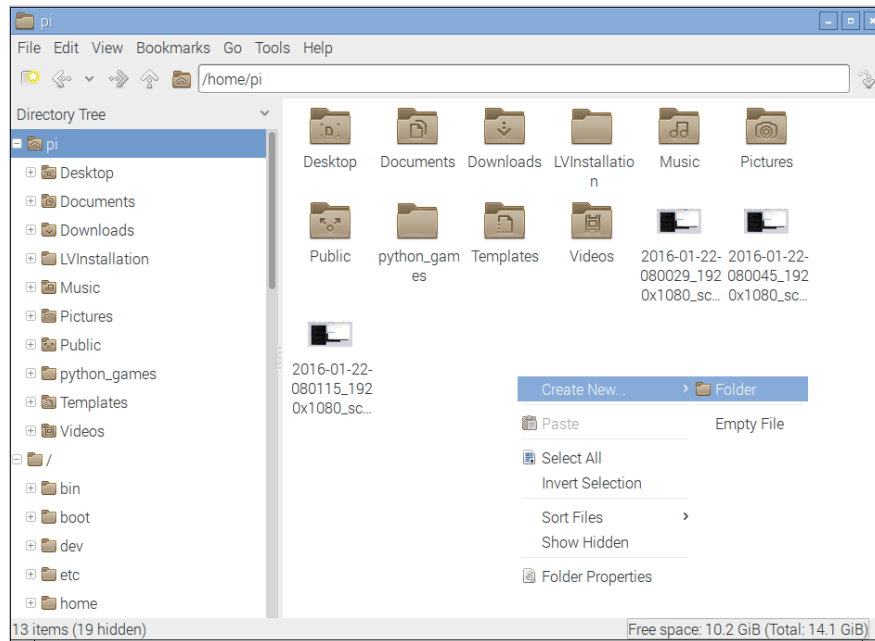
IMPORTANT: MAKE SURE THE DIRECTORY YOU CREATE DOESN'T HAVE SPACE CHARACTERS IN THE NAME. FOR EXAMPLE; a directory named LVPi Scripts should be renamed to LVPiScripts.

Navigate to the newly created directory and open a shell window, as illustrated below. The illustration shows a directory named LVPiScripts, but one can chose any directory name, provided it has no space characters.
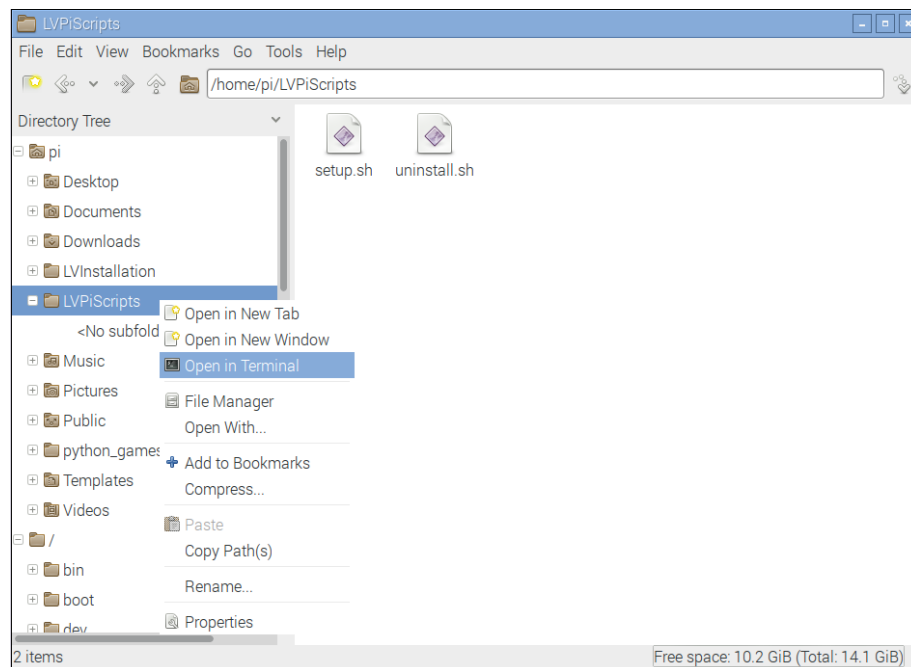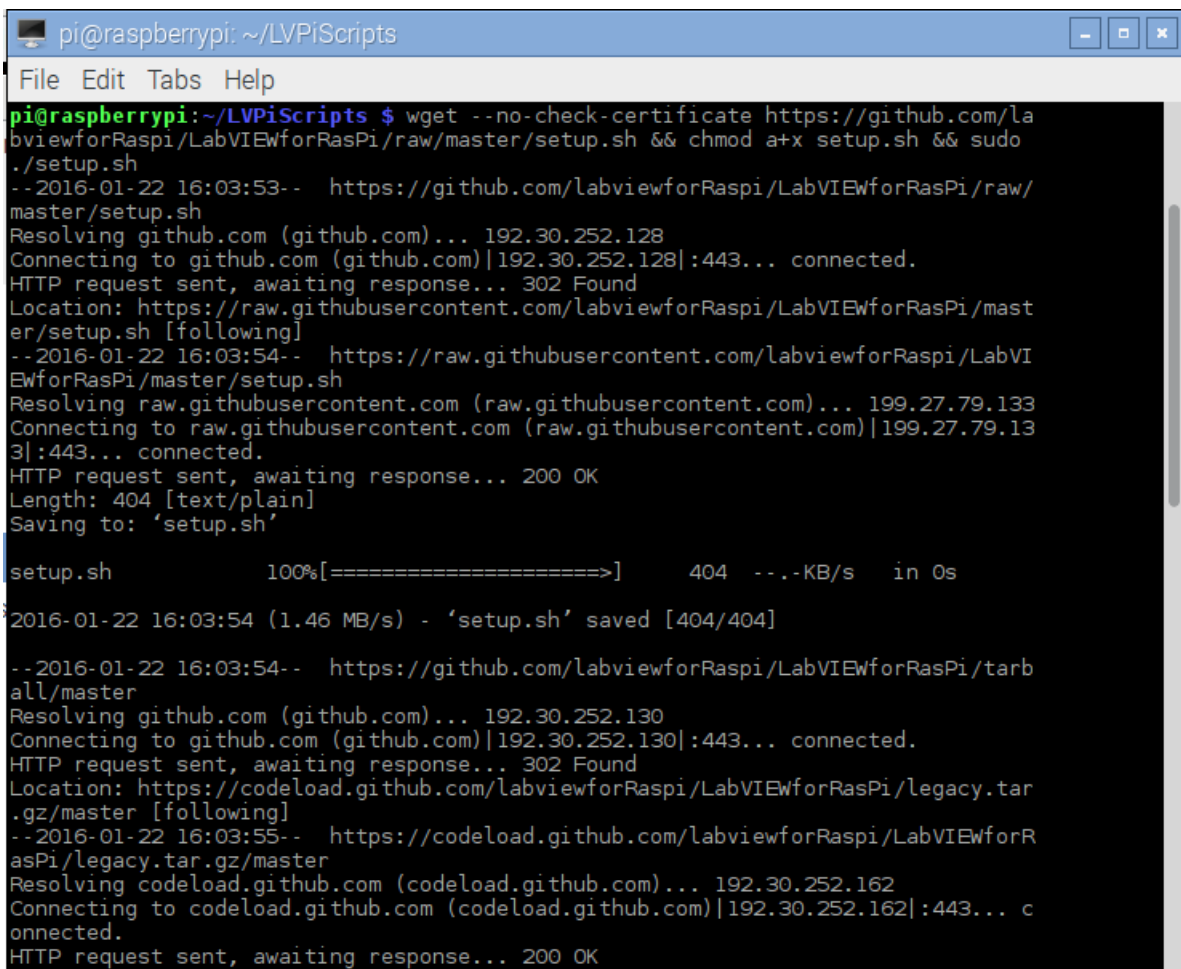
Once the shell window is open; type in the command:

wget --no-check-certificate https://github.com/labviewforRaspi/LabVIEWforRasPi/raw/master/setup.sh && chmod a+x setup.sh && sudo ./setup.sh

It is important the command above to be typed exactly right to avoid errors. Also, note that there is no carriage return after the words "no-check-" or "chmod".

Once the Enter key is pressed, your Raspberry Pi will got to the public Github repository where the Linux script to update your Pi is hosted, and will start working, as illustrated below. This process takes about 45 minutes depending on your RasPi network connection.


IMPORTANT: Once the script finishes running, MAKE SURE YOU REBOOT YOUR RASPBERRY PI BEFORE YOU ATTEMPT CONNECTION FROM THE LABVIEW COMPILER.

NOTE: It is important to note that once the command above finishes executing, there will be two files included as part of the directory: setup.sh and uninstall.sh.

The user can uninstall the components installed in the Raspberry Pi by the command above by running the uninstall.sh script. To do that, right click at the directory and open a shell window, as illustrated below.
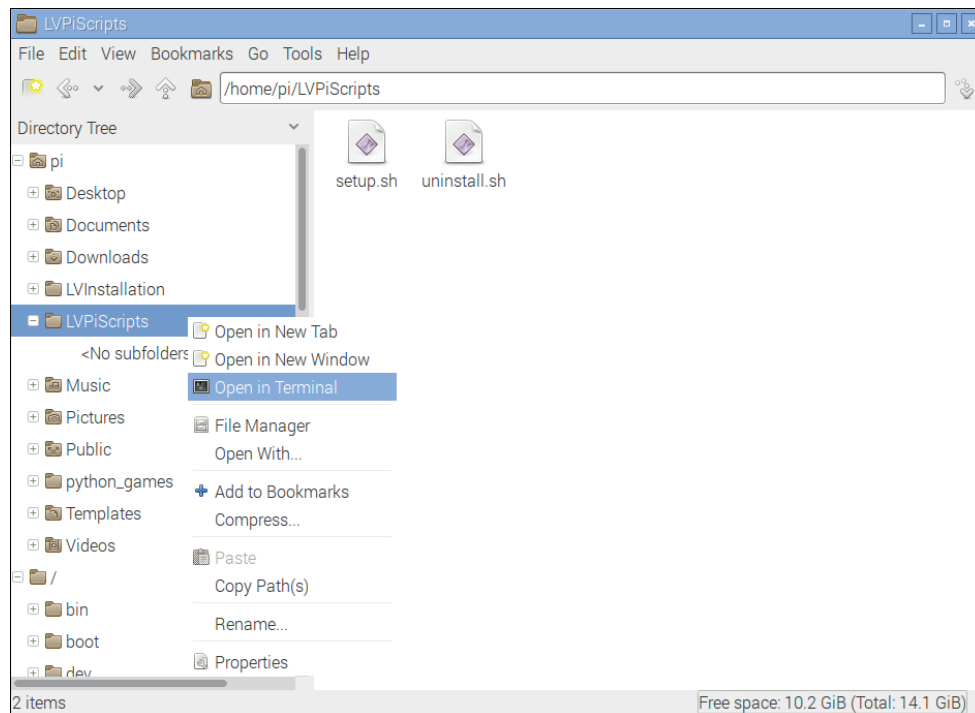
In the shell window, type the command: **sudo ./uinstall.sh**
This will uninstall all LabVIEW for Raspberry Pi components.

To reinstall the components, the user doesn't need to fetch the packages from the public Github repository again. Simply repeat the process above and run the command: **sudo ./setup.sh**

Installing the Raspberry Pi Compatible Compiler for LabVIEW

Now that your RasPi target is ready; the next step is to install the Raspberry Pi Compatible Compiler for LabVIEW on your development PC running LabVIEW for Windows. The compiler product has been packaged as a VI Package Manager vip file. Download the VIP file from this public repository.

One important detail about downloading the file that one needs to pay attention to. It is important to click of the Download ZIP button on Github, as shown by figure below. This will download all VIP

package files for the multiple versions of LabVIEW supported. Make sure to select one appropriate to the version of LabVIEW you are using.
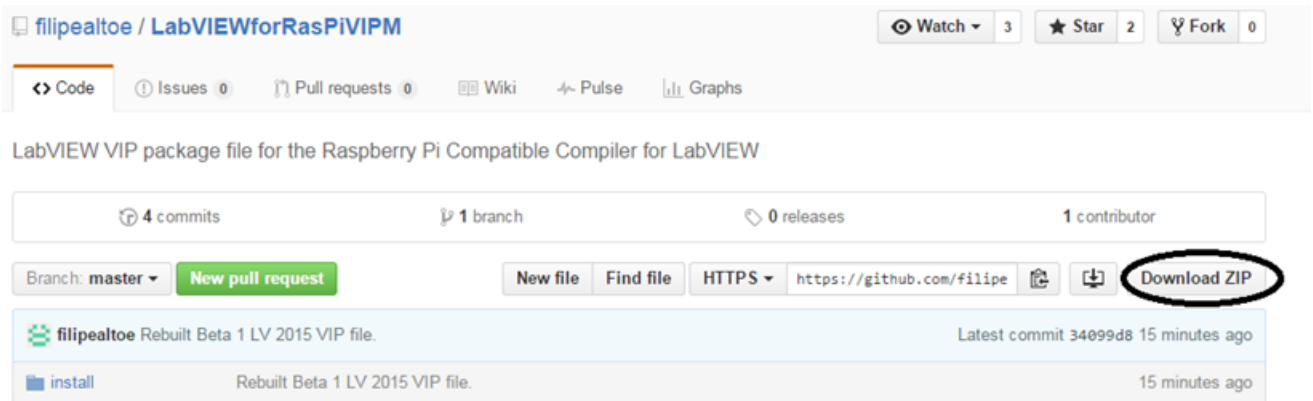
Before you proceed with the installation process, make sure your user account has administrator privileges. Installing the VIP package file in LabVIEW from an account without administrator privileges will prevent the compiler VIs to run. Once the file has been downloaded to your development computer, open VI Package Manager from LabVIEW->Tools Menu, as shown by figure below.
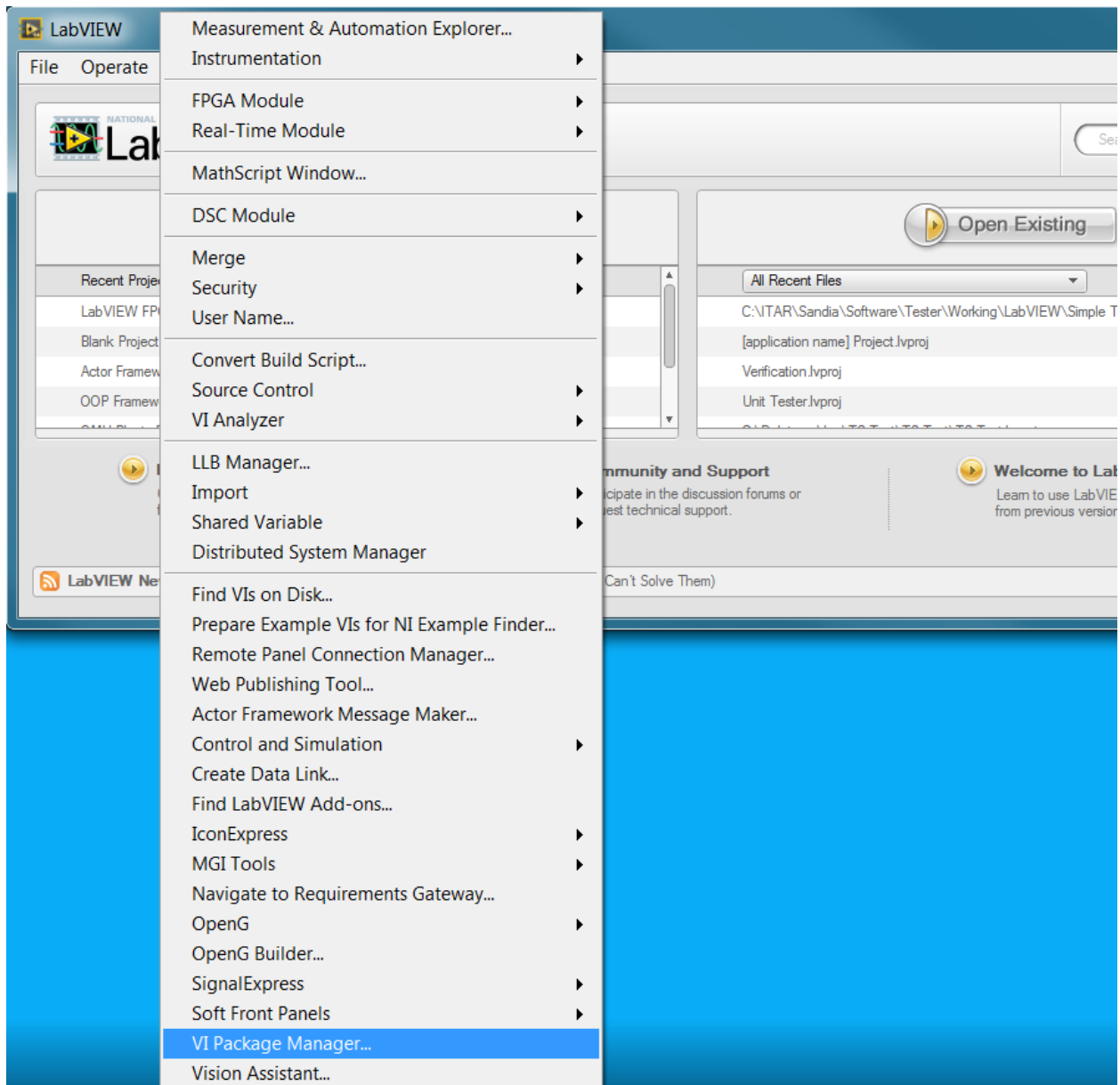
From within VIPM, open the RasPi Compiler for LabVIEW.vip file; as shown by the following figure.
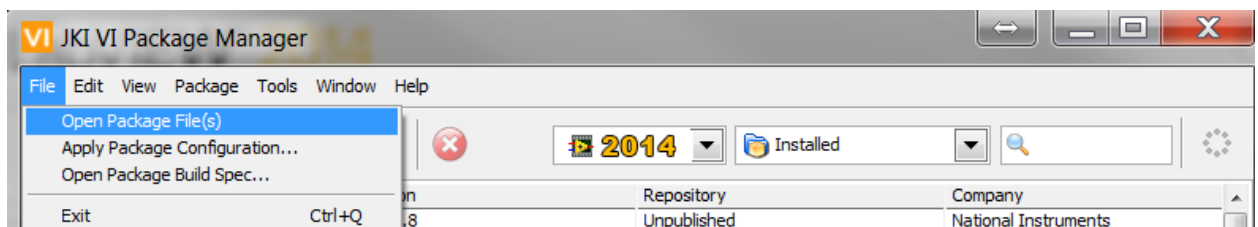
Follow the instructions on VIPM throughout the installation. Once that is complete; the VI Package Manager will show an item named Raspberry Pi Compatible Compiler for LabVIEW on the list of Installed packages, as shown below.
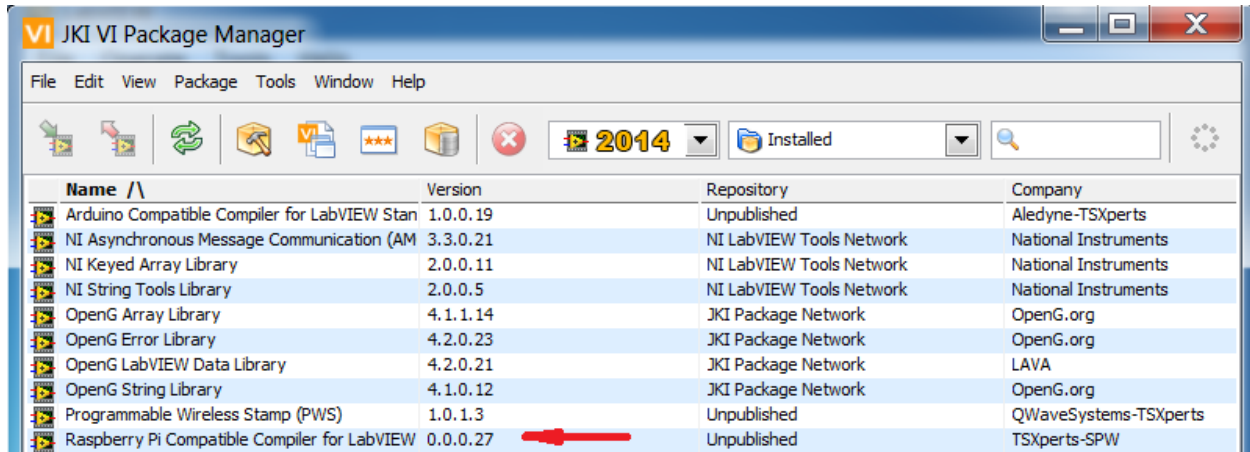
Moreover, a menu items named Raspberry Pi Compatible Compiler for LabVIEW will be added to your LabVIEW Tools menu, as shown by figure below.
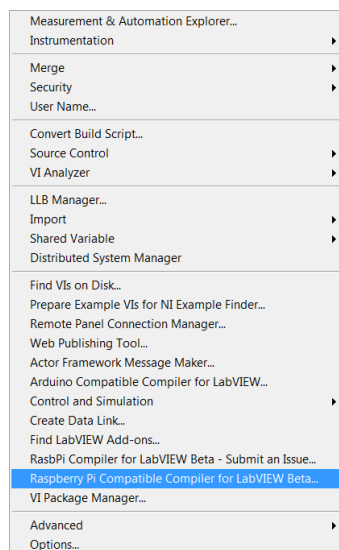
There will be in fact, two items added to your LabVIEW Tools menu; the Raspberry Pi Compatible Compiler for LabVIEW and the RasPi Compiler for LabVIEW – Submit an Issue. The Raspberry Pi Compatible Compiler for LabVIEW item will be the subject of the next section. The Submit an Issue item will be covered in the section named How to Report an Issue.

Deploying your First LabVIEW VI to the RasPi

Once you click on the item named Raspberry Pi Compatible Compiler for LabVIEW on the LabVIEW Tools menu, the following screen will be launched.
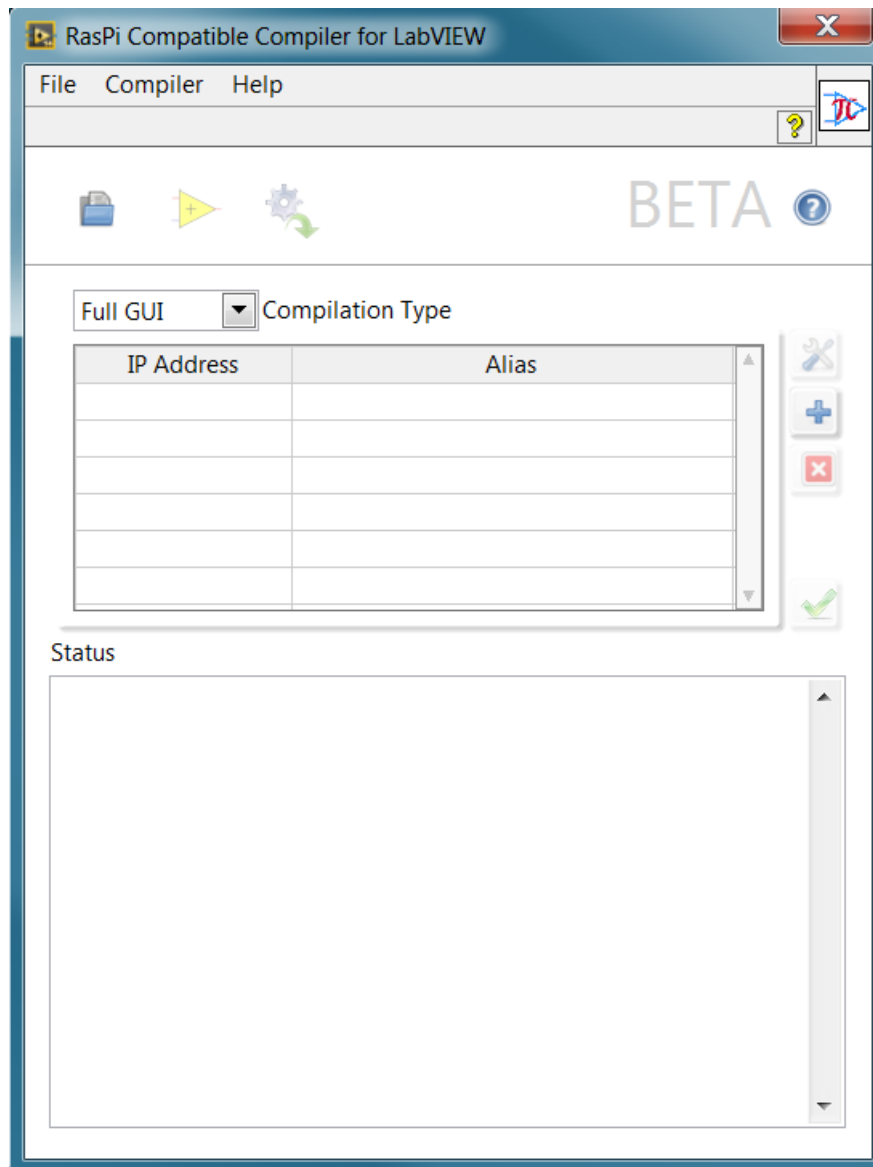
As you can see, the Beta watermark is displayed as part of the compiler main window to indicate this is the Beta release of the compiler product. The compiler screen has three main areas that are worth detailing; the RasPi connectivity area, the RasPi Compilation Tools area and the RasPi Compilation Status

area. The first one we will focus our attention to is the area in the middle, which we will call the RasPi connectivity area. This area is highlighted in the following figure.
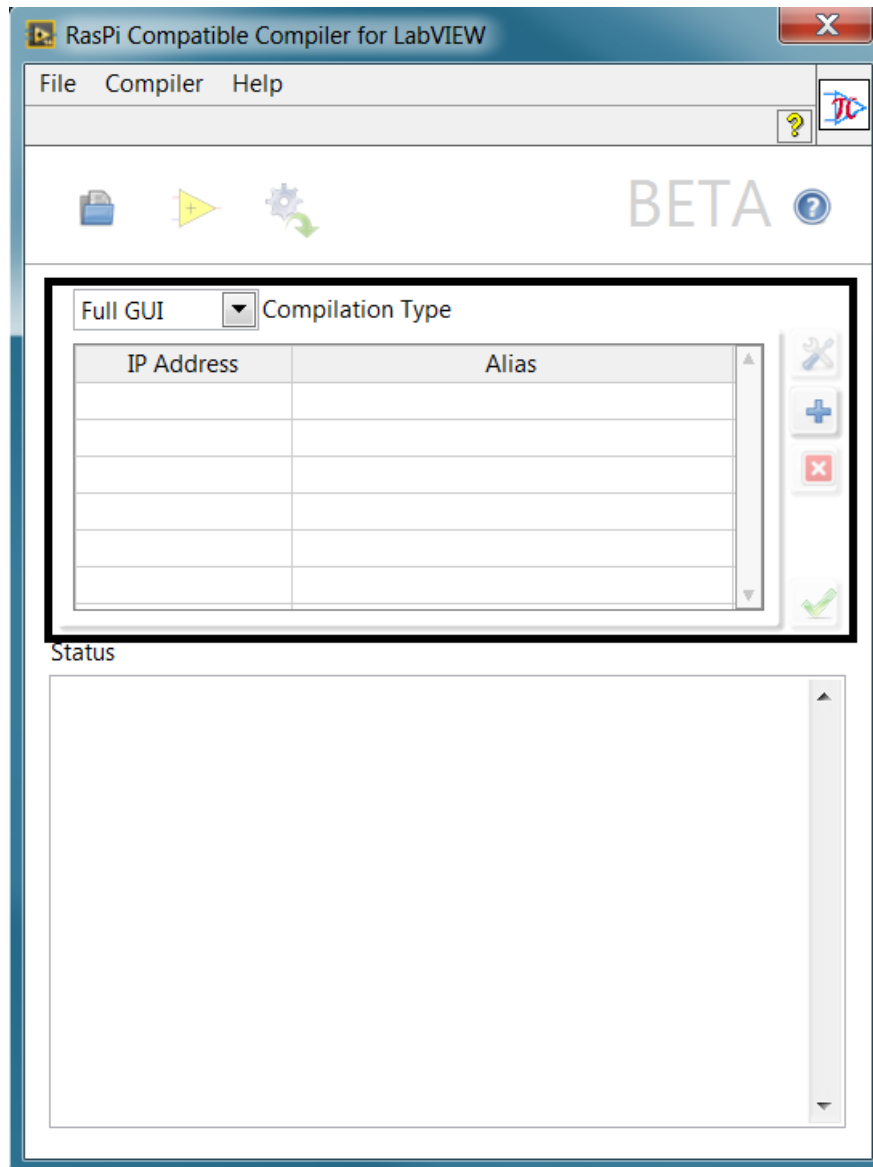
This is the area the user should start with when running the compiler for the first time. This section of the screen is where the user will configure one or multiple RasPi targets to receive compiled LabVIEW code from the Compiler product. The first step is for the user to enter a valid IP address for a RasPi target that has been setup to work with the compiler, as detailed on a previous section. In order to do that, the user should click on the "+" button of this area. This will launch the following screen.
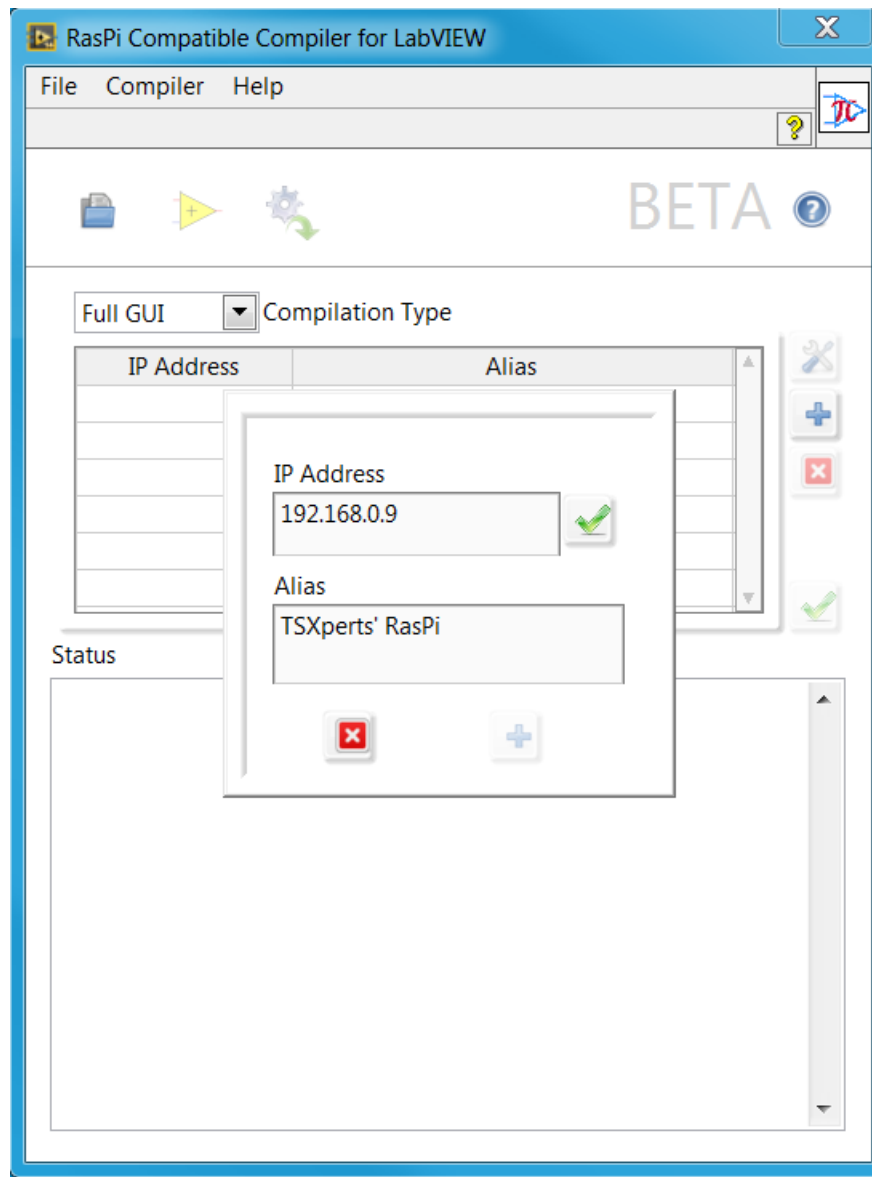
As one can see, this window allows the user to type a valid IP address for a previously configured RasPi target, as well as an Alias that will help the user in identifying which RasPi target the corresponding IP address relates to. The Alias is an optional field and can be left blank. It is important to notice that the "+" button will remain disabled until the user has clicked on the verification button; the green checkmark on the right hand side of the IP address field.

The verification button will trigger the compiler to attempt a connection with the RasPi IP Address, as well as verify if the corresponding RasPi target has been properly setup to receive compiled LabVIEW VIs from the compiler. Once this activity is complete, the user can add the target to the list by simply clicking the "+" button.

If you get an error at this step; either the IP address is not valid or the Raspberry Pi you have specified hasn't been properly setup to work with the compiler.
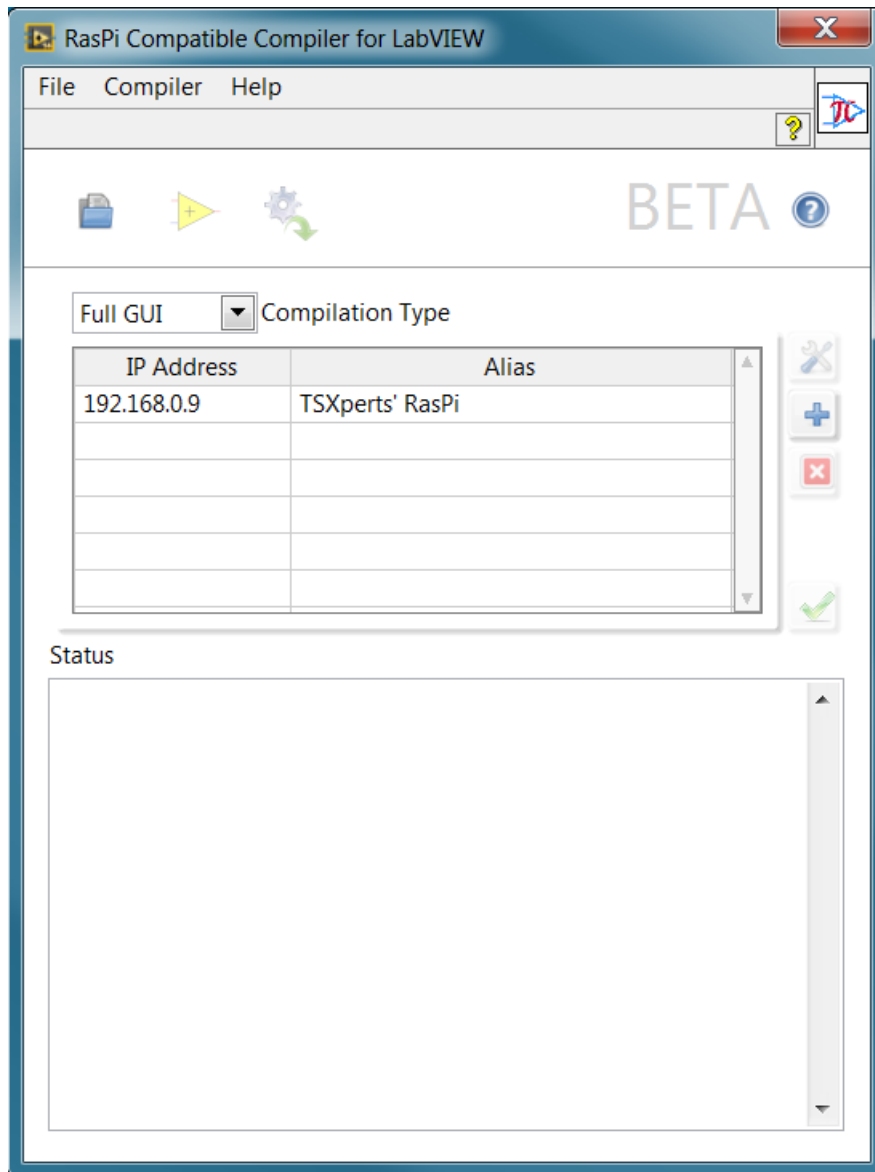
Once that is complete, the configured RasPi will show up on the list of configured IP addresses, as illustrated above. This activity only needs to be performed once per each new target. The information will be remembered by the compiler application even after it has been restarted.

This area of the compiler also offers the option for the user to edit an existing configured RasPi target's IP address and/or Alias; to remove an existing target and to run the connectivity verification on a previously configured RasPi target. Once a target is selected in the list, as shown by the following figure, the corresponding buttons for Editing, Deleting and Verifying Connectivity are enabled.
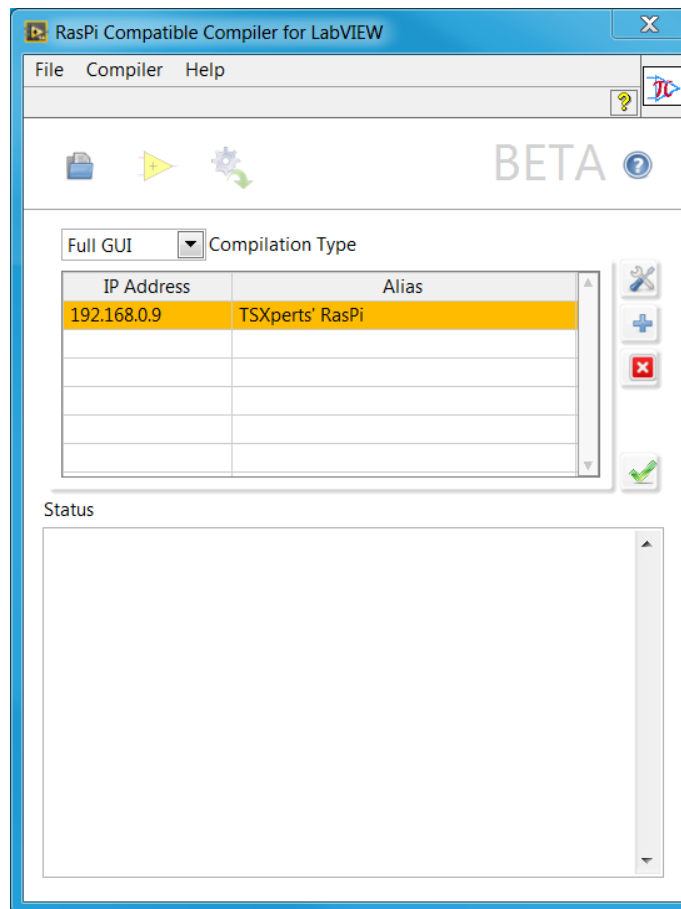
Clicking the editing button will open a separate screen the user can change either the target IP address, alias or both.
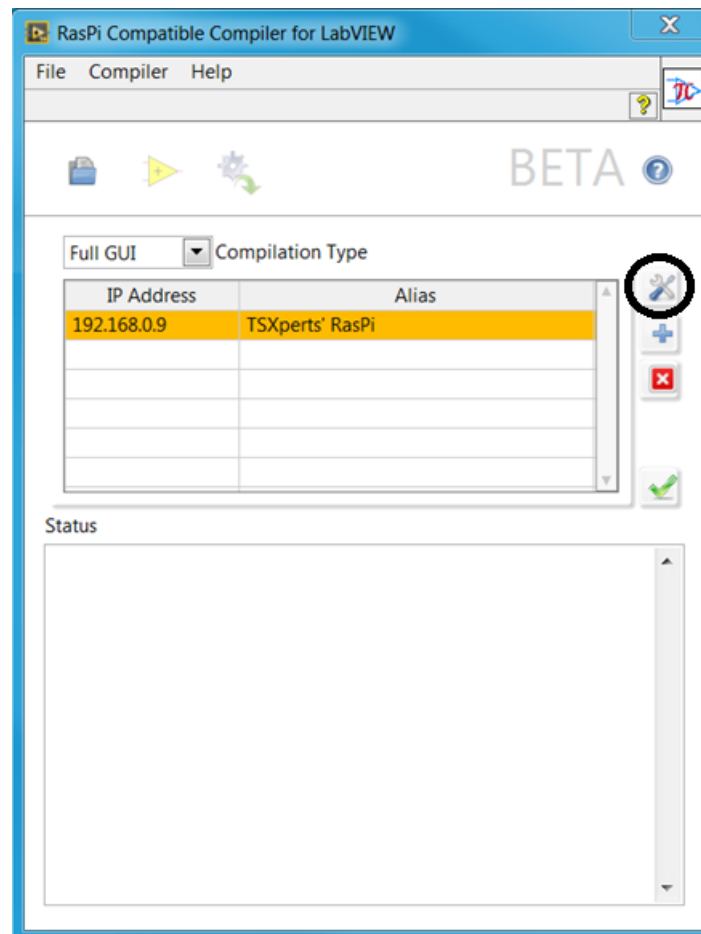
The user can have multiple Raspberry Pi targets setup to work with the compiler. To do that, just click the "+" button as illustrated below and configure the next Raspberry Pi target by following the steps you have done in setting up the first target.
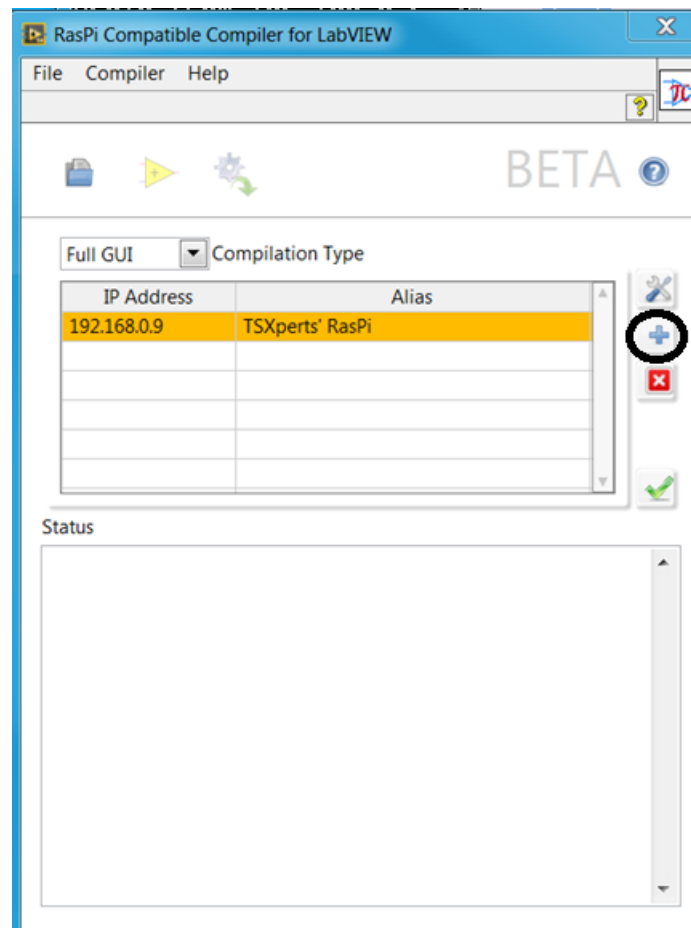
The next button below the one to add new targets can be used to delete the selected target from the list.

The final button is the one that allows verification of the connectivity between the development computer and the selected target from the list.
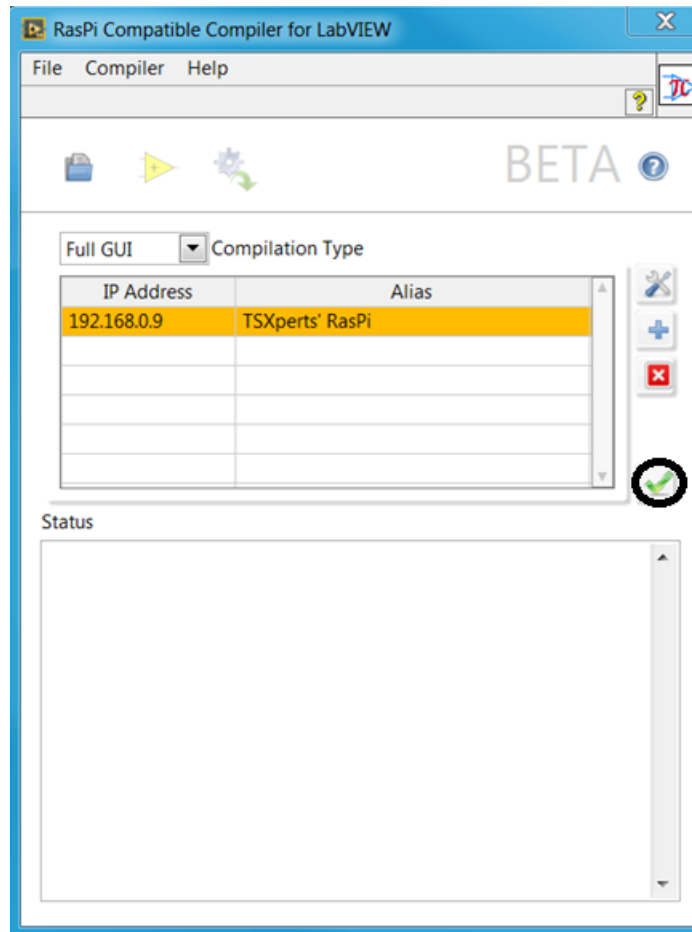
The last control in this area is the Compilation Type drop down. Let's postpone the description of this feature as it deserves a separate section to best describe what if offers the users.

The next area of interest of the main compiler application that deserves attention is the RasPi Compilation Tools area, highlighted in the next figure.
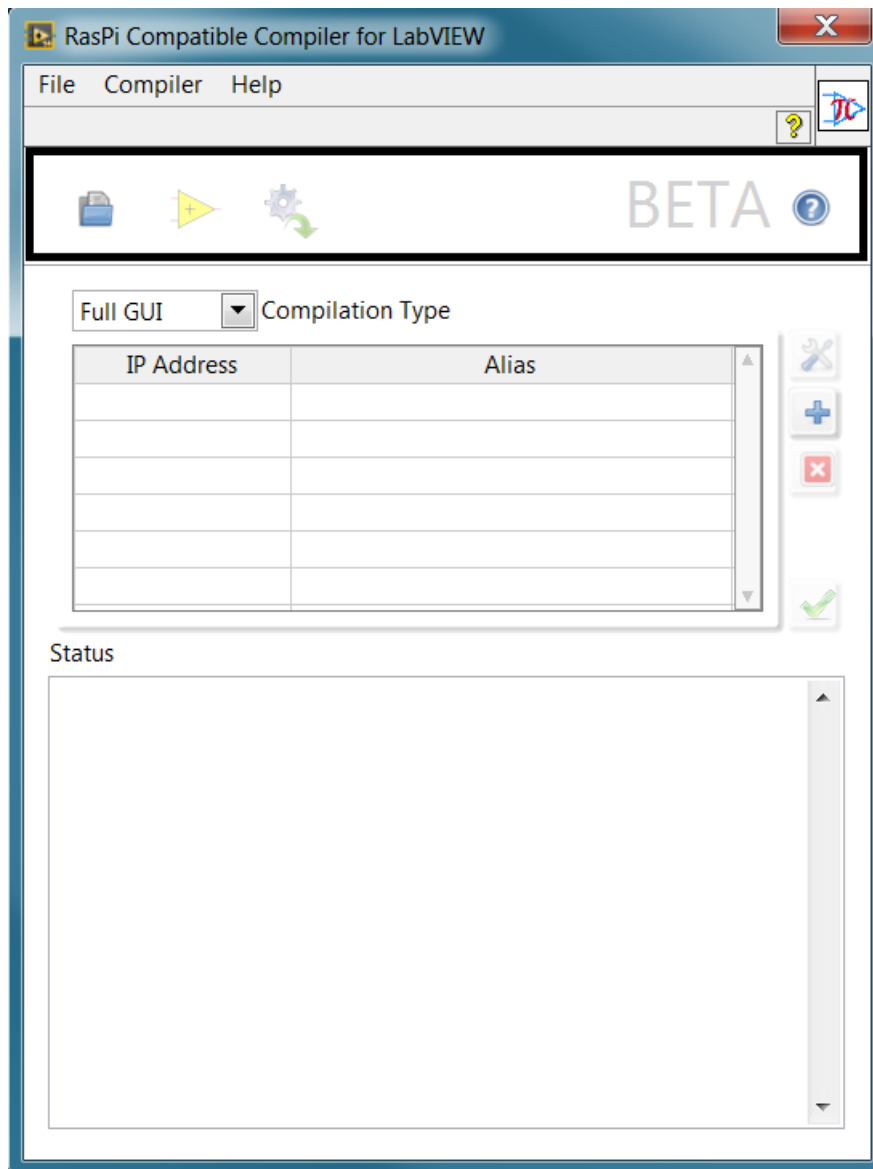
This area has four buttons, from left to right, the Load VI button, the Open Front Panel button, the Compile and Download button and the Open User Manual button.

The Load VI button allows the user to select the top level VI that will be compiled and downloaded to the selected and verified RasPi target. Once a VI is loaded, if it is in runnable state, the Open Front Panel button will be enabled and the compiler main window will show the loaded VI name as part of its title bar, as shown in figure below that has a VI named Edit Lists.vi loaded. If the loaded VI is not in a runnable state, an error will be generated and the Open Front Panel button will not be enabled.
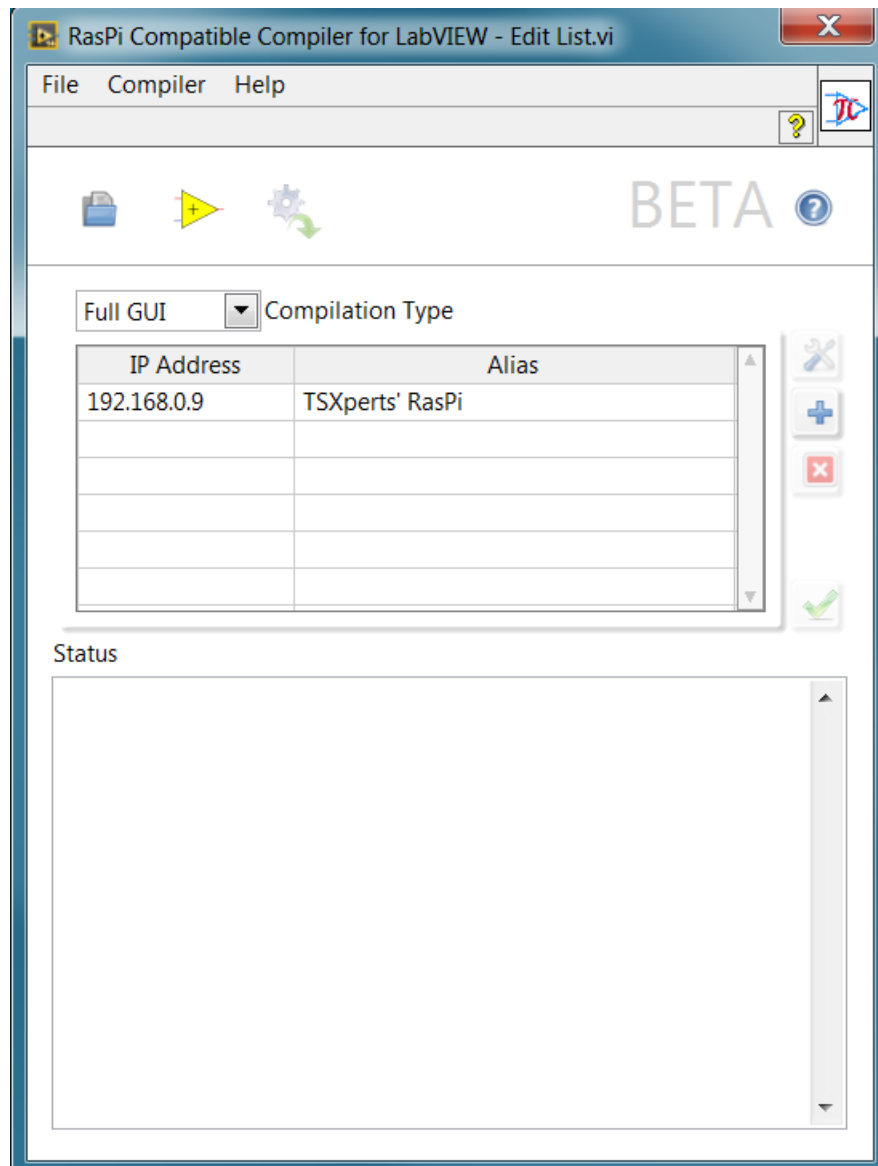
The Open Front Panel button is a convenience for the user. Clicking at that button will open the loaded VI front panel so the user can inspect it if needed.

Once a VI is successfully loaded, the last step prior for a compilation and download task is the selection of a RasPi target from the list of targets. Once a target is selected, the Compile and Download button will be enabled, as illustrated by the following figure.
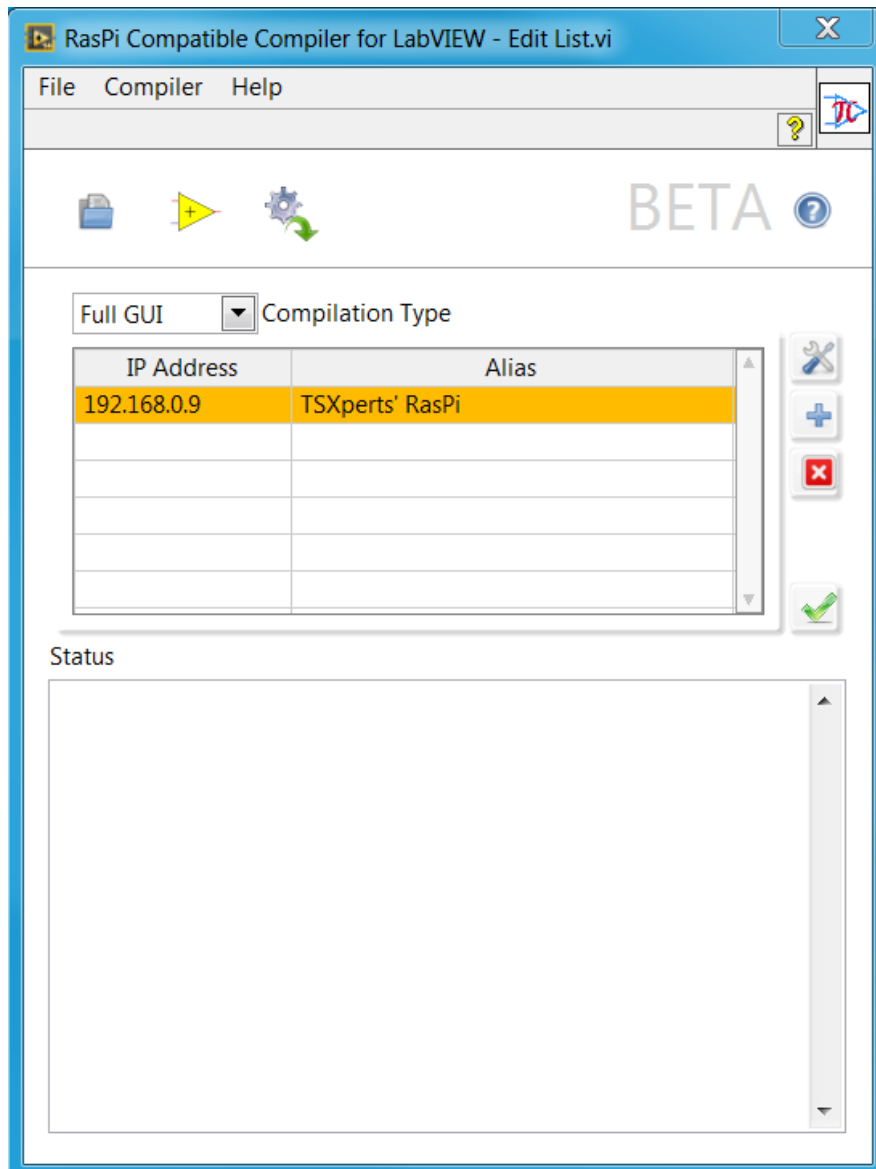
At this point, clicking at the Compile and Download button will trigger a compilation. Once the compilation is complete, the compiled VI will be downloaded to the selected RasPi target.

The last of the four buttons of this area is the help button, on the right hand corner. Clicking at this button will bring up this document.

This brings us to the analysis of the last of the three areas of interest of the main compiler window; the RasPi Compilation Status area; as illustrated below.
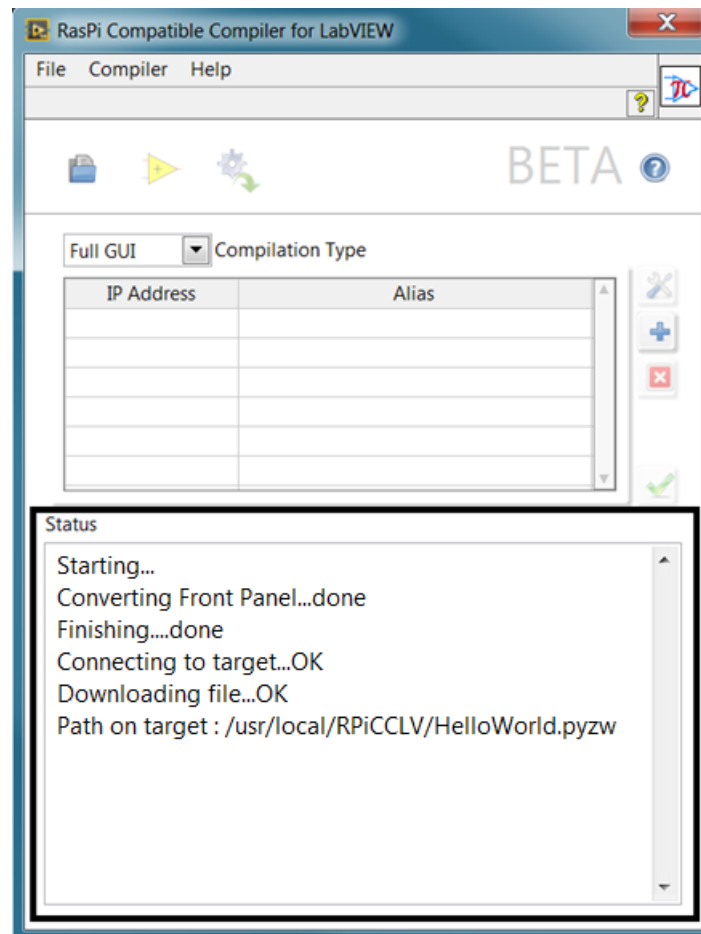
The Status area is where the messages generated by the compiler, during the process of compilation and download of the loaded VI to the target will be displayed. Also, any potential errors generated during the compilation process will be displayed on this indicator.

One important point to be made about the deployment process is that the Beta version of the compiler allows for multiple VIs to be deployed to the target. For example, assume one compiles and downloads a VI named HelloWorld.vi to the Raspberry Pi target. This VI, as will be seen on the next section, will be available to be executed embedded in the RasPi target. If the user compiles and downloads another VI named HelloBeautifulWorld.vi to the target, both VIs be available to be executed on the target.

Compiling and Downloading an Example VI

Now that you understand all resources of the compiler application, it is time to download your first LabVIEW VI to your configured RasPi target. From the application File menu, navigate to Examples->Hello World GUI.vi, as illustrated below.
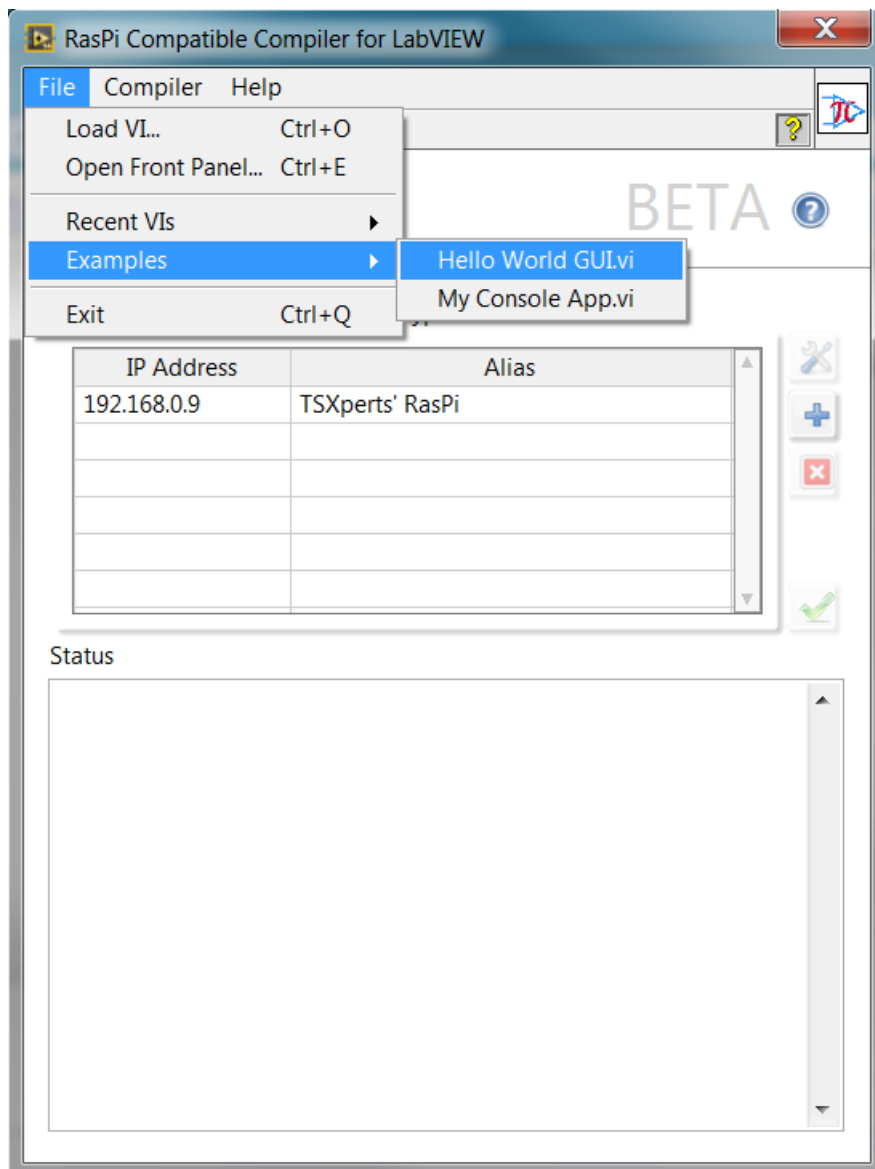
This will open the Hello World GUI.vi example and load the VI to be compiled and downloaded. Inspect the Hello World GUI.vi block diagram to get familiar with its code. Run the example VI on your development machine and notice how its user interface behaves. This will be useful in comparing with the behavior of the downloaded compiled VI, once we run it on the RasPi target.

Select your verified RasPi target from the list of targets. This will enable the Compile and Download button. Click the Compile and Download button and wait until the compilation finishes. Once it is complete, the compiled VI is downloaded to your RasPi target. Follow the steps included in the Running a Deployed VI on your RasPi section to run the downloaded VI on your RasPi target.

Running a Deployed VI on your RasPi

Once you press the Compile and Download button on the Raspberry Pi Compatible Compiler for LabVIEW main screen, the application will compile and download the selected VI to the Raspberry Pi target you have selected from the list of valid targets. At the end of the process; you will notice a message on the Status window indicating the Path on the target where the VI got downloaded to. That is where you will need to navigate to, on your Raspberry Pi, in order to run your deployed VI. This guide will assume the VI got deployed to /usr/local/RPiCCLV; but you can follow the same steps on any other directory the VI got deployed to.

On you Raspberry Pi, click on the File Manager icon on the top left of your Raspberry Pi screen as shown below.
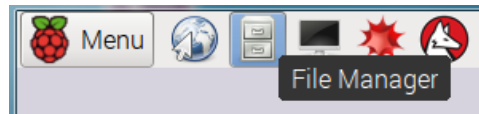
This will open the RasPi File Manager, an application similar to Windows Explorer on a Windows computer. Using the File Manager, navigate to the directory your VI got deployed to, as illustrated below.
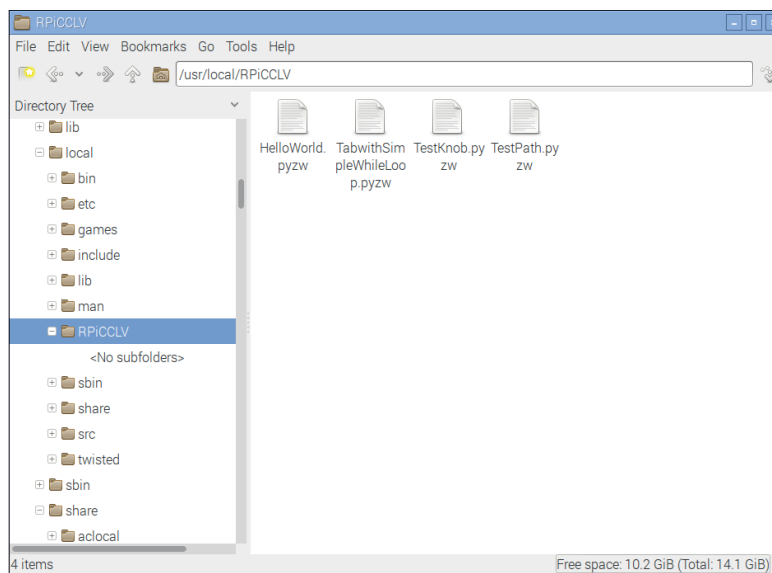
Next, right click at the RPiCCLV folder and select Open in Terminal as illustrated below.
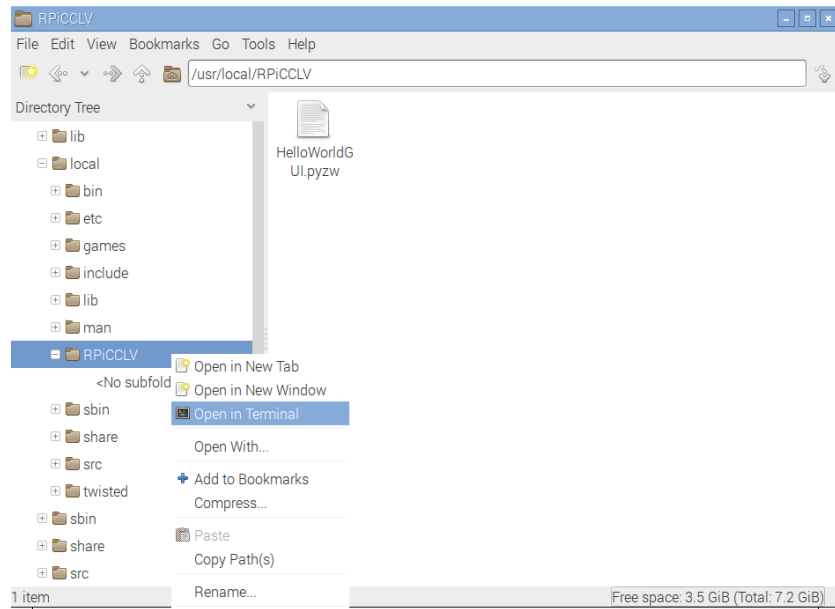
This will open a Shell Terminal screen on your Pi. On that screen, type the following command: **sudo ./<VI Name>.pyzw**. The illustration below runs a HelloWorld VI that got deployed to the RasPi target.
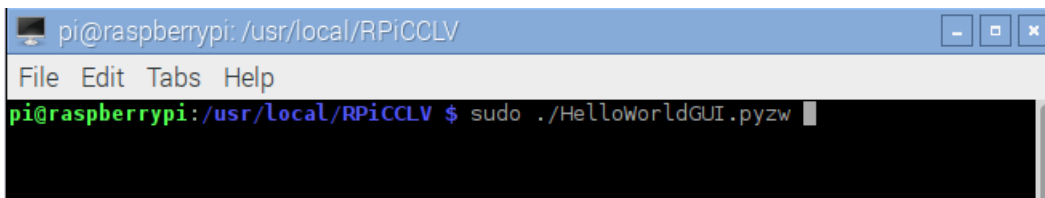
Once you press Enter to run the command, the HelloWorld deployed VI will open.

Simply clicking at its Run button will run the VI, exactly like you would do on regular LabVIEW.

Supported Compilation Types

The last important feature of the main compiler application window to discuss is the supported compilation types. There are two types of compilation types: Full GUI and Console App. The selection of which type to be executed by the compiler is done via the Compilation Type drop down control.

The Full GUI compilation type, as the name indicates, will compile both the VI block diagram and front panel as part of the compiled VI to be downloaded to the RasPi target. This will allow the RasPi to function basically as a regular computer running LabVIEW VIs. Full-fledged GUI VIs will run on the RasPi if that compilation type is selected. One thing that is important to make clear is that the RasPi will run what would be the corresponding version of a compiled executable VI. It would be the equivalent of a computer without the LabVIEW development environment running an Executable VI without its block diagram. The user will not be able to see the VI block diagram when the VI is deployed to the RasPi target, only its front panel.
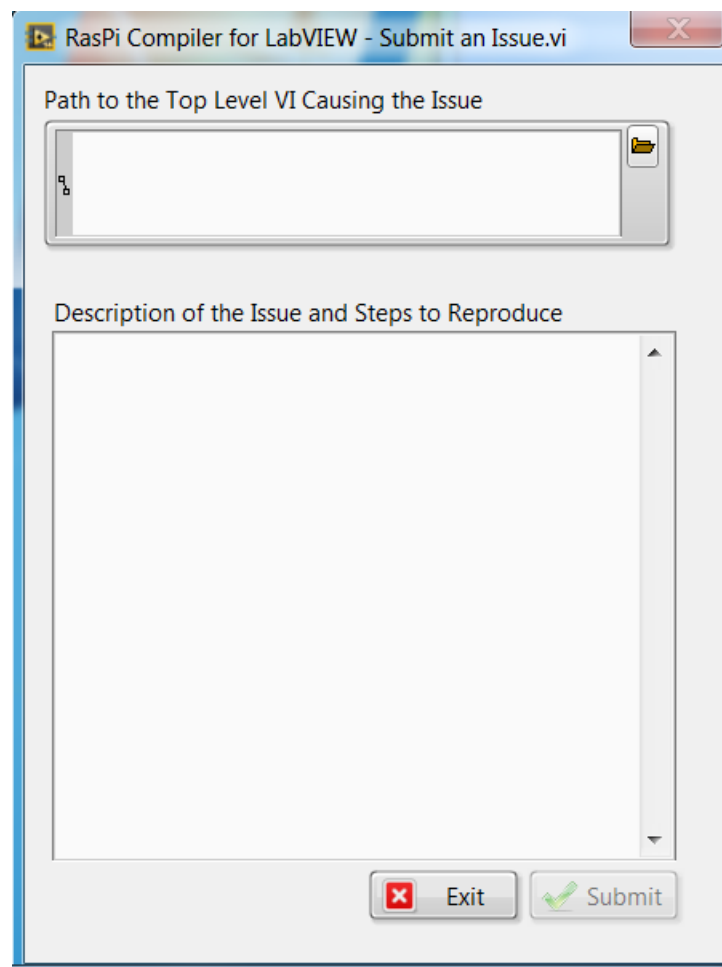
The Console App compilation type **is not currently available for the Beta version of the compiler**; however when it is made available it will strip the VI front panel and deploy only the VI block diagram code. One important feature to highlight as part of this compilation type is that the resulting console app created from the compilation will allow the user to call the app via command line and also pass arguments to it. The arguments would be the equivalent of the VI input terminals.

One interesting use case for this compilation type would be the creation of a Web service, in LabVIEW, to execute in the RasPi target. Some Web services don't require a user interface, but to respond to communication requests from other applications. Another example of use for such compilation type would be the creation of blocks of code in LabVIEW that only need to process data, without the

requirement to present any user interface. In such cases, the selection of the Console App type of compilation is preferable, as it strips the VI code to its most fundamental components, saving target resources and executing in a more optimal manner.

How to Report an Issue

This section describes the steps for submission of issues found with the compiler. In the LabVIEW Tools menu, selecting the option RasPi Compiler for LabVIEW - Submit an Issue will launch the following application screen.


This application allows the user to point to the top level VI that generated the issue as well as enter a thorough description of what the issue is as well as the steps to reproduce it.

Once the information above is filled in, the Submit button is enabled.

Clicking the Submit button triggers the application to package all needed information related to the VI being reported and send that information to the Raspberry Compatible Compiler for LabVIEW support channels. It is required the development PC that is running the compiler product has internet access for the submission to be successful.

Features not Included in the Beta Release that will Be Included on Version 1.0

All the LabVIEW function primitives and front panel controls that are currently included in the Beta release have been packaged into a separate LabVIEW palettes; for the convenience of the user. The following figure illustrates where to find the control palette for the RasPi compiler product.
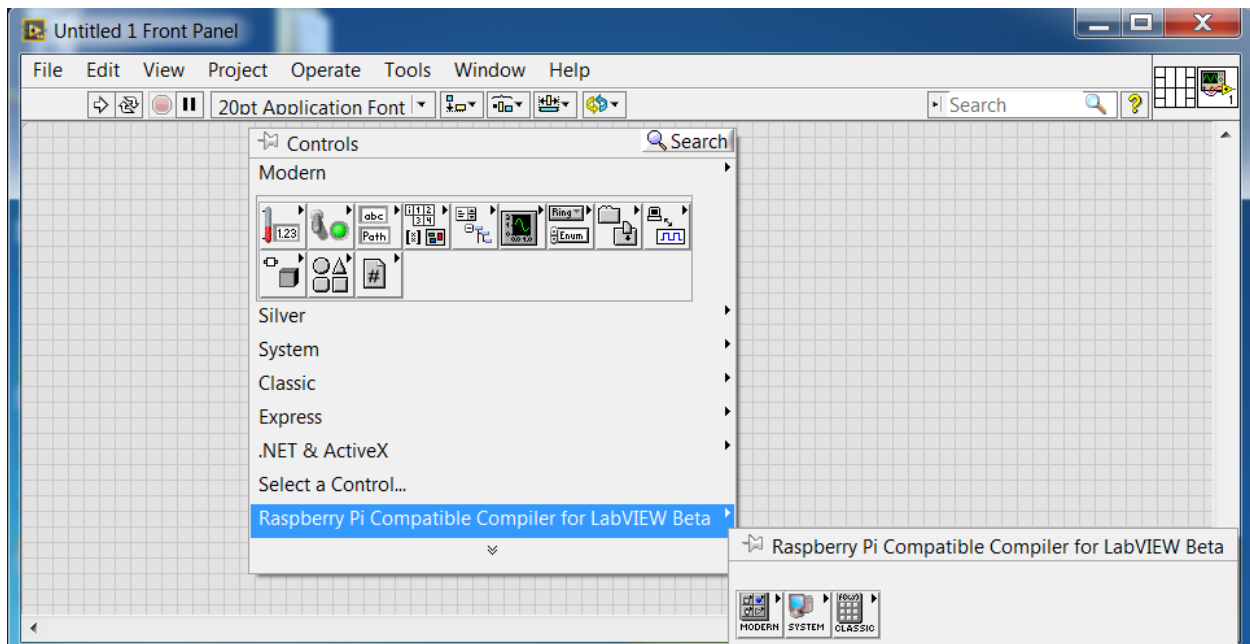
As one can see; there are three control subpalettes available for the user; Modern, System and Classic. The user is free to use any control included in these three sub-palettes. The use of controls not included in these palettes will generate a compilation error.

Similarly, a special palette containing all LabVIEW functions supported by the compiler can be accessed from the LabVIEW block diagram. The following figure shows the location of the RasPi compiler function palette.
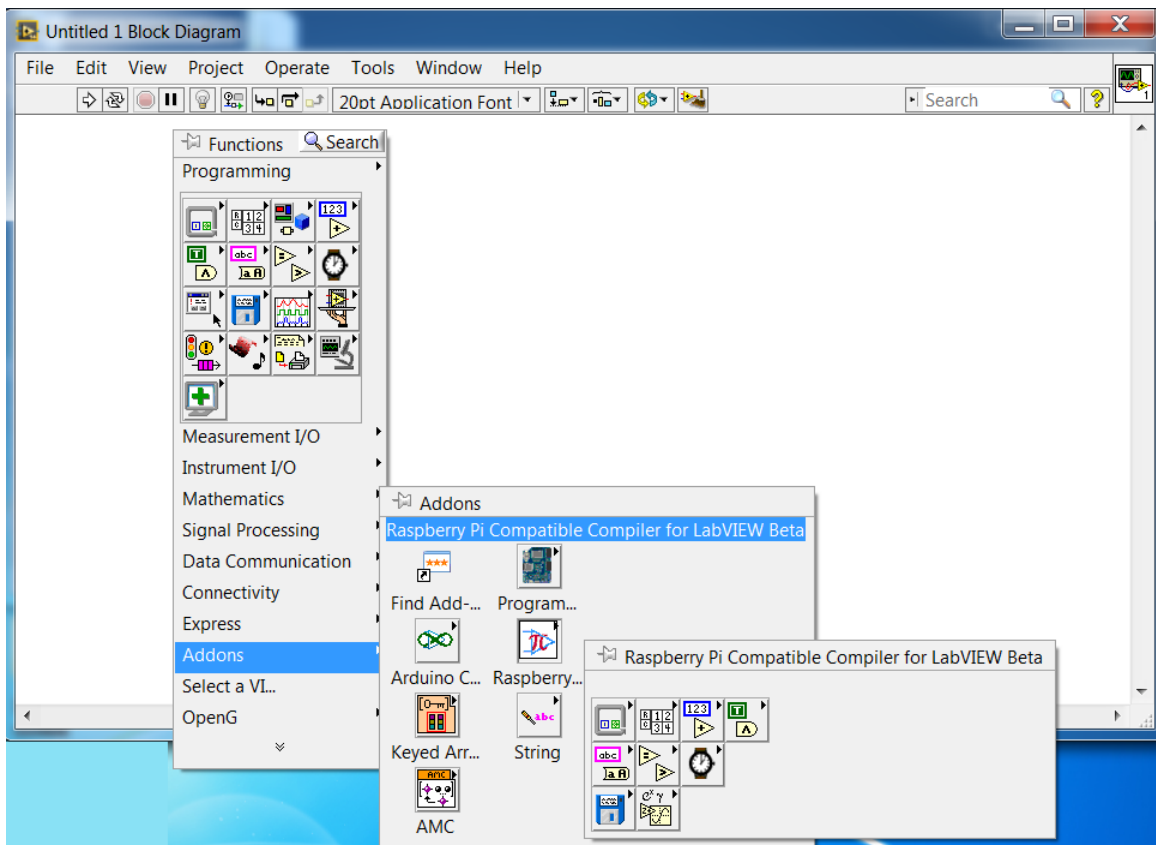
The use of any LabVIEW primitive not included in the Raspberry Pi Compatible Compiler for LabVIEW will generate a compilation error.

There are some features that will be included in version 1.0 of the compiler that are not currently part of the Beta release. We understand these features are important, but as we will need extra time to complete their implementation and wanted to get folks from the community using the compiler sooner than later, we have decided to release the Beta version without them. This Wiki Page shows the list of these features.

Differences with Regular LabVIEW and Features not Currently Supported

It is important to highlight that the deployment of LabVIEW VIs to a Raspberry Pi target involves the migration of an application that natively runs on one operating system onto another operating system. Extreme care has been taken in order to maintain the look and feel of LabVIEW front panels as much as possible, once the operating system migration is complete. The vast majority of LabVIEW front panel controls, once deployed, will look exactly as they do on the development machine. However, it is possible that some controls may have subtle differences once deployed to the Raspberry Pi target.

Another important point to make is that LabVIEW is a mature product that has been improved upon for over thirty year. The Raspberry Pi Compatible Compiler for LabVIEW development team worked extremely hard to implement all right click options and features that are part of the supported LabVIEW primitives that have been included as part of the compiler. However, some of these options and features weren't implemented at the time this user guide is being created. This Wiki Page includes a comprehensive list of all known differences and exceptions one can expect when programming a Raspberry Pi target with LabVIEW using the Raspberry Pi Compatible Compiler for LabVIEW.

FAQ

This section includes a list of frequently asked questions that may help you debugging common issues with your environment.

1) After installing the VIP package file in LabVIEW I launch the Raspberry Pi Compatible Compiler for LabVIEW application and get a VI with a broken Run arrow. What am I doing wrong?

Answer: There are two possible issues you may be facing. The first one is incompatibility between the version of the VIP package you have downloaded and the current version of LabVIEW you have installed the package to. There are multiple VIP files, each one compiled for a specific version of LabVIEW. For example, the file that ends with the suffix LV 2014 is a package that shall be installed to LabVIEW 2014. Another possible issue is that your Operating System user account doesn't have administrator privileges. If that is the case, if you installed the VIP file you will also see the broken Run arrow. The VIP package requires installation from a user account with administrator privileges.

2) Even though I can successfully ping my Raspberry Pi from my development machine, the Compiler application can't successfully connect to the Pi. What can be going wrong?

Answer: The most immediate thing to try is to reboot your Pi and try the connection from the Compiler application once the Pi complete power cycling. If that doesn't work; check if your LabVIEW development environment is allowed by your development machine firewall to make external connections. If that looks good, check if there are other applications on your computer that may have set up separate firewalls; typically antivirus Software, that may be preventing the communication between the two.

3) The compiler no longer works after I have setup a Wifi dongle on my Pi, even though I can successfully ping its IP address from my development machine. What is wrong?

Answer: You should setup your Wifi dongle prior to installing the linux artifacts of the Raspberry Pi Compatible Compiler for LabVIEW on your Pi. If you have already installed the compiler, you will need to reimage your SD Card with the suggested Raspbien distribution per the How to Setup the SD Card section and configure the Wifi dongle prior to installing the compiler package.

4)

Support

If the support request is related to a problem with a specific VI, the most efficient way to request support is by using the Submit an Issue Screen, as described in the Section named [How to Report an Issue](#).

If you have a question, comment or suggestion, we would love to hear from you at the e-mail: lvforpisupport@tsxperts.com.