Application Report
SPRACB2–October 2017

Qt Training: Multipage, Resizable Graphical User Interfaces Containing Media

Sahithi Bonala

ABSTRACT

Qt is a GUI development tool that is a component of the Sitara™ SDK. This application report is designed to help engineers become acclimated with Qt Creator, the Qt integrated development environment (IDE) for creating Qt Widgets Applications. The application report goes through the steps necessary for creating GUIs with the following characteristics: resizable, multipage, and contain media. This document has been validated on ti-processor-sdk version 04.00.00.04 using Qt Creator version 5.7.0 on a 64-bit, Ubuntu® 14.04 Virtual Machine.

Contents

List of Figures

Trademarks

Sitara is a trademark of Texas Instruments.
Ubuntu is a registered trademark of Canonical Ltd.
Linux is a registered trademark of Linus Torvalds.

1 Software and Hardware Configurations

This section discusses the software and hardware configurations of the Qt Creator.

1.1 Hardware

- AM335x EVM-SK (TMDSSK3358)

NOTE: This application report is applicable to all Sitara boards with an LCD display or HDMI/DVI. However, the steps related to serial and Ethernet connections may differ.

- Ethernet switch connecting the AM335x EVM-SK and Linux® host
- USB cable connection between the AM335x EVM-SK and Linux host using the micro-USB connector (J3/USB0)

NOTE: The AM335x EVM uses a standard DB9 connector and serial cable. PCs may require a USB-to-serial cable, because newer laptops do not have serial ports. Alternatively, you can telnet or SSH into the target over Ethernet instead of using a serial terminal for target interaction.

- 5-V power supply (typically provided with the AM335x EVM-SK)

1.2 Software

- Linux host PC, configured as per the requirements listed in the Linux Host Configuration wiki page
- Sitara Linux SDK (installed)

NOTE: This application note assumes the latest Sitara Linux SDK (ti-processor-sdk version 04.00.00.04) is installed in /home/sitara. If you use a different location, modify the following steps accordingly.

- SD card with Sitara Linux SDK (installed)

NOTE: For help creating a 2-partition SD card with the SDK content, see the create_sdcard.sh script page.

- Qt Creator 5.7.0 (installed on the Linux host)

NOTE: You can download Qt Creator from the open source distribution version of Qt.

1. Qt Creator will download as a .run file.
2. Make the file executable by running the following commands. These commands should launch the Qt installer.

```
chmod + <qtfile>
./<qtfile>
```

3. Extract the package. Qt creator is under the Tools/Qt5.7.0 folder. In some cases, it may also be under the /opt folder. Type cd /opt/Qt5.7.0 into the command line to locate the file.

1.3 Supported Platforms and Revisions

The following platforms are system tested to verify proper operation.

- AM57x
- AM43xx
- AM335x
- AM37x/ AM35x
- AM180x

2 Qt IDE and Widgets Application Setup

This section covers setting up the Qt Creator.

Key Points:

- Setting up Qt Creator to find your tools
- Setting up Qt Creator to communicate with the target platform
- Creating a project and deploying it using Qt Creator

2.1 Setting Up Qt Creator

Follow these steps to set up the QT Creator:

1. Source the environment setup file to ensure all the paths are setup correctly:

```
source /home/sitara /AM335x/ti-processor-sdk-linux-<machine>-<sdk-version>/linux-devkit/environment-setup
```

2. Launch Qt Creator:

```
:  cd /home/sitara/Qt5.7.0/Tools/QtCreator/bin
: ./qtcreator
```

Qt Creator should be up and running now (see Figure 1).

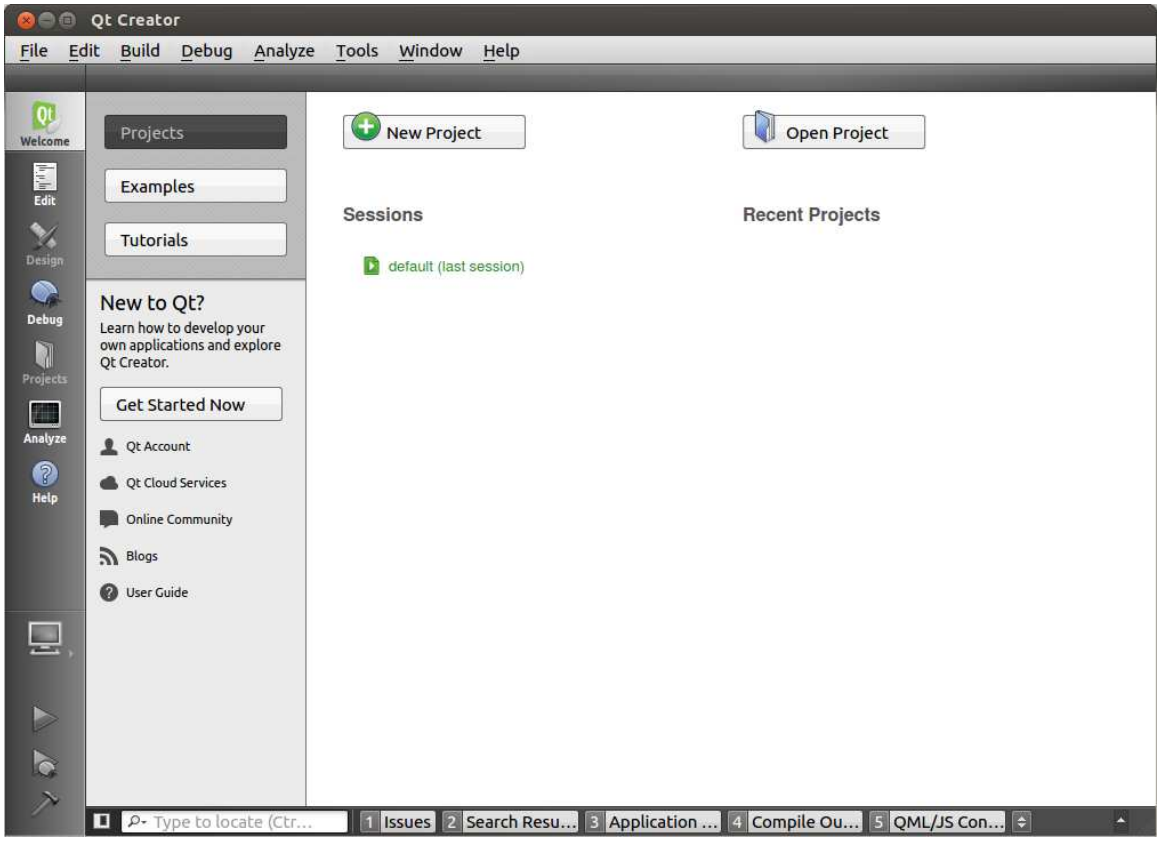

Figure 1. Qt Creator Landing Page

Follow these steps to set up the Qt creator to configure qmake. From the Qt Creator main menu shown in Figure 1:

1. Select Tools → Options...
2. On the left side vertical menu bar click the Build & Run button.
3. Click the Qt Versions tab under the Build & Run option.
4. Remove any previous versions that may already exist to ensure you start with a clean configuration.
5. Click the Add... button on the right.
6. Navigate to /home/sitara /AM335x/ti-processor-sdk-linux-<machine>-<sdk version>/linux-devkit/sysroots/x86_64-arago-linux/usr/bin/qt5.
7. Select qmake then click Open.
8. Double click on Version Name and give the Qt Version a descriptive name, such as Qt 5.7 Sitara (see Figure 2).

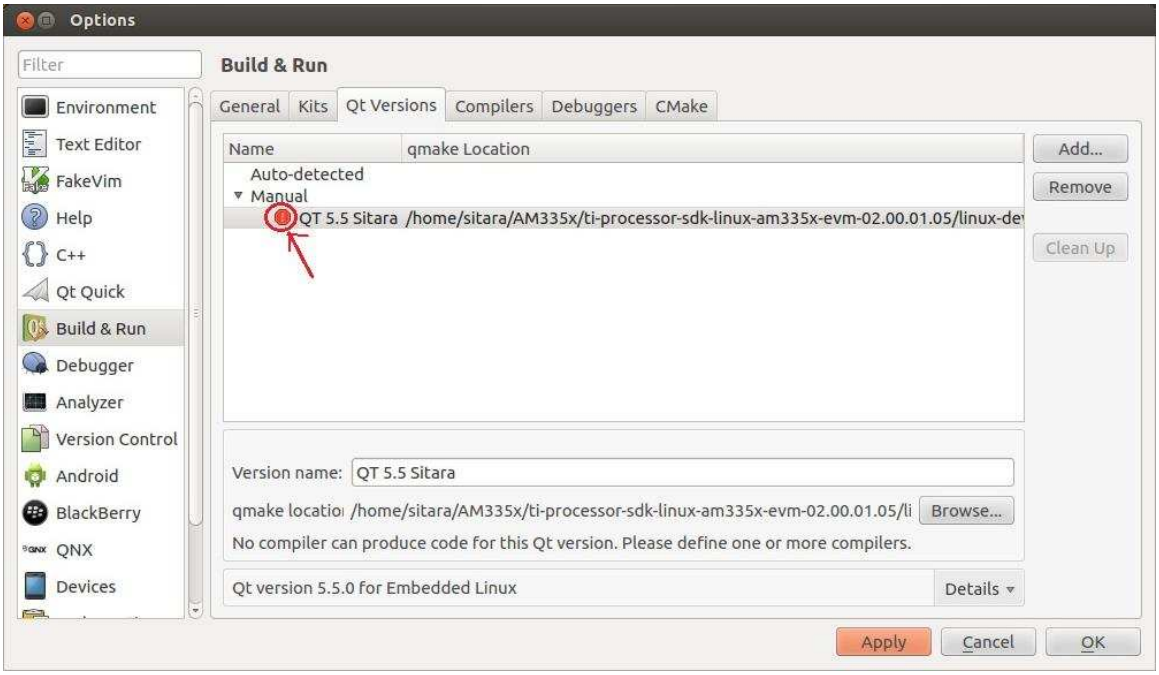

Figure 2. Qt Version Configuration

9. Click the Apply button to save your changes.

Follow these steps to set up the toolchain (see Figure 3).

1. Click the Compilers tab under the Build & Run option.
2. Click the Add button on the top right corner and add an instance of GCC.
3. Change the name to *arm-linux-gnueabihf-gcc*, this can be done by editing the Name field.
4. For the Compiler Path, select Browse then:
    1. Navigate to /home/sitara/AM335x/ti-processor-sdk-linux-<machine>-<sdk version>/linux-devkit/sysroots/x86_64-arago-linux/usr/bin.
    2. Select arm-linux-gnueabihf-gcc and click on Open.
    3. Make sure to click on Apply to save your changes.

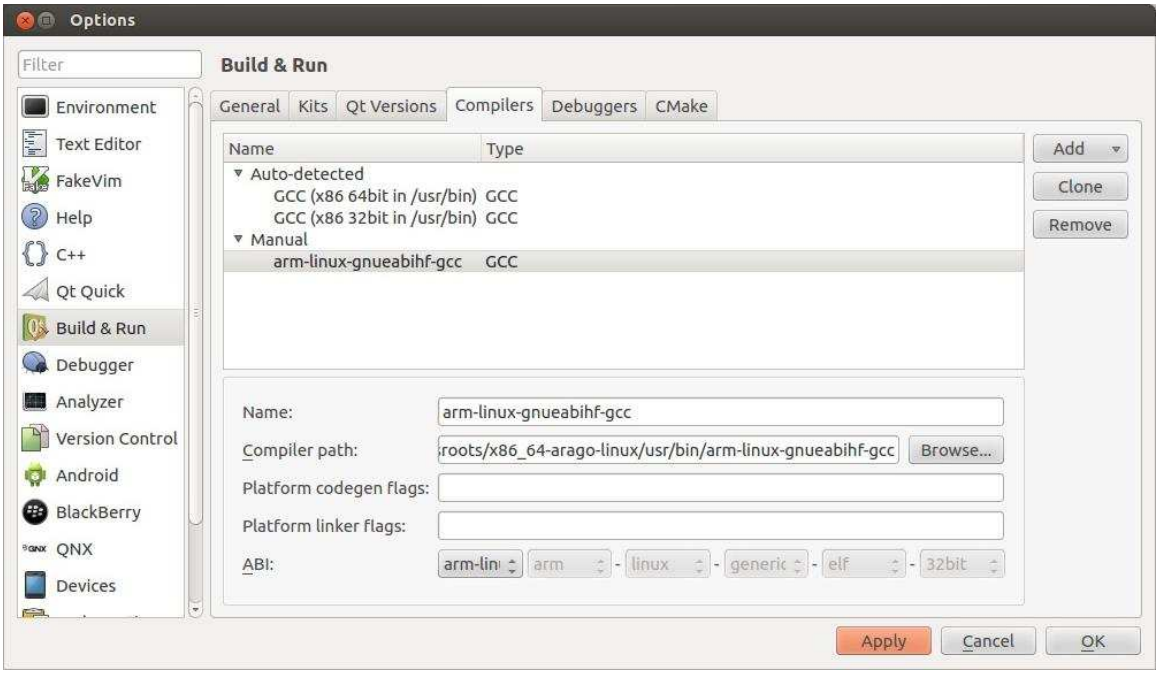

Figure 3. Qt Compiler Configuration

Follow these steps to set up the Debuggers (see Figure 4).

1. Click the Debuggers tab under the Build & Run option.
2. Click the Add button in the top right corner.
3. Change the name to *GDB Engine*, this can be done by editing the Name field.
4. For the Debugger Path, select Browse then:
    1. Navigate to /home/sitara/AM335x/ti-processor-sdk-linux-<machine>-<sdk version>/linux-devkit/sysroots/x86_64-arago-linux/usr/bin.
    2. Select *arm-linux-gnueabihf-gdb* and click Open.
    3. Ensure to click on the Apply button to save your changes.

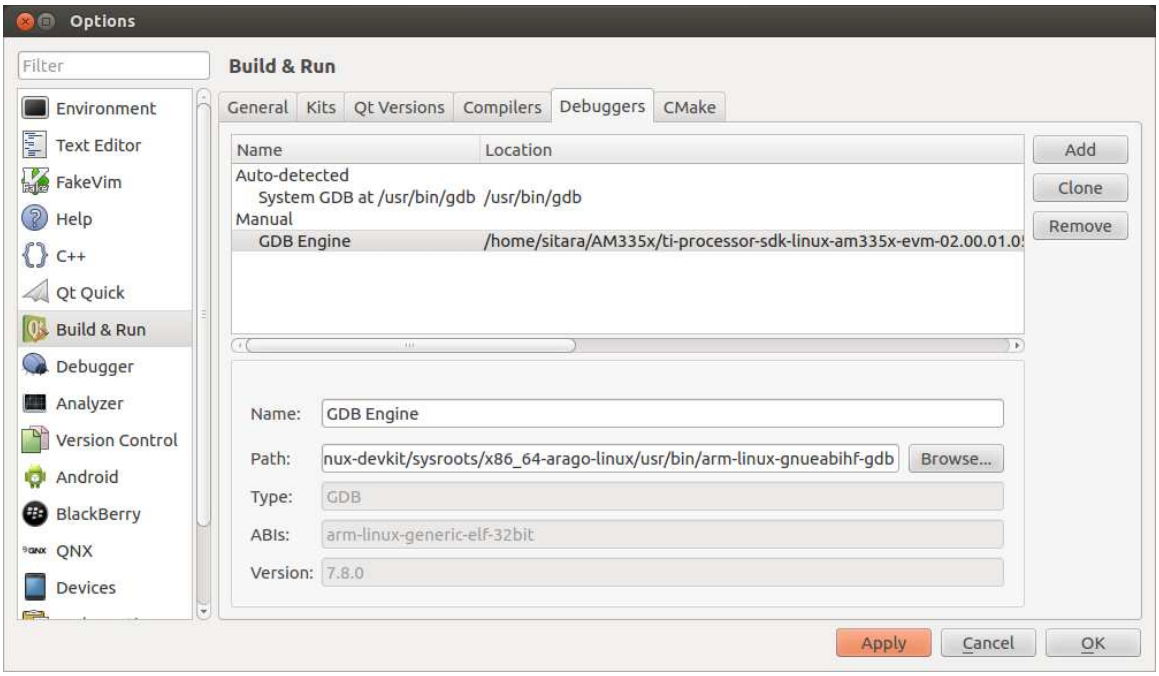

Figure 4. Qt Debugger Configuration

Next, click the Kits tab under the Build & Run option (see Figure 5).

1. Click the Add button.
2. Change the name to give the device a unique name: AM335x EVM.
3. Select Device type Generic Linux Device instead of Desktop.
4. Select Compiler arm-linux-gnueabihf-gcc instead of the host gcc.
5. For Debugger select GDB Engine.
6. For Qt Version select Qt 5.7 Sitara.
7. Click Apply to register the options.

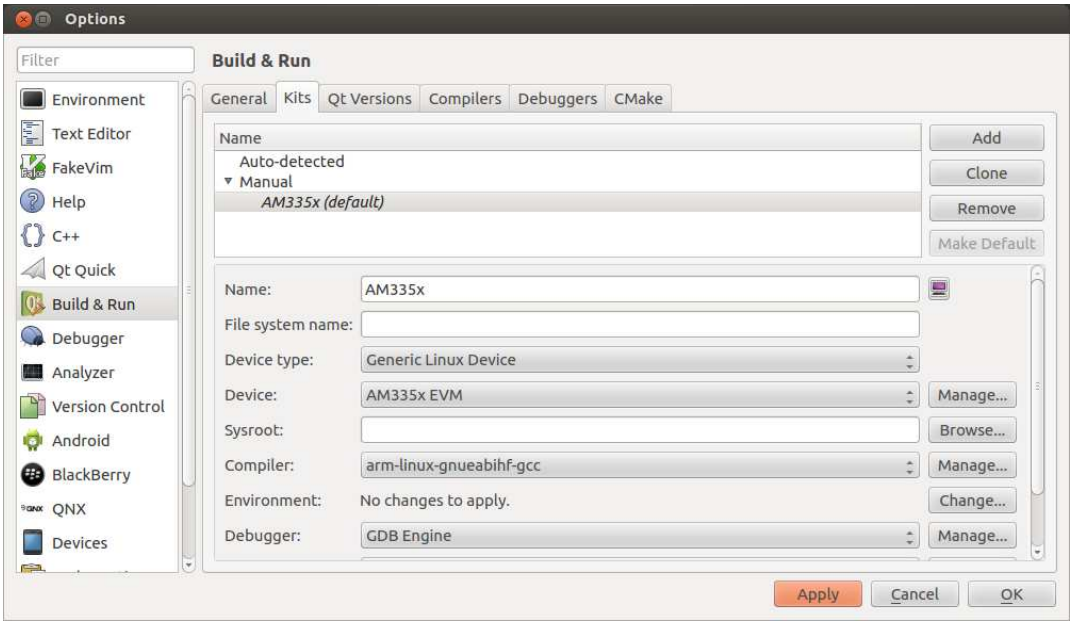

Figure 5. Qt Kit Configuration

2.2 Setting Up Target Platform

Follow these steps to set up the target. While still in the Tools → Options menu (see Figure 6):

1. On the left side of the window, select the Devices tab.
2. In Devices, click the Devices tab.
3. Click the Add... button in the top right corner.

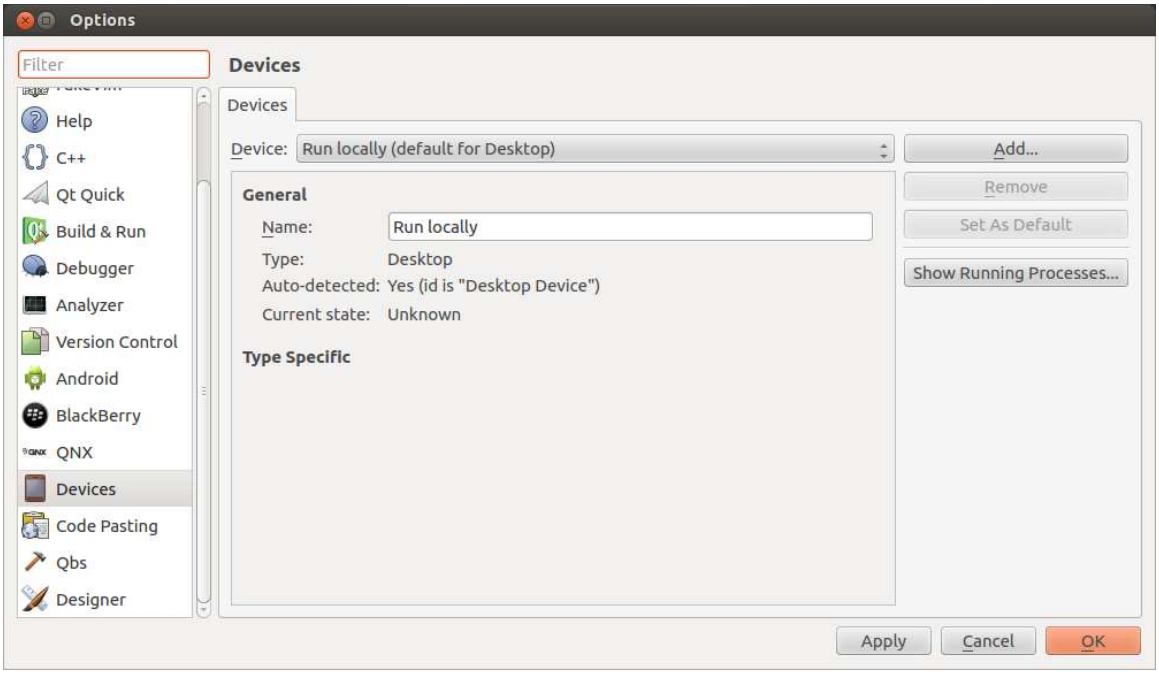

Figure 6. Qt Kit Configuration

4. Select Generic Linux Device and click on Start Wizard (see Figure 7).

Figure 7. Qt Device Configuration Wizard

5. The Device Configuration Wizard Selection Dialog box appears (see Figure 8), do the following:
   1. Type in the name of the device: AM335x EVM.
   2. Type in the IP address of the embedded Linux device. Use the IP address for your board, not the one shown in Figure 8.
6. For the user name field, use root (most TI boards have this username).
7. Make sure Authentication type is Password, and enter the root user's password (default is blank).
8. Click the Next button.

Figure 8. Qt Device Configuration Wizard

9. Click the Finish button. You should see that the target test passed (see Figure 9). You can close the Device Test window now.

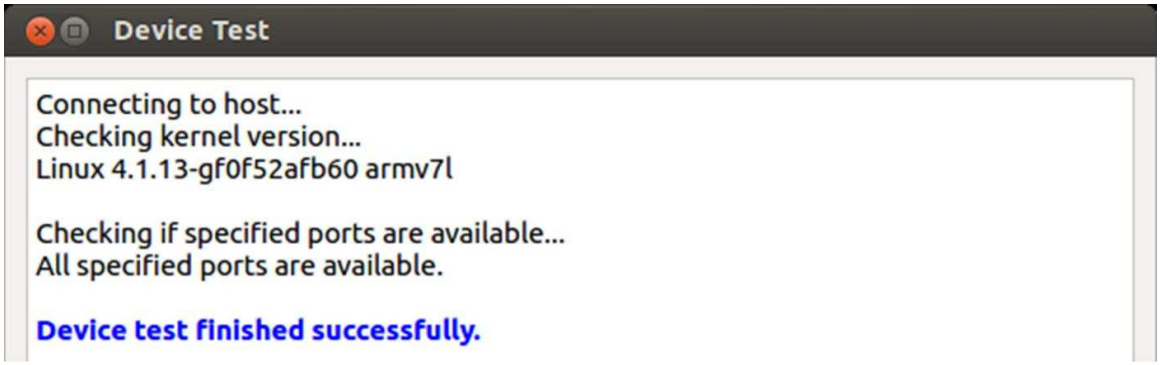

Figure 9. Qt Device Test

Now an SSH key must be set up so that the host can communicate with the target (see Figure 10), follow these steps:

10. Still under the Devices tab, click the Create New button in the Private key file field.

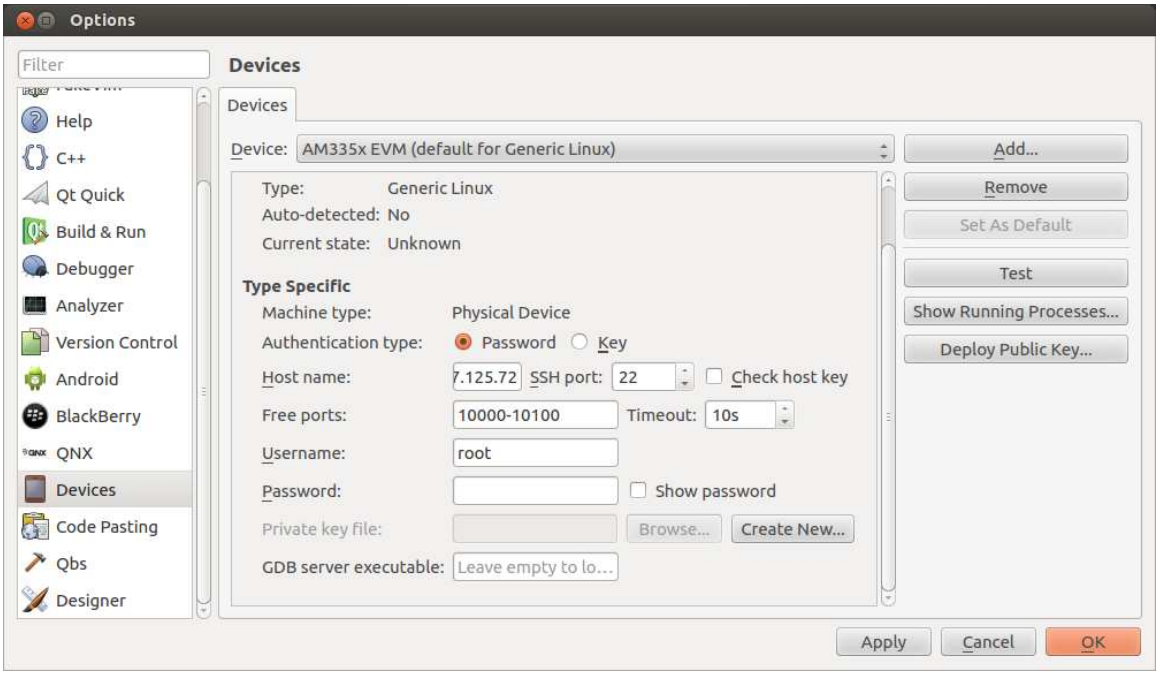

Figure 10. Qt Devices Tab

11. Complete the SSH key configuration with the following options selected (see Figure 11):
    - Key algorithm: RSA
    - Key size: 1024

    Then click the Generate and Save Key Pair... button.

Figure 11. SSH Key Configuration

12. Select the Do Not Encrypt Key File button on the next dialog box (see Figure 12).
    1. Under the Devices tab, now click the Deploy Public Key... button (see Figure 13).
    2. Select the file just generated (should be under /home/sitara/.ssh).

Figure 12. Password for Private Key

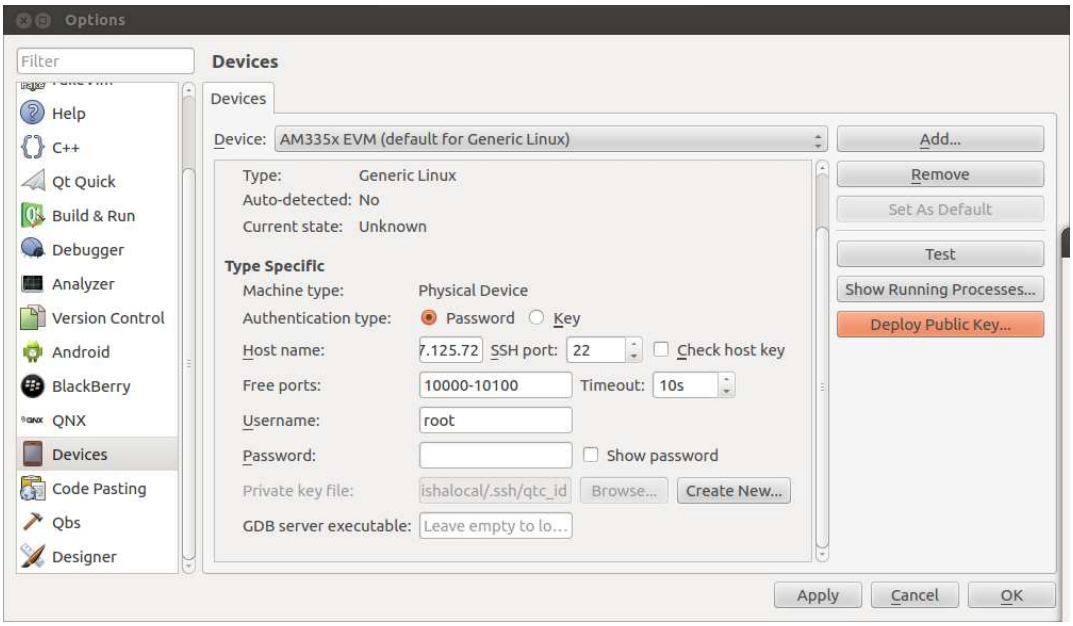

Figure 13. Deploy Public Key

13. Select the file *qtc_id.pub*, and click on Open. Shortly, a window should appear saying Deployment finished successfully (see Figure 14).

Figure 14. Successful Deployment

14. Close the window and click the Ok button to exit the Linux Devices window (see Figure 15).

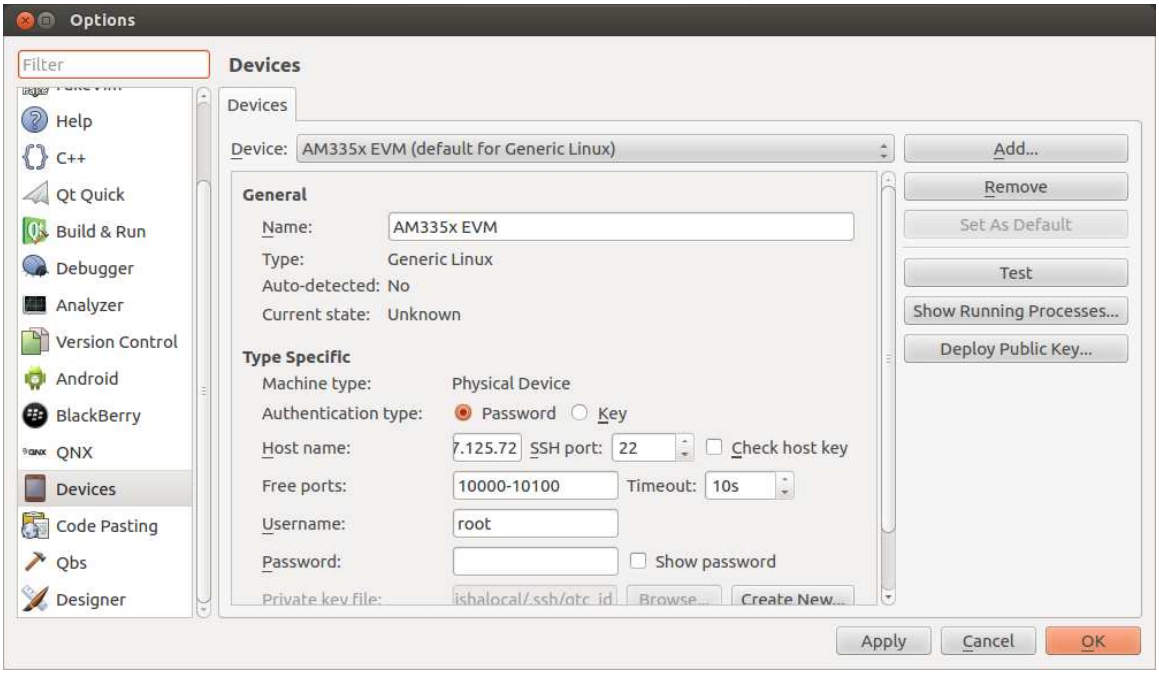

Figure 15. Closing Device Window

2.3 Creating a Project

Now that setup is complete, follow these steps to create a project, build it, and run it on the host:

1. Select File → New File or Project.
2. Select the Applications option under projects, then select Qt Widgets Application at the top (see Figure 16).
3. Click on the Choose button.

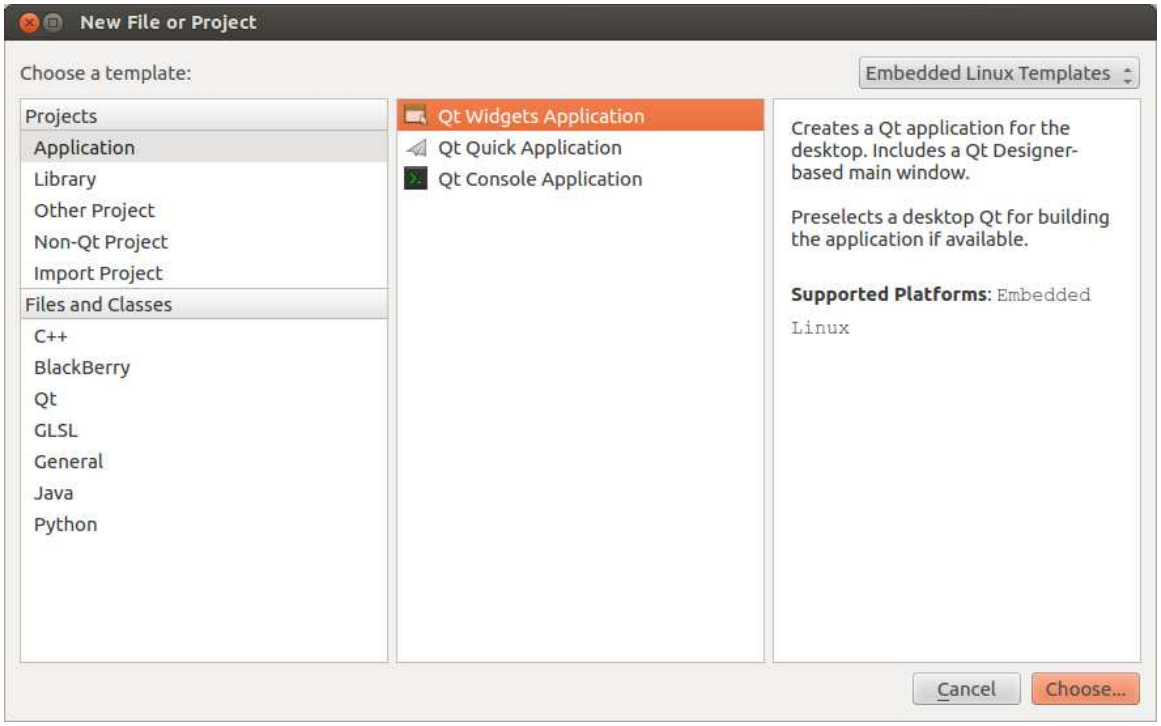

Figure 16. New Qt Widgets Application

4. Type in the name of the project as demoGUI (see Figure 17). This project will be built on in the next section.
5. Change the Create in value to /home/sitara (this project is created under /home/s-bonala), or the directory of your preference.
6. Click on the Next button.

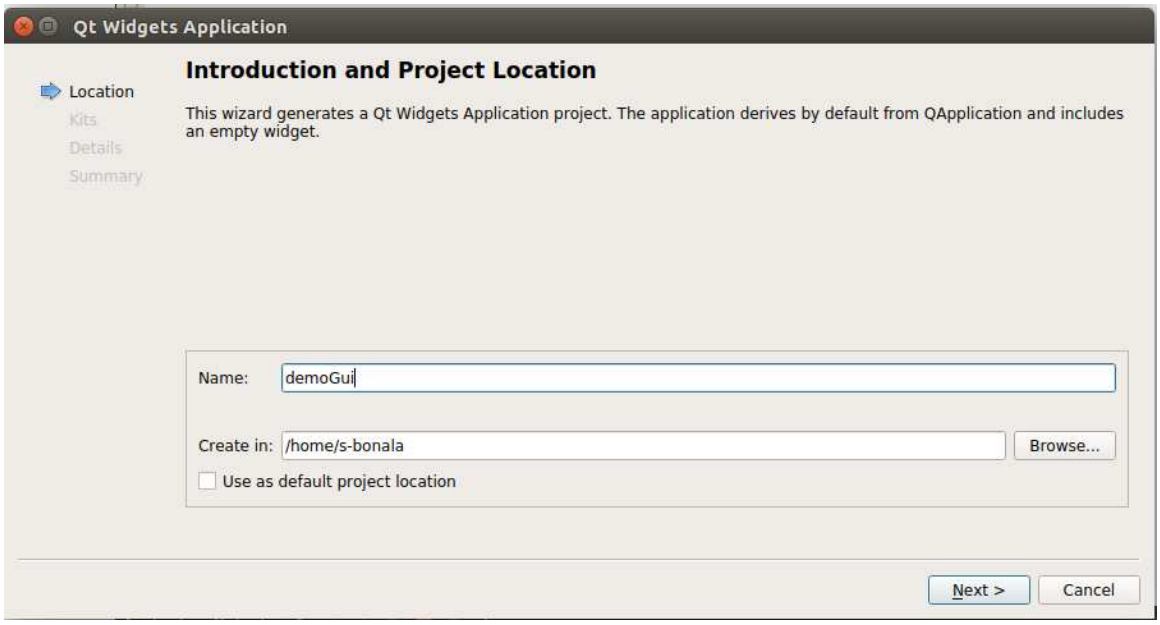

Figure 17. New Qt Project Name and Location

7. Click on the Next button again.
8. Type in demoGUI for the Class name (see Figure 18).

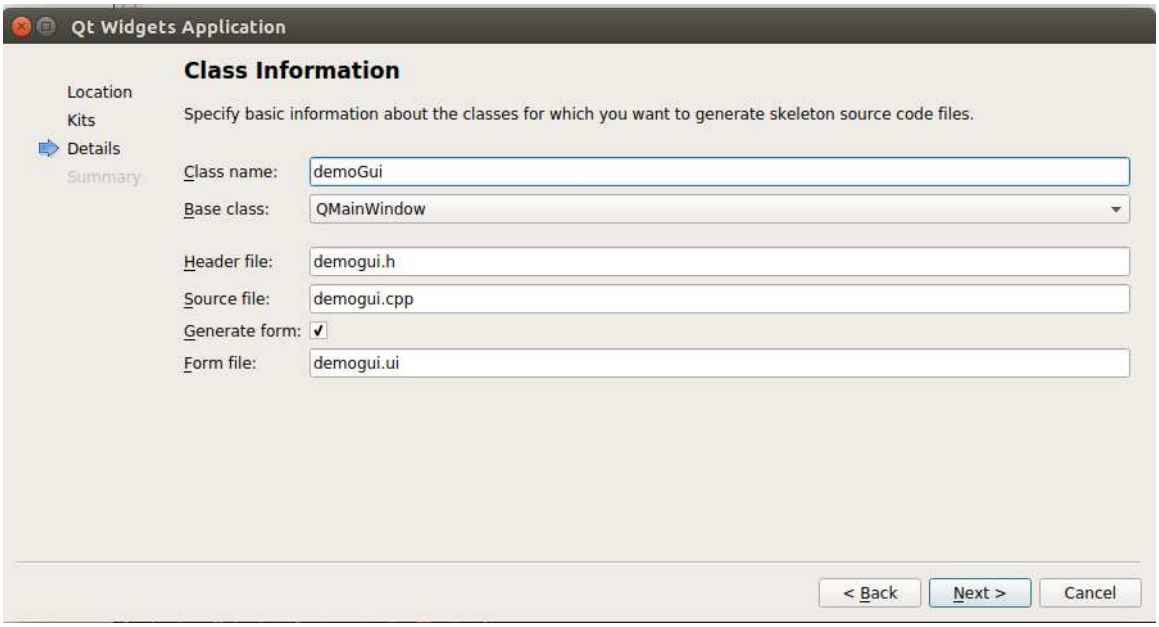

Figure 18. New Qt Project Class Information

9. Click the Next button.
10. Click the Finish button.

Set the remote path of the device so that the Qt Creator can deploy the GUI Application to the correct place on the device (see Figure 19):

1. Click the Edit button on the left side vertical menu, then click on demoGUI.pro.
2. Add the following two lines to the bottom of demoGUI.pro, as shown in Figure 19.

```
target.path += /home/root
INSTALLS += target
```

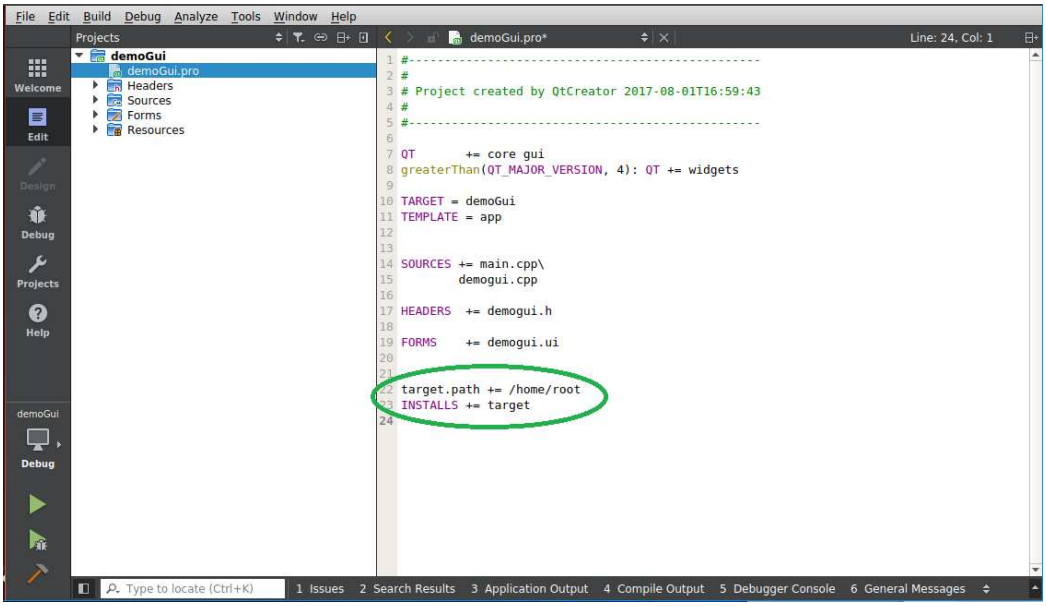

Figure 19. demoGUI.pro Edits

Check and update the build and run settings, as follows:

1. On the left side vertical menu bar select Projects.
2. Select the Build & Run tab, then select the Build button under AM335x.
3. Uncheck the Shadow build option (see Figure 20).

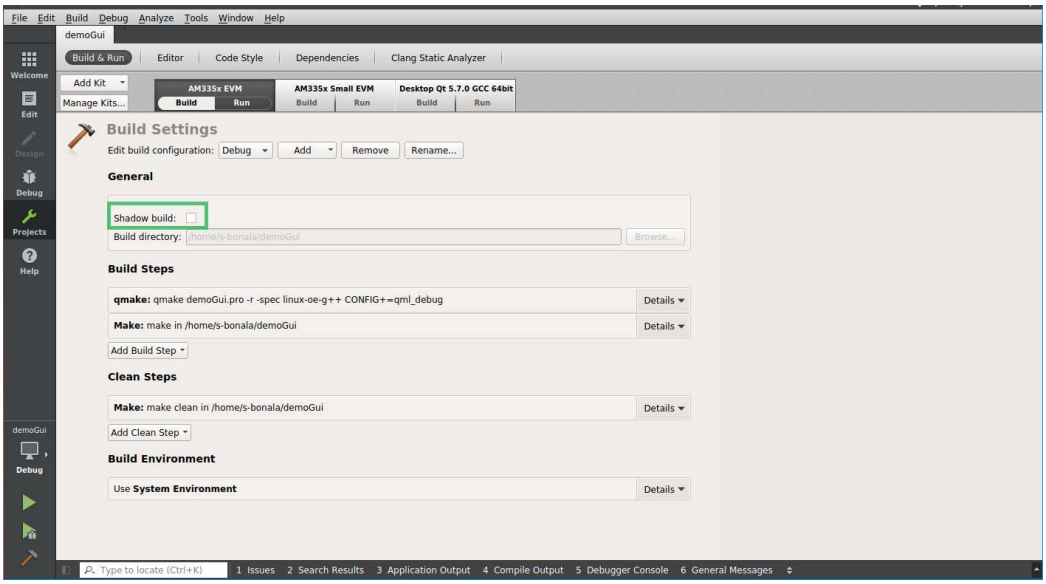

Figure 20. demoGUI Build Settings

4. Under the AM335x EVM, select the Run tab (see Figure 21).
5. Under the Deployment section in the Method field, select Add and Deploy to Remote Linux Host.
6. Under the Run section next to the Run configuration field, select Add and demoGUI (On Remote Device).

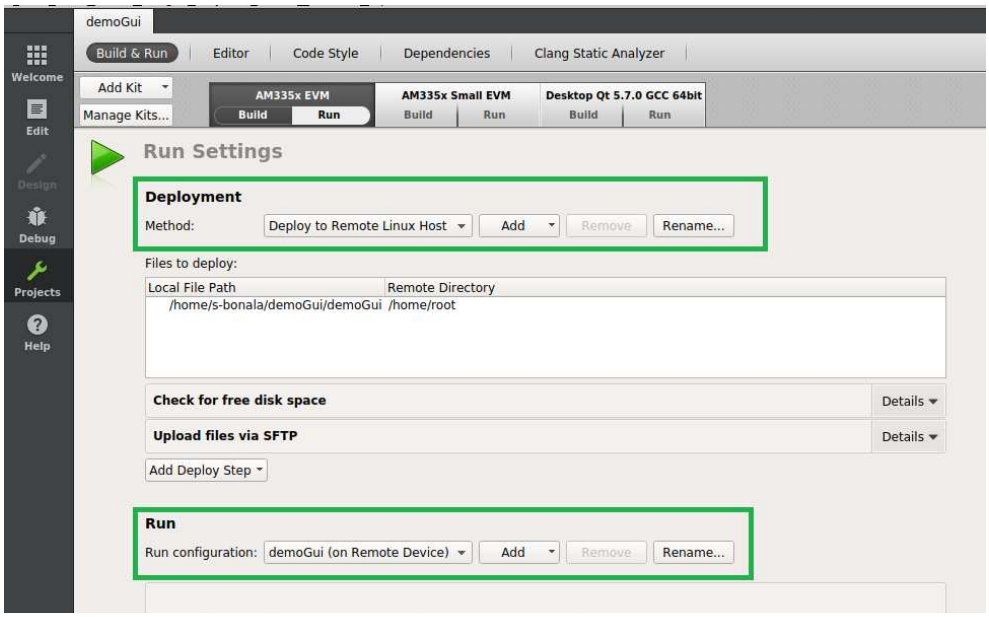

Figure 21. demoGUI Run Setting

Finally the tool is ready to run, click the Green Arrow on the bottom left to run the project. The screen on Figure 22 should appear on your EVM.

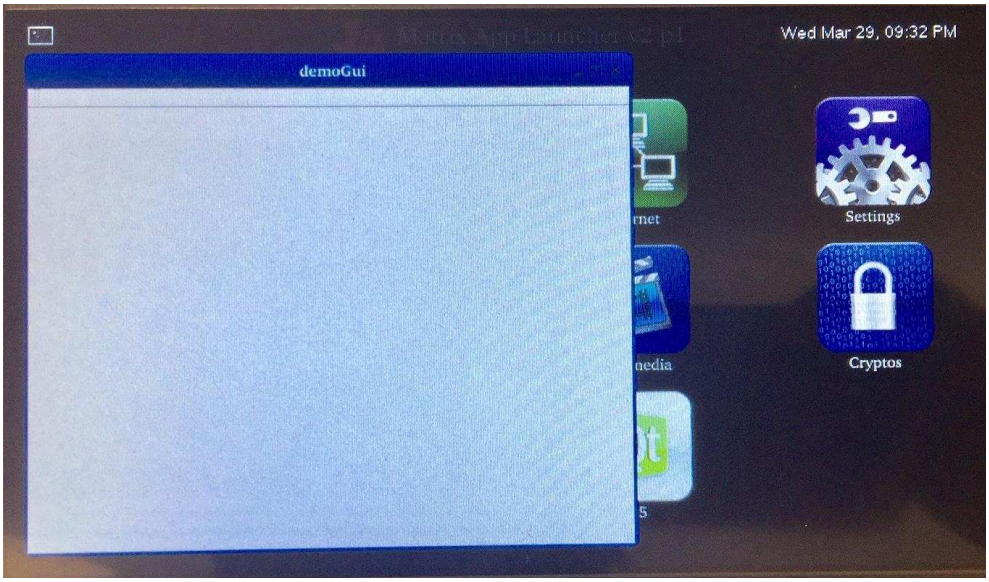

Figure 22. demoGUI Deployed to EVM

NOTE: If you receive the error *g++: Command not found* or *gnu/stubs-soft.h: No such file or directory*, navigate to tools → options → build & run → kits, then add *linux-oe-g++* to the Qt mkspec text box (see Figure 23).

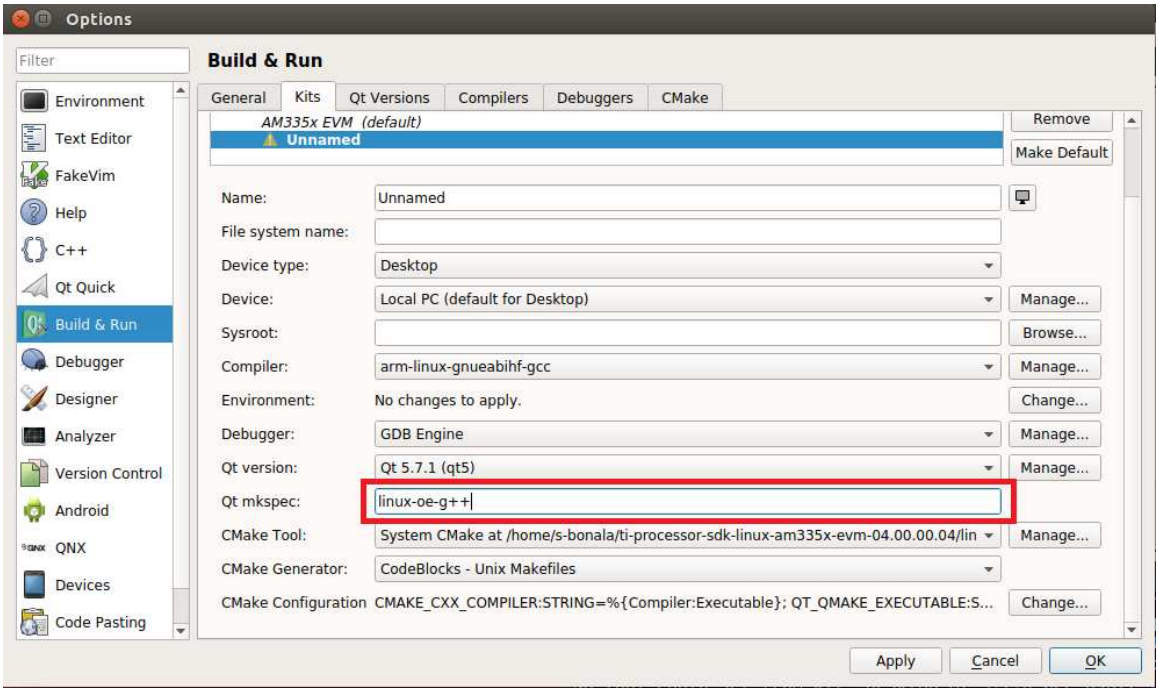

Figure 23. Qt mkspec Property

3 Creating a Multipage Resizable Application Using Qt Widgets

This section covers how to create a Qt Widgets application that has multiple pages, and can scale and resize for any sized screen.

Key Points:

- Applying layouts to make the GUI resize properly
- Using horizontal and vertical spacers
- Using the Property Editor to adjust the sizePolicy of onscreen elements

3.1 Applying Layouts For Resizing

Follow these steps to create a Qt Widgets application.

1. Open the demoGUI project created in Section 2, (see Figure 24).
2. Launch the Qt Creator and navigating to File → Open File or Project.

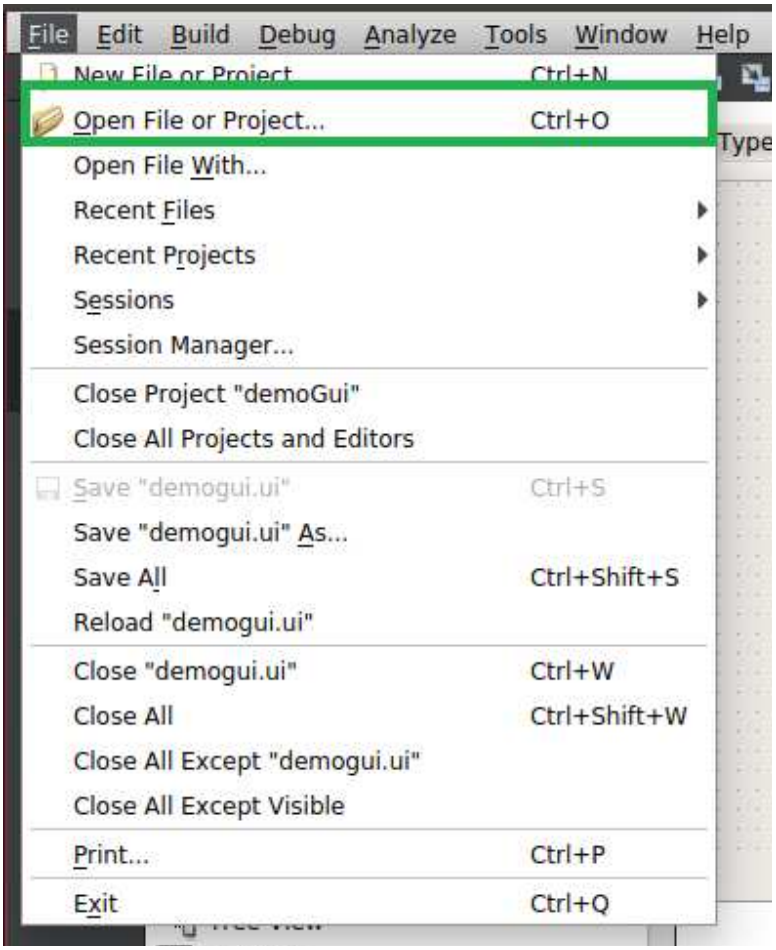

Figure 24. Opening Qt Project

3. Navigate to the demoGUI project folder, then select demoGUI.pro, and click the Open button (see Figure 25).

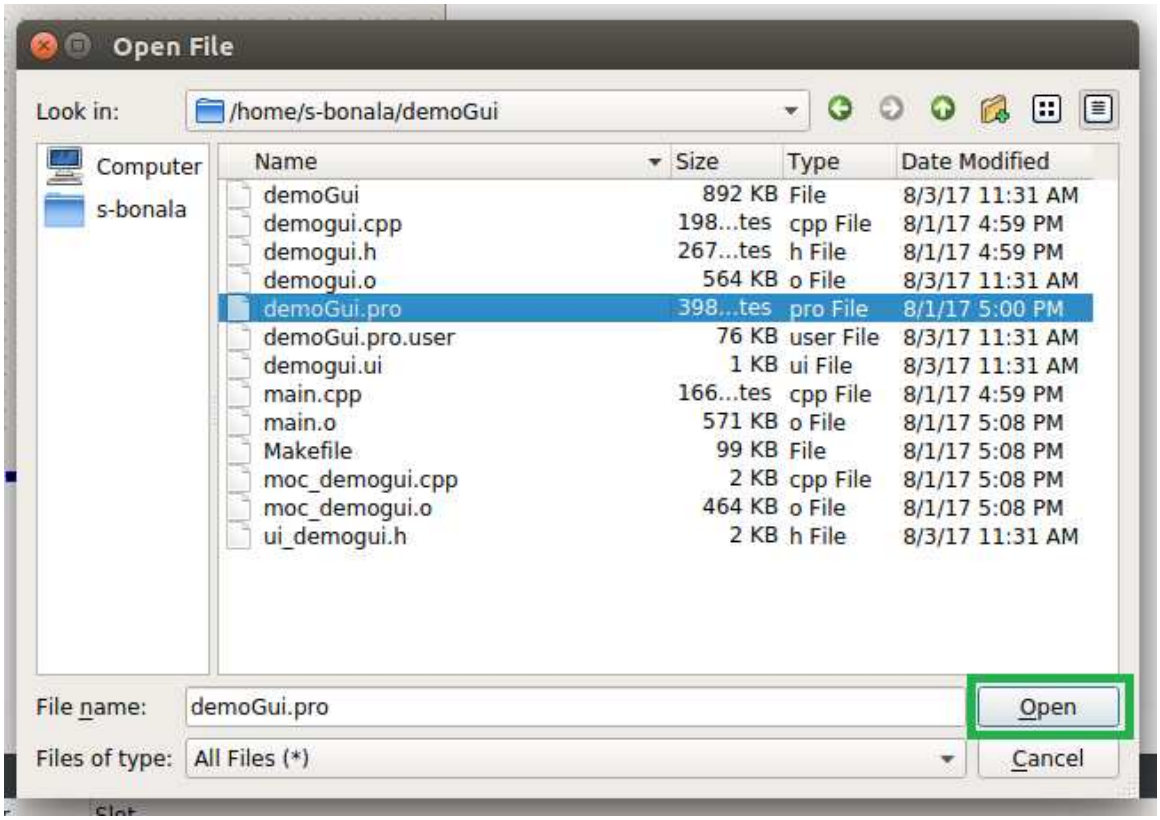

Figure 25. Opening demoGUI.pro

4. Under Forms, double click on demoGUI.ui. This brings up the widget editor.
5. Remove the menuBar from the GUI. The menuBar has Type Here written on it. Right click on the menuBar and select Remove MenuBar (see Figure 26).

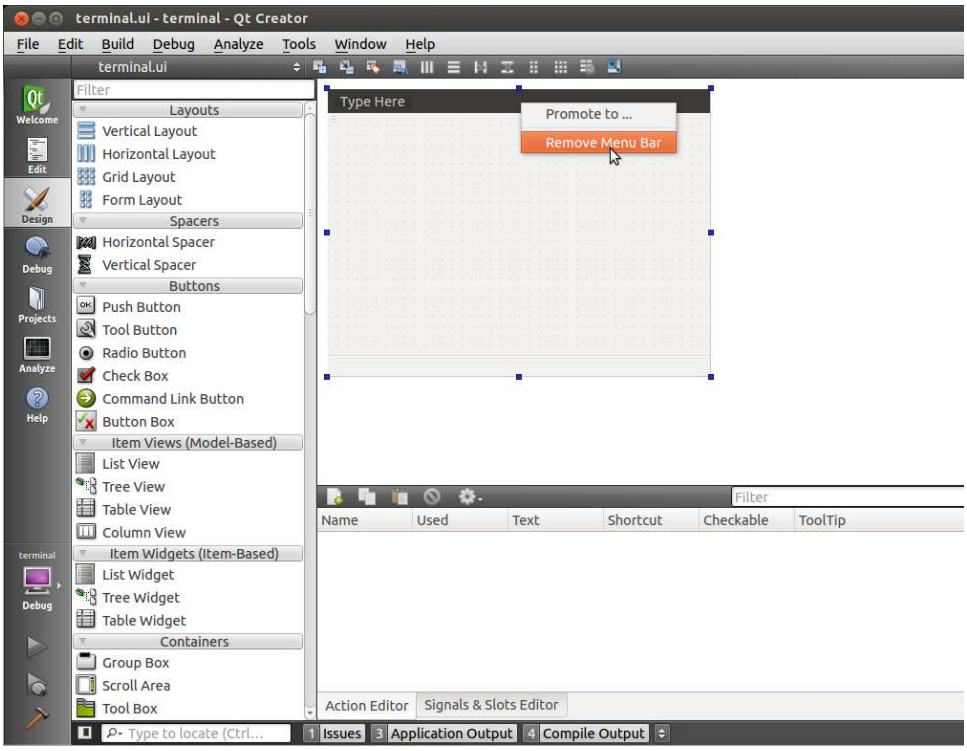

Figure 26. Removing the Menu Bar

6. Repeat this same procedure to remove the statusBar (located at the very bottom of the screen) and the toolBar (located at the top of the screen). Note that both of these elements are hard to see.

3.1.1 Adding Stacked Widget to the GUI

1. On the left side of the widgets editor, find the column labeled Containers (see Figure 27).

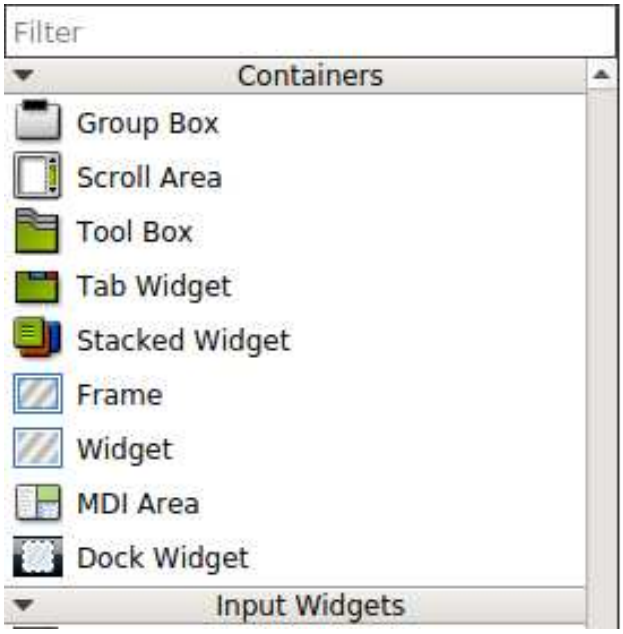

Figure 27. Widgets Editor Containers Section

2. Drag and drop the Stacked Widget onto the GUI (see Figure 28).

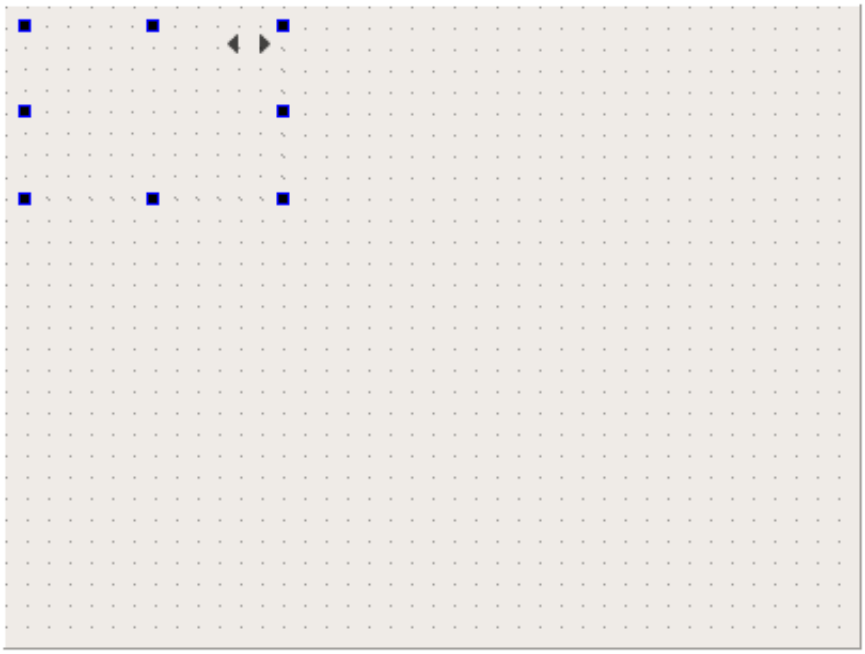

Figure 28. demoGUI With Stacked Widget

3. Right-click on the Stacked Widget and select change styleSheet (see Figure 29).

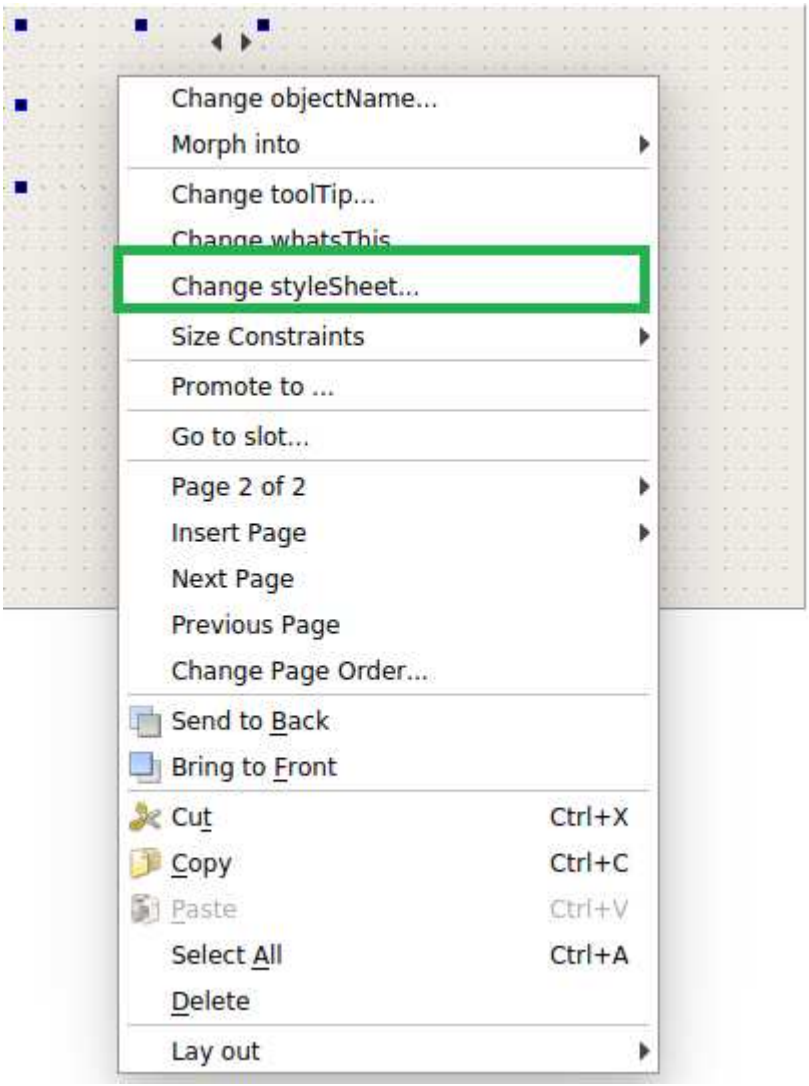

Figure 29. Change styleSheet Option

4. Edit the styleSheet (see Figure 30) to contain the following statement:
   background-color:rgb(170,85,255);

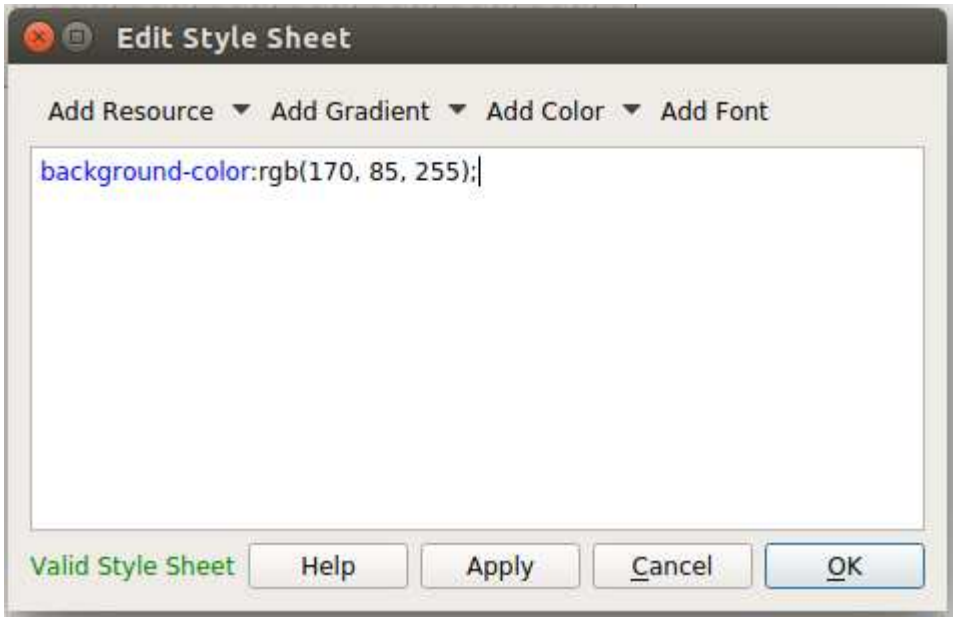

Figure 30. Stacked Widget styleSheet

At this point, your GUI should look like Figure 31.

Figure 31. demoGUI With Customized Stacked Widget

Now make the Stacked Widget occupy the entire GUI. Drag the corners of the Stacked Widget to occupy the entire area of the GUI (see Figure 32).

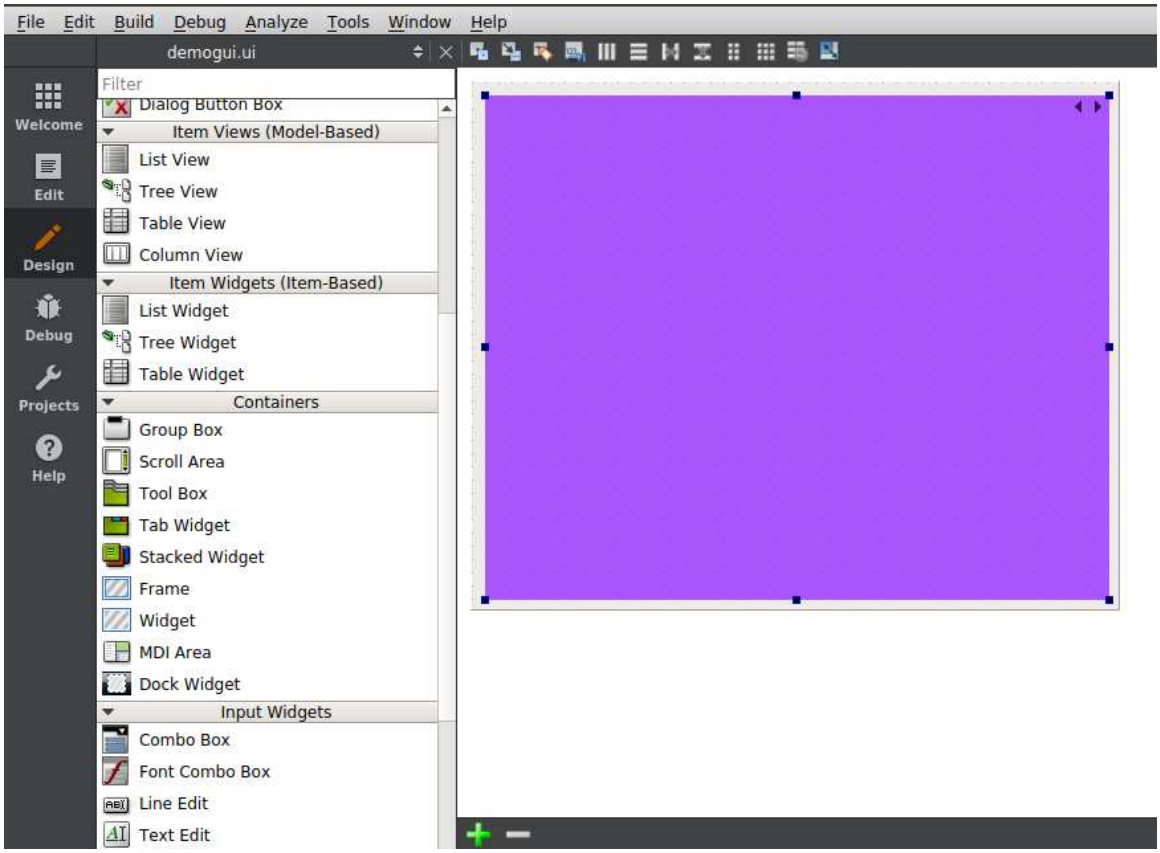

Figure 32. demoGUI With Stacked Widget

3.1.2 Making the GUI Fill Entire EVM Screen

1. Click the Edit tab and navigate to main.cpp under the Sources folder.
2. Change w.show() to w.showFullScreen() (see Figure 33).

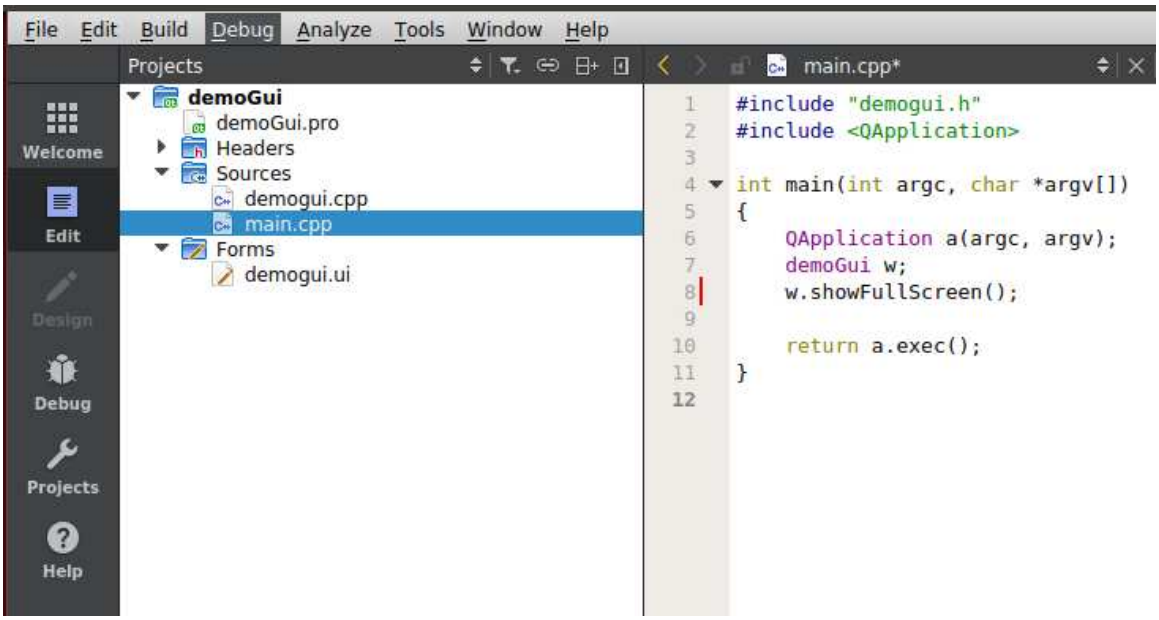

Figure 33. main.cpp Edits Show demoGUI in Full Screen Mode (1/2)

3. Click File → Save All.
4. Click the Build button located on the bottom left side of the screen (see Figure 34).
5. Next, click the green deploy button located above to see the GUI on your EVM.

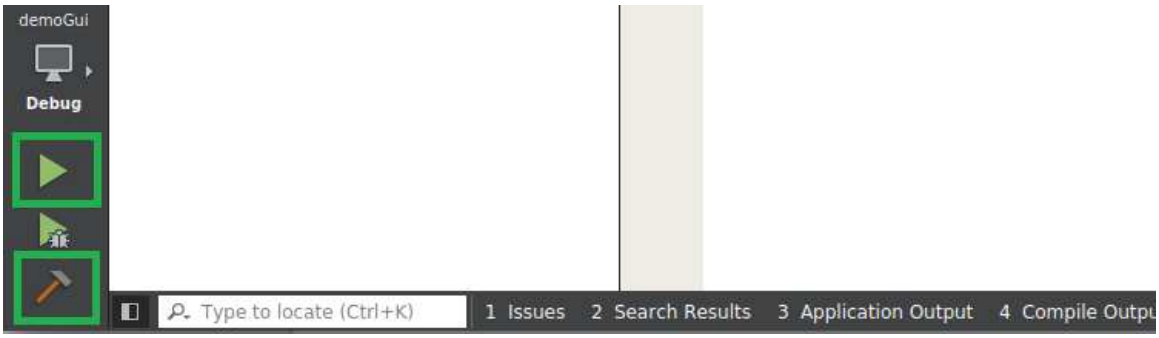

Figure 34. main.cpp Edits Show demoGUI in Full Screen Mode (2/2)

Figure 35 shows what the GUI should look like. The GUI is now occupying the entire screen as intended. However, the Stacked Widget is not full screen. It appears the size of the Stacked Widget is static because it is not expanding to fit the size of the GUI.

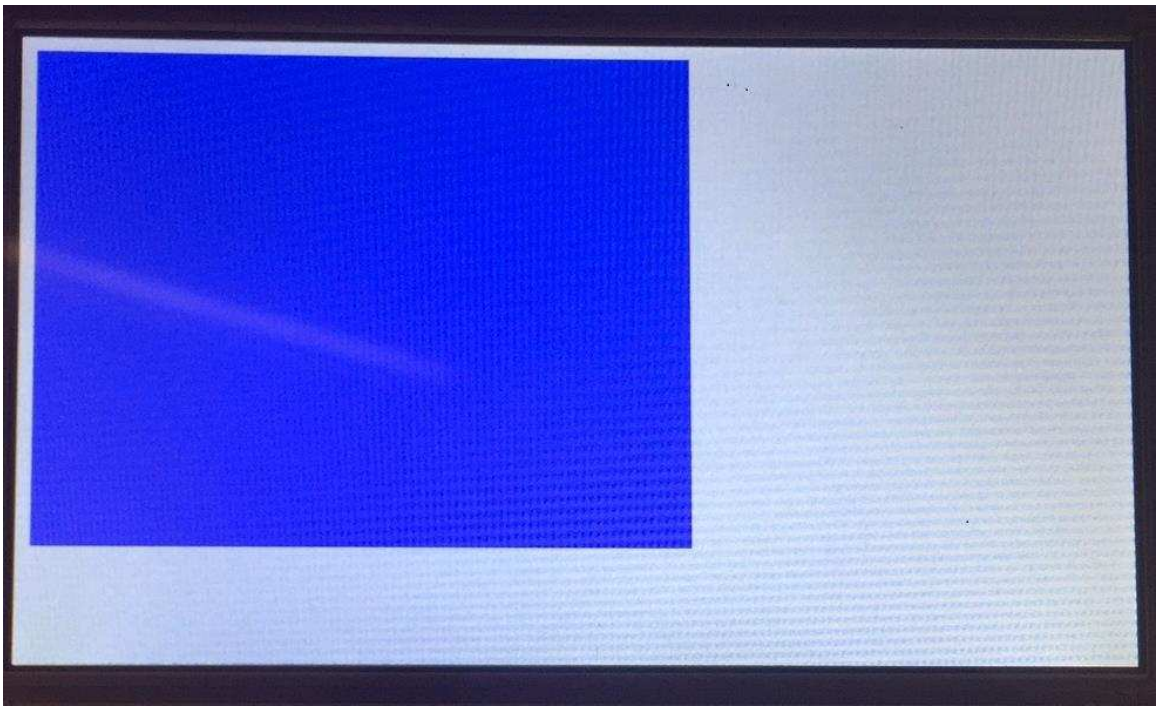

Figure 35. EVM With Full Screen GUI Deployed

3.1.3 Inserting Layout to Make Stacked Widget Resize

Inserting a layout scheme for the overall GUI allows elements on the GUI to grow as the size of the screen it is displayed on increases.

1. In the Object Inspector, under the Class Column right click on QmainWindow and select Layout (see Figure 36).

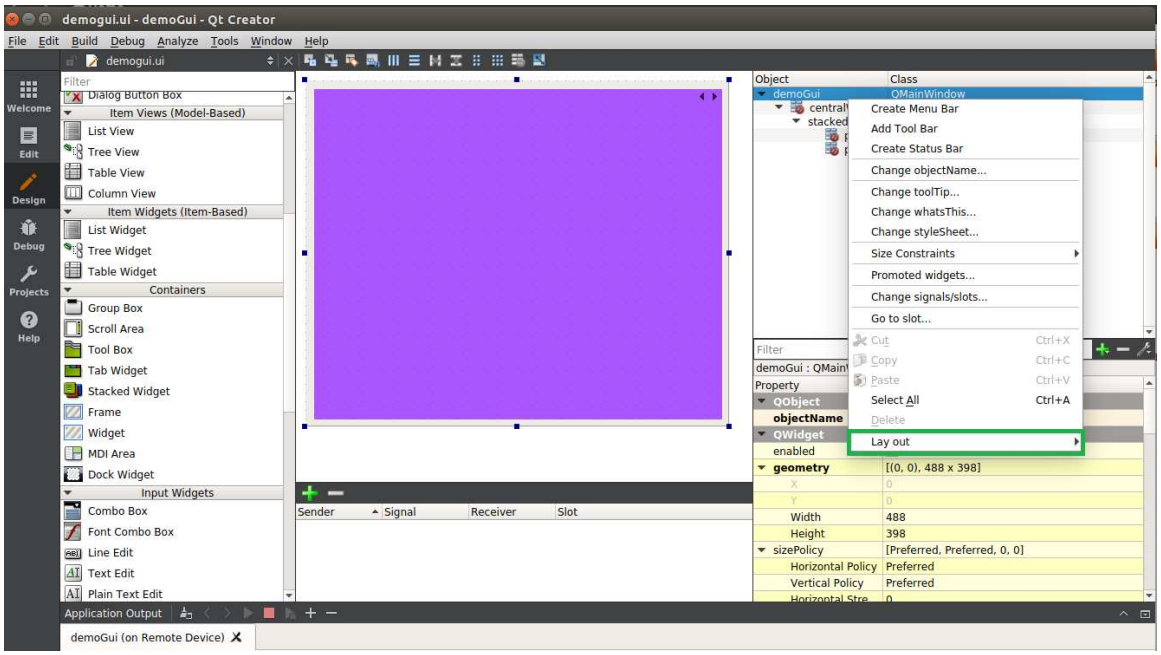

Figure 36. Applying a Layout to demoGUI

2. Select Layout Horizontally, Layout Vertically, or Layout in a Grid. Each has the same effect as of now.
3. Save, build, and deploy the GUI. The Stacked Widget should now fill the entire GUI (see Figure 37).

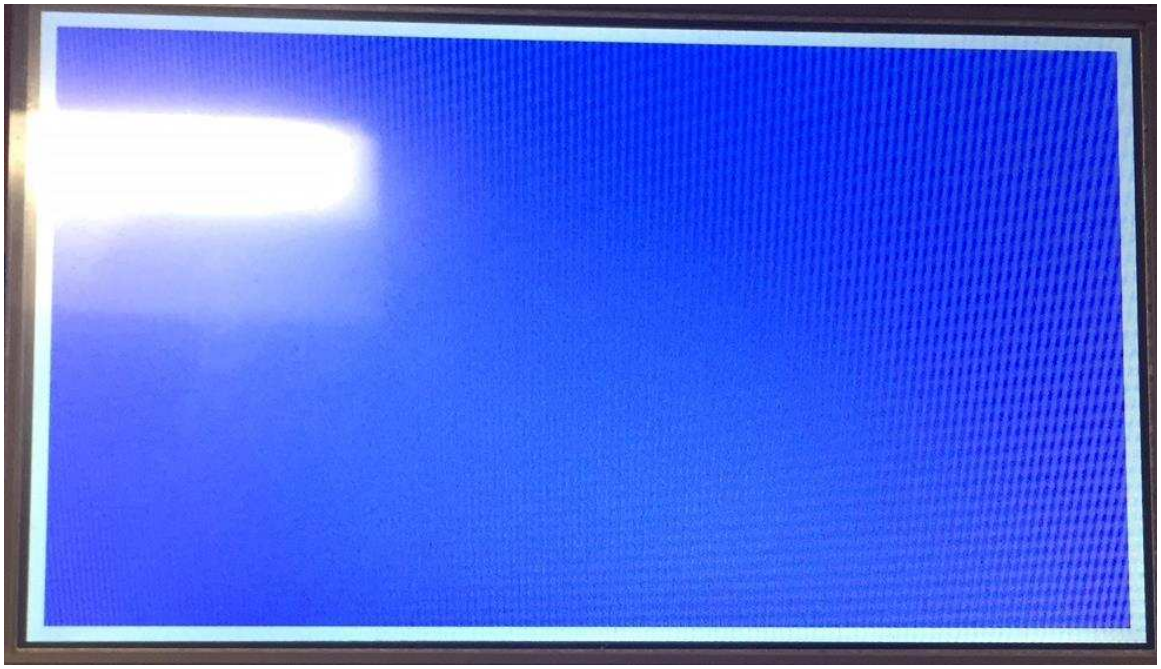

Figure 37. EVM With Full Screen Stacked Widget

NOTE: If you have a full screen GUI already running on your EVM, you are not able to deploy your updated GUI. Click on the Application Output Tab and press the stop button (see Figure 38).

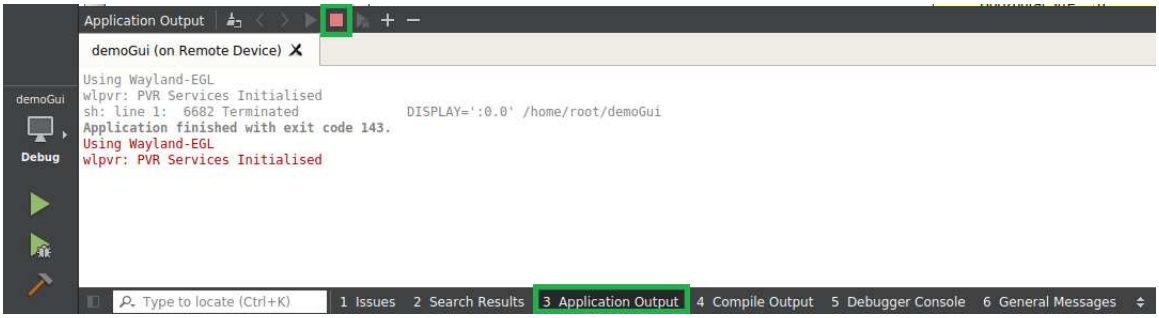

Figure 38. Application Output Tab

3.1.4 Adding Text Label to First Page of GUI

1. In the Widget Box, scroll down to the section labeled Display Widgets. Drag and drop the Label widget onto your GUI (see Figure 39).

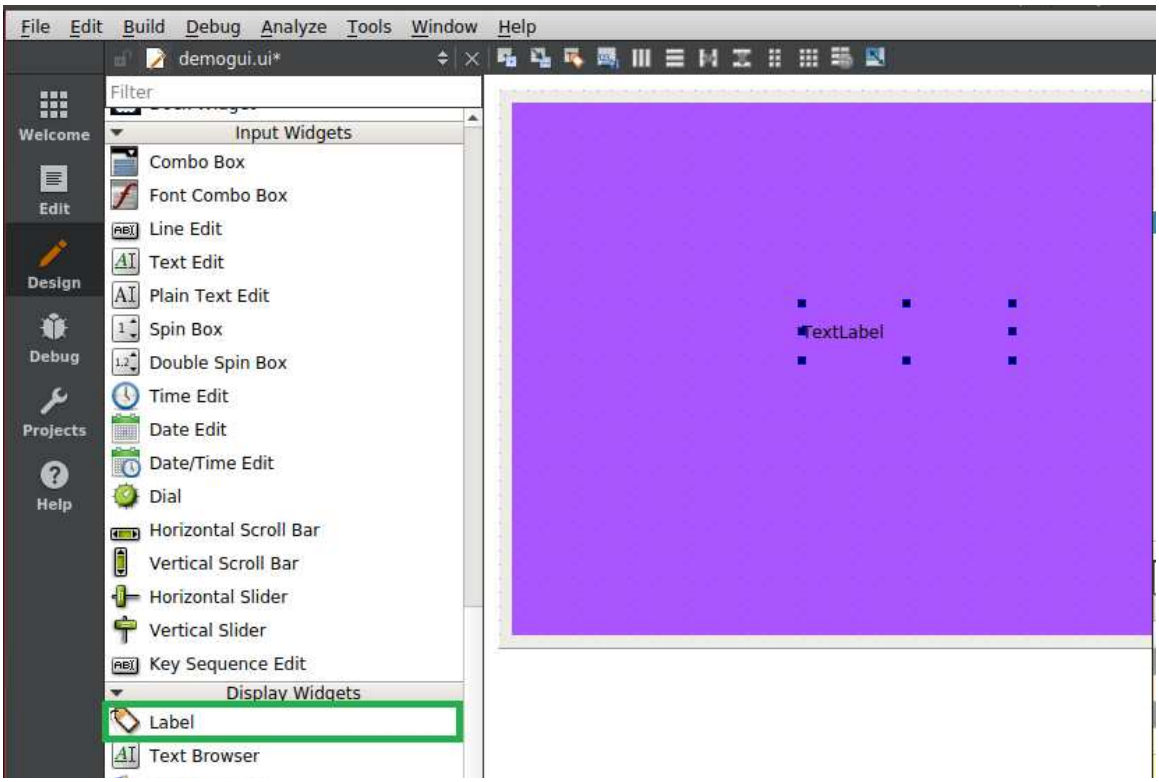

Figure 39. Adding Text Label to GUI

2. Right click on the text label and select the Change rich text… option (see Figure 40).

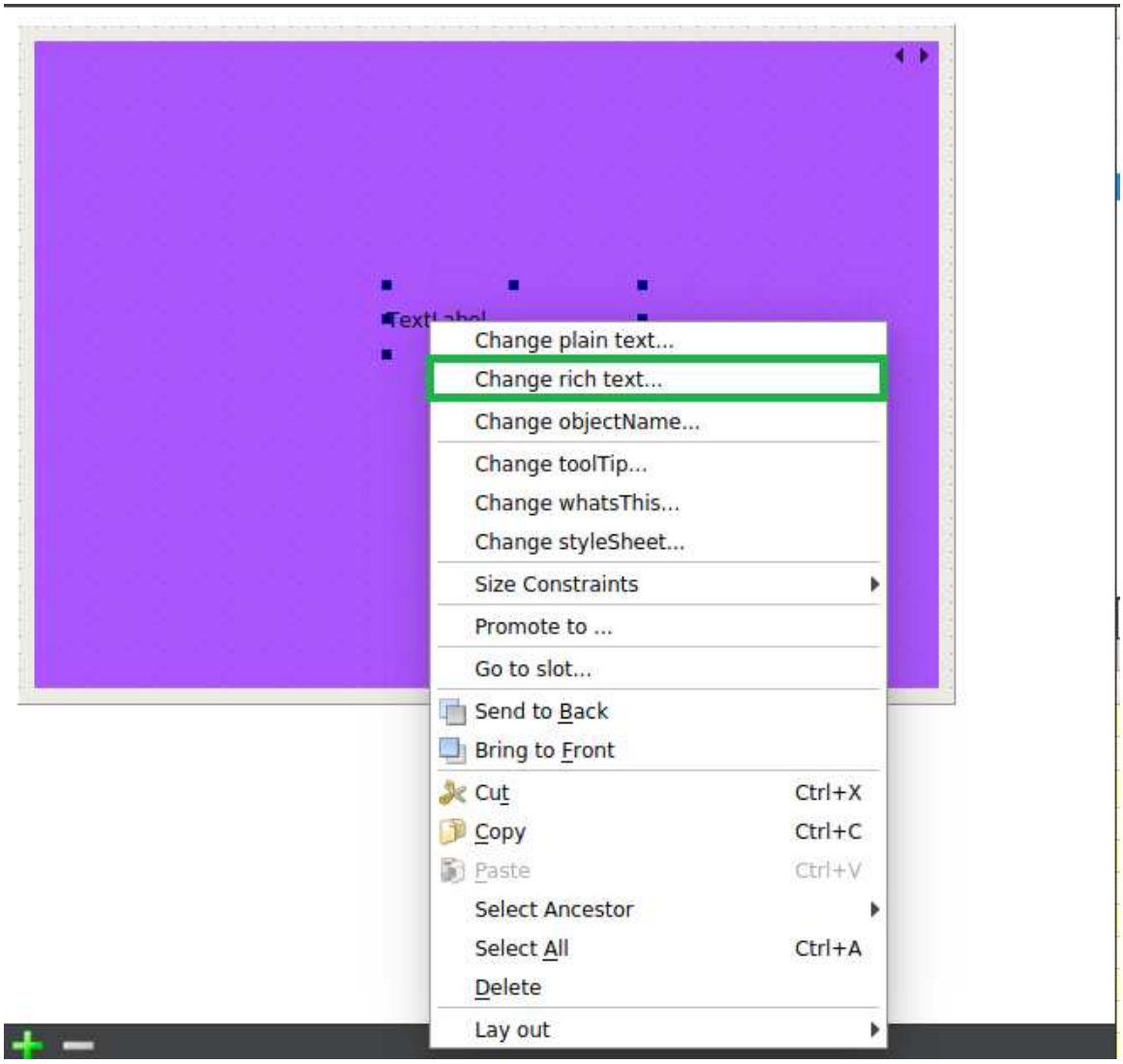

Figure 40. Change Rich Text Option

3. Edit the text to say Welcome! (see Figure 41).
4. Change the text size to 28 (see Figure 41).
5. Press Ok and the modified text appears on your GUI (see Figure 41).

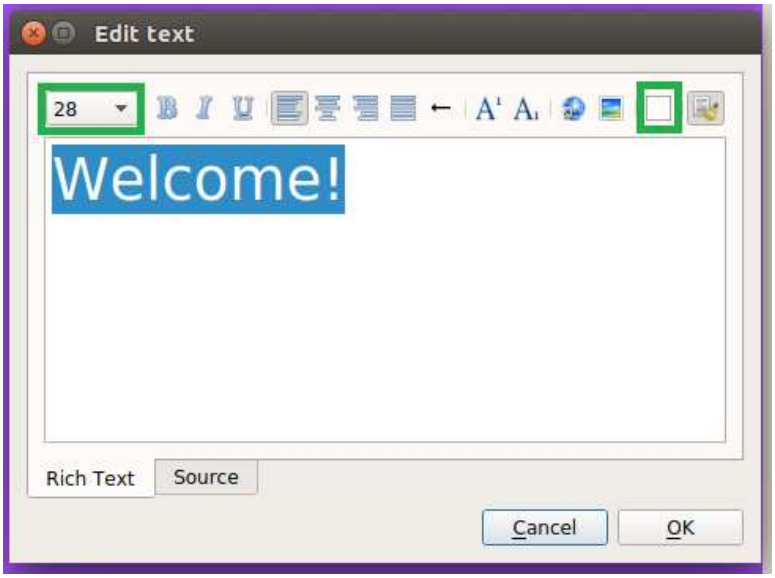

Figure 41. Editing Rich Text on Text Label

3.1.5 Adding Menu Button and Exit Button

Follow these steps to add buttons to navigate the pages and exit the GUI.

1. In the Widget Box, scroll down to the section labeled Buttons, then drag and drop two Push Buttons onto the GUI.
2. Double click on each of the buttons and edit the text to read *Menu* on one button and *Exit* on the other.
3. Edit the positions of the Welcome label and the two buttons to match Figure 42.

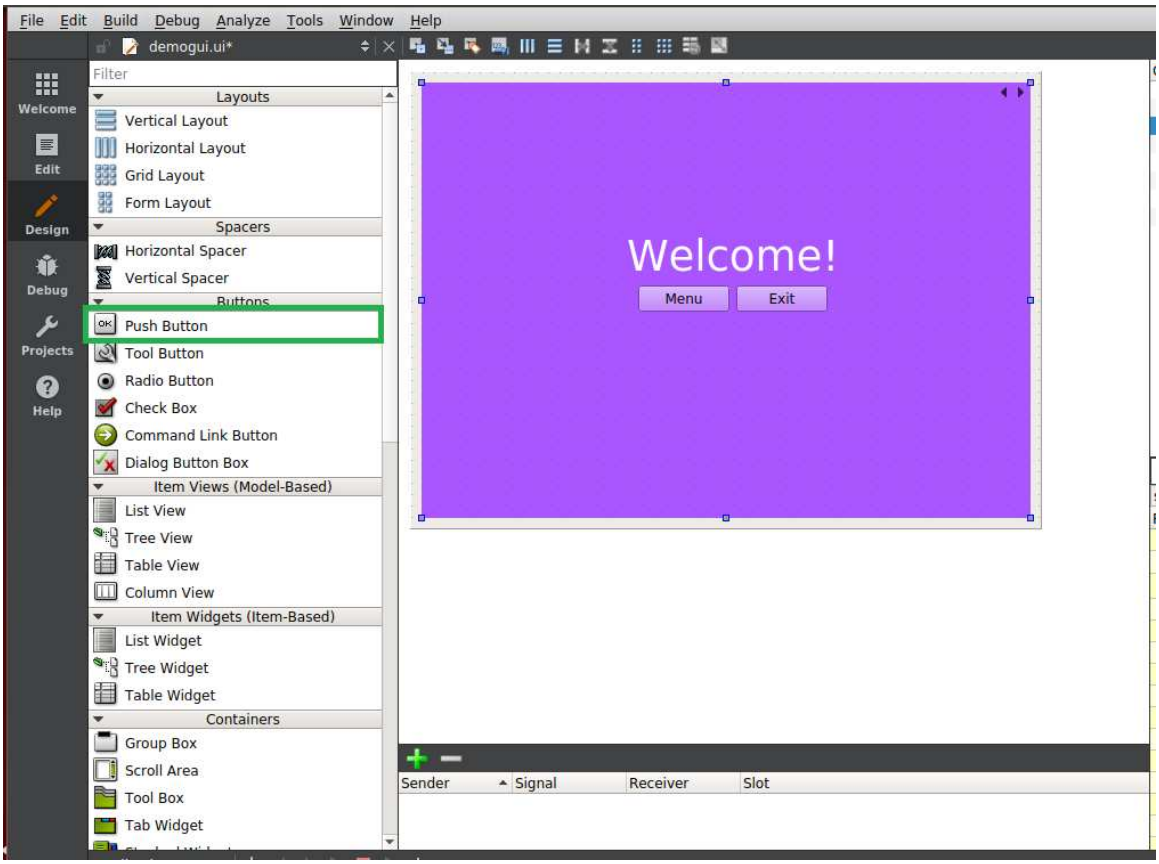

Figure 42. Centering Menu Button, Exit Button, and Welcome Label

4. Right click on the Menu button, then click the Go to slot… option.
5. Select the clicked() option, and press Ok (see Figure 43).

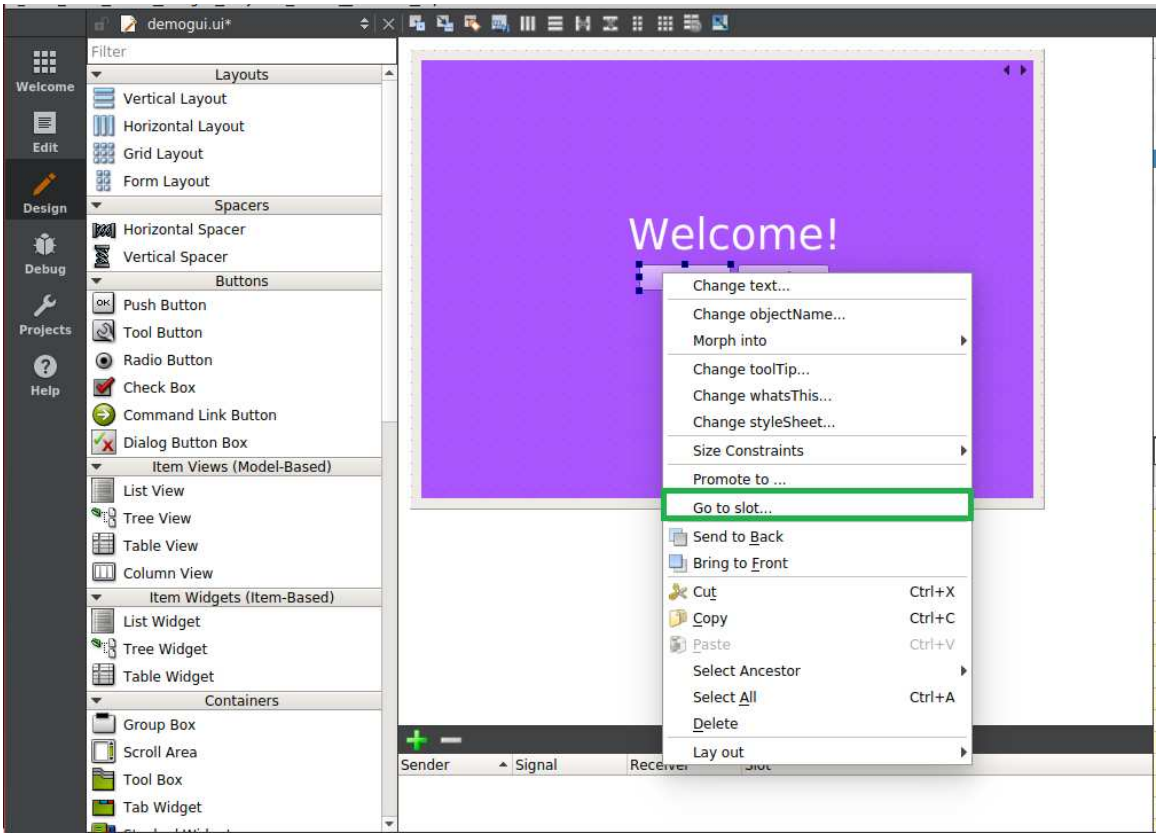

Figure 43. Go to Slot… Option

QT should automatically take you to the demoGUI.cpp file. The demoGUI.cpp file should have a new function added to the bottom of it called on_pushButton_clicked().

6. Add the following line of code to that function:

```
ui -> stackedWidget -> setCurrentIndex(1);
```

Indexing of Stacked Widget pages starts from 0, so index 1 corresponds to the second page of the GUI.

7. Navigate back to the ui and follow the same process to create the function on_pushButton_2_clicked()
8. Add the following line of code to that function:

```
qApp -> quit();
```

Your demoGUI.cpp file should look like Figure 44.

```
#include "demogui.h"
#include "ui_demogui.h"

demoGui::demoGui(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::demoGui)
{
    ui->setupUi(this);
}

demoGui::~demoGui()
{
    delete ui;
}

void demoGui::on_pushButton_clicked()
{
    ui -> stackedWidget -> setCurrentIndex(1);
}

void demoGui::on_pushButton_2_clicked()
{
    qApp -> quit();
}
```

Figure 44. Edited demoGUI.cpp File

Notice that demoGUI.h now has the on_pushButton_clicked() and on_pushButton_2_clicked() functions added to it (see Figure 45).

```
1  #ifndef DEMOGUI_H
2  #define DEMOGUI_H
3
4  #include <QMainWindow>
5
6  namespace Ui {
7  class demoGui;
8  }
9
10 class demoGui : public QMainWindow
11 {
12     Q_OBJECT
13
14 public:
15     explicit demoGui(QWidget *parent = 0);
16     ~demoGui();
17
18 private slots:
19     void on_pushButton_clicked();
20
21     void on_pushButton_2_clicked();
22
23 private:
24     Ui::demoGui *ui;
25 };
26
27 #endif // DEMOGUI_H
```

Figure 45. Modified demoGUI.h File

9. Save all your changes, then build and deploy your application (see Figure 46).

You will notice the same problem addressed earlier, the Welcome label, and two buttons did not adjust and recenter when the screen size increased.

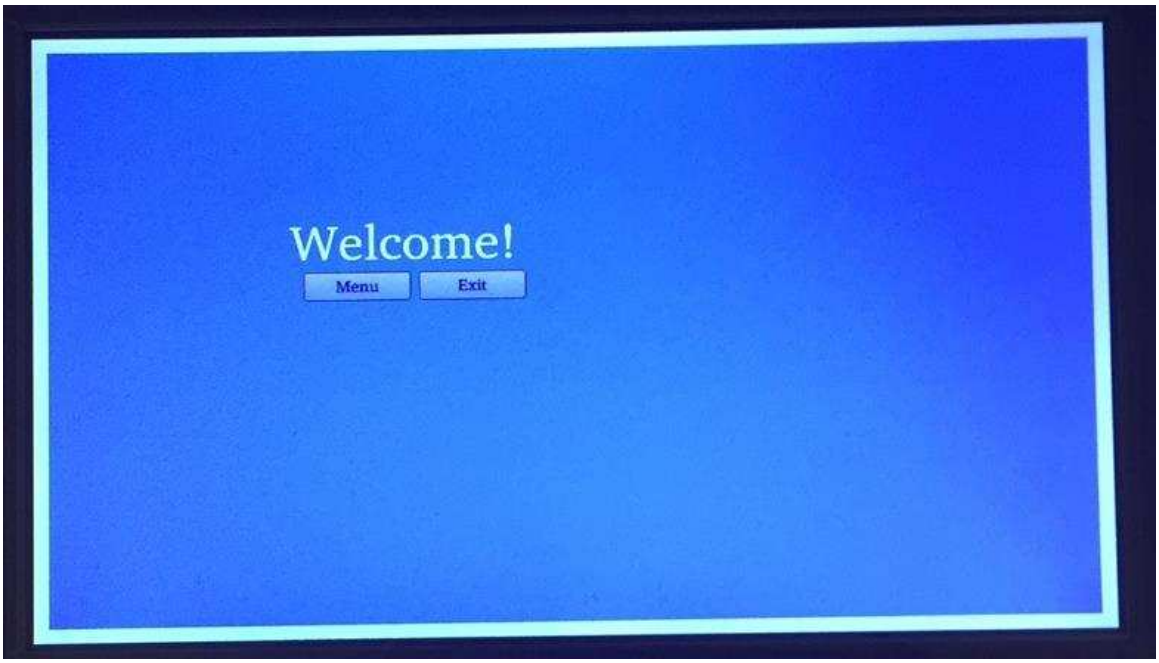

Figure 46. demoGUI Deployed Without Layout Applied to Stacked Widget

3.1.6 Experimenting With Layers For Stacked Widget

Right click on stackedWidget in the Object Inspector, then select Layout and experiment with the different layout options. Notice how they affect the screen. Figure 47 shows how QStackedWidget looks with the vertical layout applied to it (see Figure 47).

Figure 47. Stacked Widget With Vertical Layout Applied

After experimenting with the different layout options, you will realize that none of the available options allows us to position our buttons and label correctly.

3.2 Using Spacers to Position Stacked Widget Elements When Layout Is Applied

1. In the Spacers section of the Widget box find the vertical spacer.
2. Drag and drop two vertical spacers onto your screen (see Figure 48).

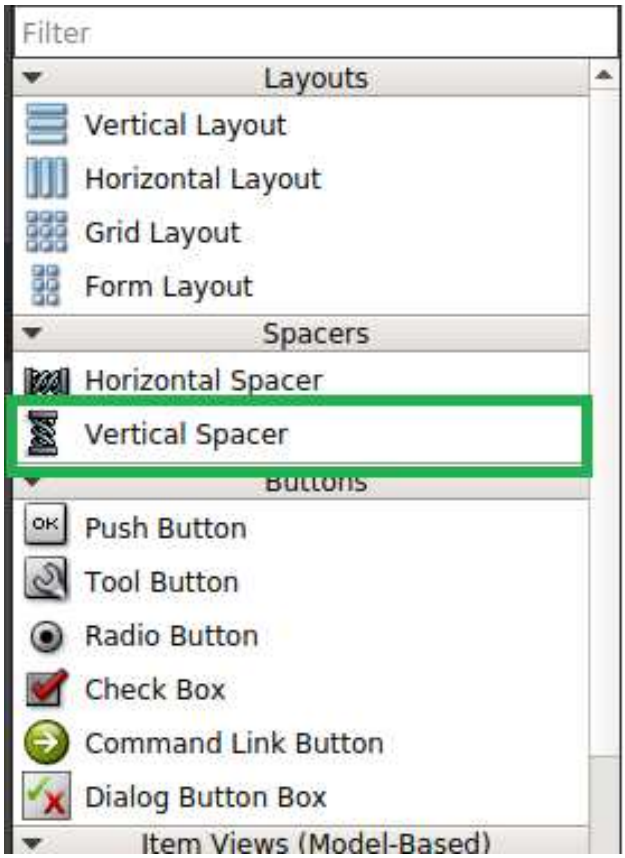

Figure 48. Vertical Spacer Entry in Widget Box

3. Expand and position the spacers as shown in Figure 49.

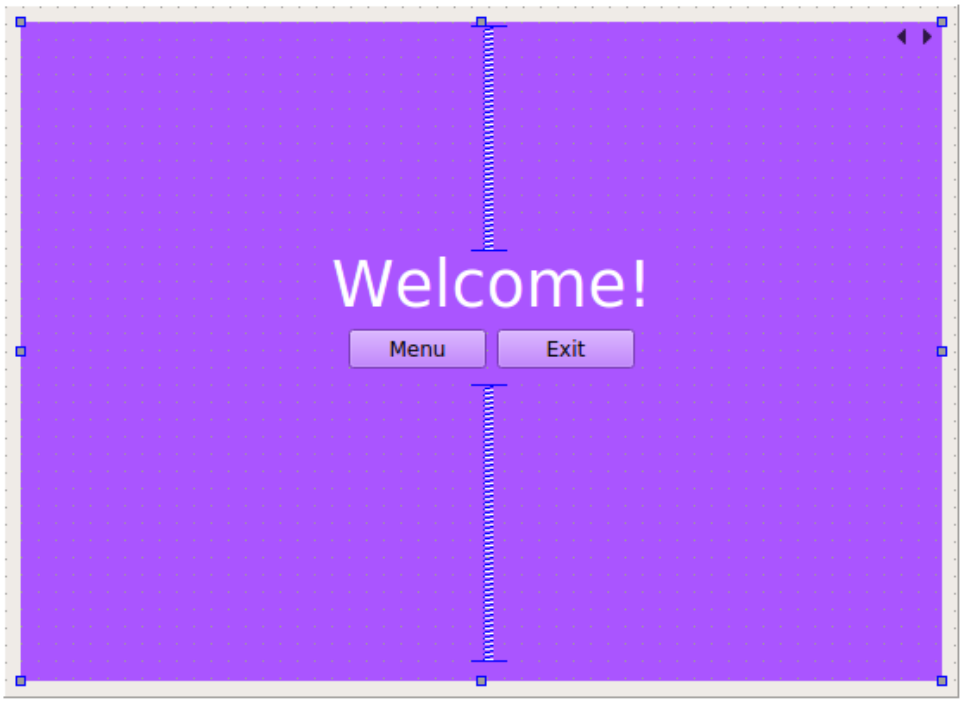

Figure 49. Vertical Spacers Placed on Stacked Widget

4. Now apply the vertical layout to stackedWidget (see Figure 50).

Figure 50. Vertical Layout Applied to Stacked Widget

The screen looks better than it did in Figure 47. However, it still needs some work.

To remove the vertical layout go to the Object Inspector, right click on stackedWidget, then select Layout and choose the Break Layout option. The elements on your screen will be moved around. You can place them back in their original positions.

3.2.1 Grouping Elements

An issue with the vertical layout is that it consistently places the two push buttons stacked on top of each other instead of side-by-side. To solve that issue we can apply a horizontal layout to just the two buttons.

1. Select the menu button, hold down the ctrl button, and select the Exit button. At this point you should have both buttons selected.
2. Release the ctrl button and right click one of the selected buttons.
3. Go to Layout and select Layout Horizontally. A red box appears around the two buttons indicating they have been grouped together (see Figure 51).

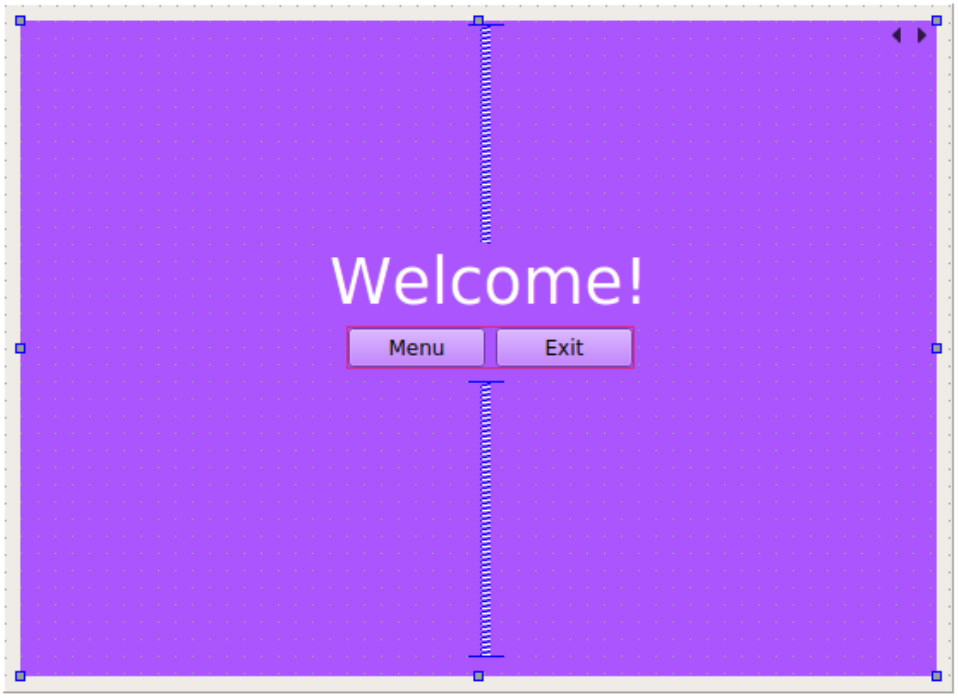

Figure 51. Horizontal Layout Applied to Menu and Exit Buttons

4. Now apply the vertical layout to the stackedWidget.

The GUI is getting closer to what we want it to look like. However, the buttons are stretching to fill the entire width of the screen. Group the buttons with two horizontal spacers to fix the issue with buttons stretching to fill the entire width of the screen.

1. Break the layout of stackedWidget.
2. Drag and drop two horizontal spacers onto the screen, and adjust your GUI to look like Figure 52.
3. Select the two spacers and the box grouping the two buttons together.

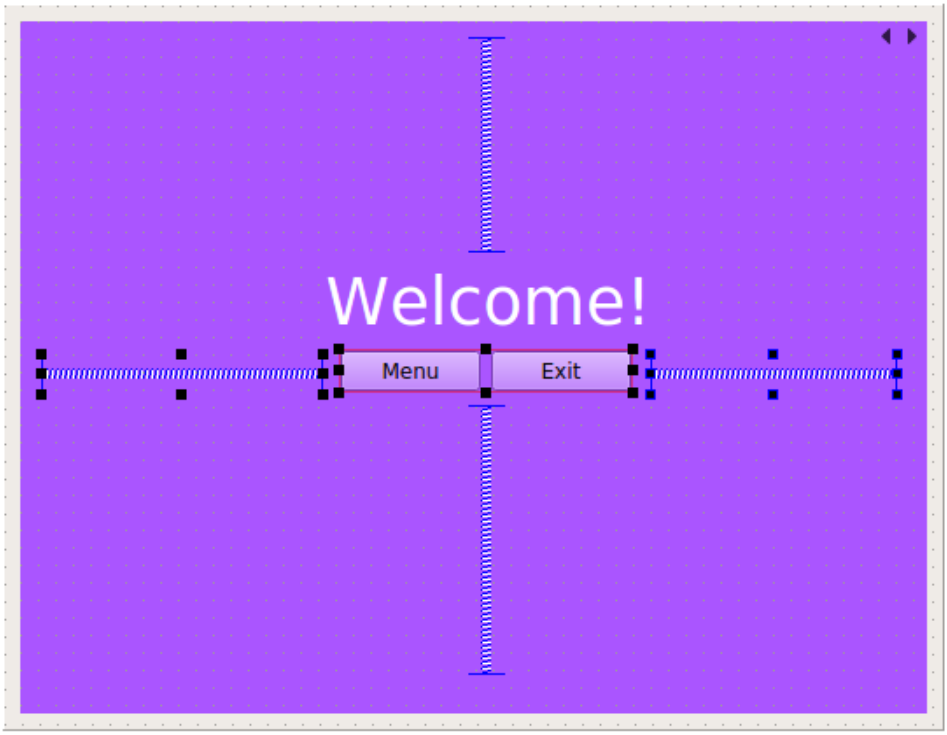

Figure 52. Selecting Horizontal Spacers and Horizontal Layout Box

4. Right click and apply a horizontal layout to the three elements currently selected.

5. Now apply a vertical layout to stackedWidget. Your screen should look like Figure 53.

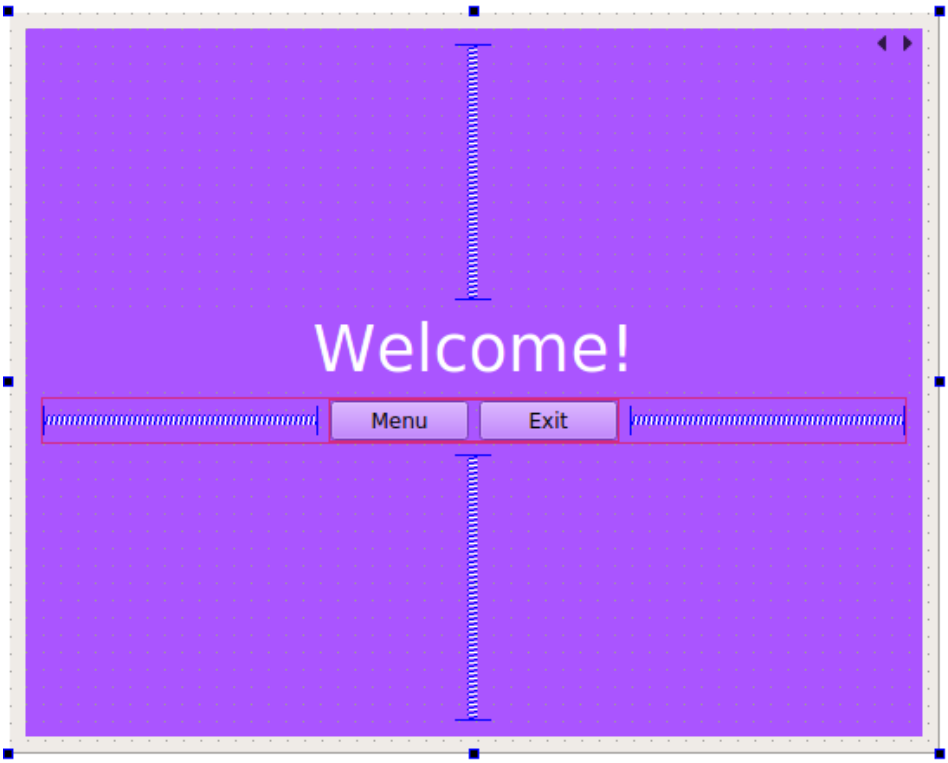

Figure 53. Vertical Layout and Horizontal Spacers Added to demoGUI

6. Deploy the application to see how it looks on your EVM.

   The EVM should look like Figure 54. The EVM looks pretty good, however, the buttons look really small in relation to the rest of the screen.

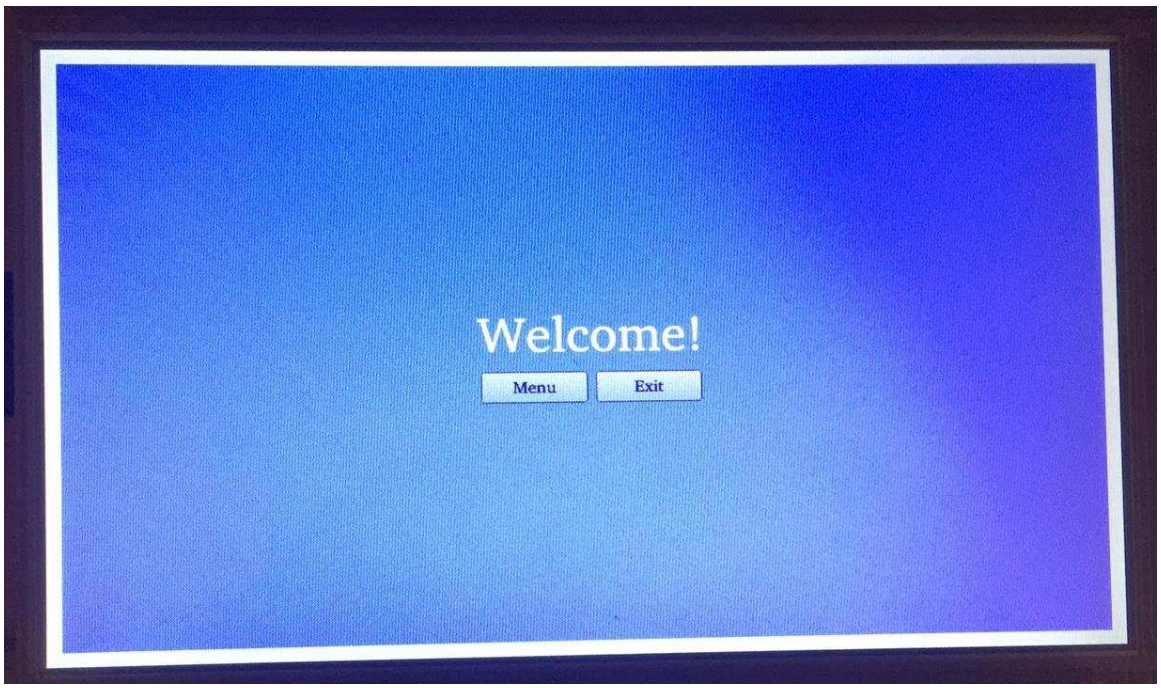

Figure 54. demoGUI Deployed With Spacers

3.3 Property Editor

3.3.1 Expanding Buttons

1. Click on the Menu Button and go to the Property Editor (see Figure 55).
2. Under the QWidget section there is a section called sizePolicy. Change the Horizontal and Vertical Policies to Expanding (see Figure 55).

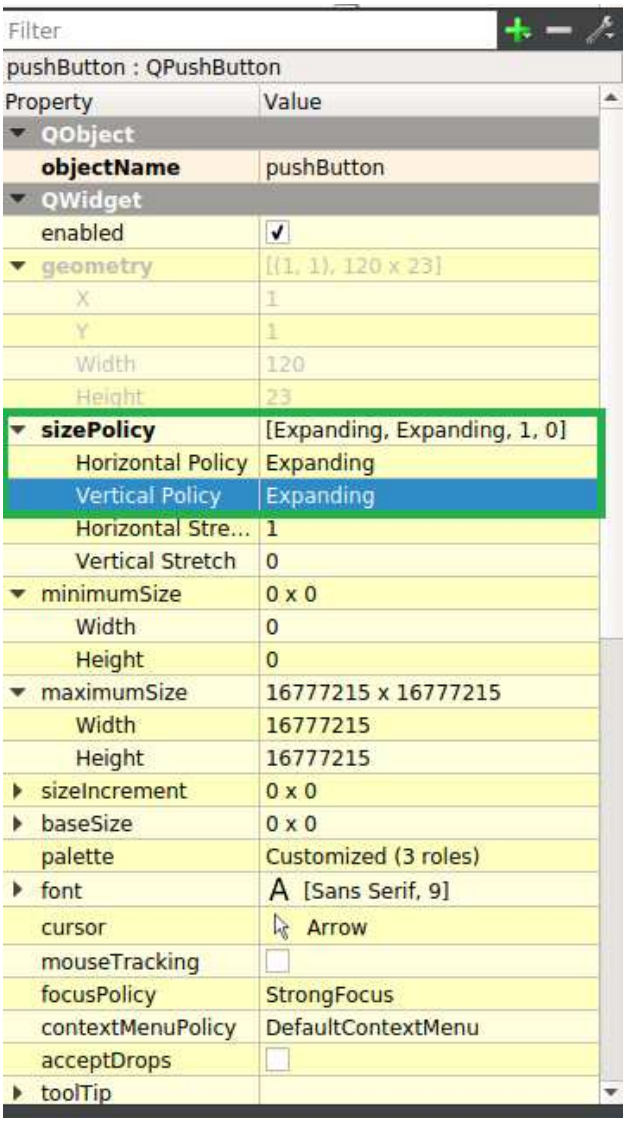

Figure 55. QPushButton (Menu Button) Property Editor

3. Repeat steps 1 and 2 for the Exit button as well.
4. Now deploy your GUI, it should look like Figure 56. Notice, the buttons have expanded too much in the vertical direction.

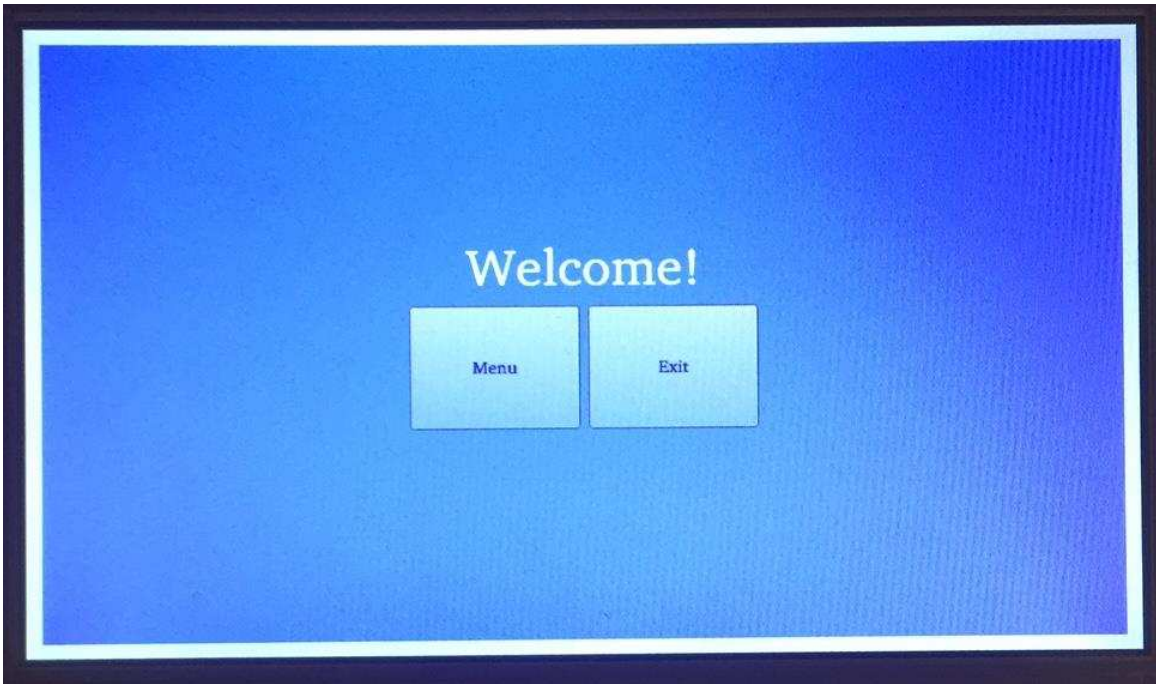

Figure 56. demoGUI Deployed With Expanding Property Applied to Push Buttons

5. Break the layout of stackedWidget.
6. Choose the box holding the two buttons and two horizontal spacers and break its layout.
7. Drag and drop two vertical spacers onto the screen.
8. Adjust the height of the spacers to be 10 using the Property editor.
9. Adjust the spacers and box containing the two buttons to look like Figure 57.

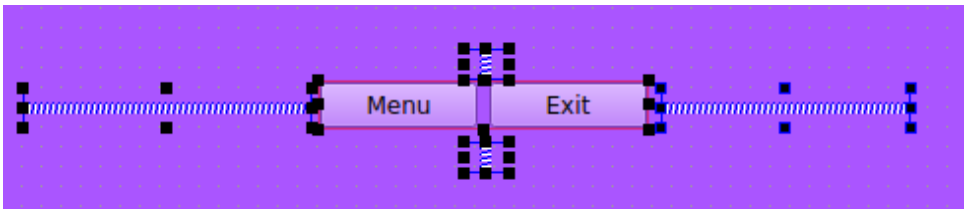

Figure 57. Adding Vertical Spacers Above and Below Push Buttons

10. Select all five elements (four Spacers and Horizontal Layout containing the two pushbuttons), as shown in Figure 57. Right click and apply the grid layout.

11. Apply a vertical layout to stackedWidget, it should look like Figure 58.

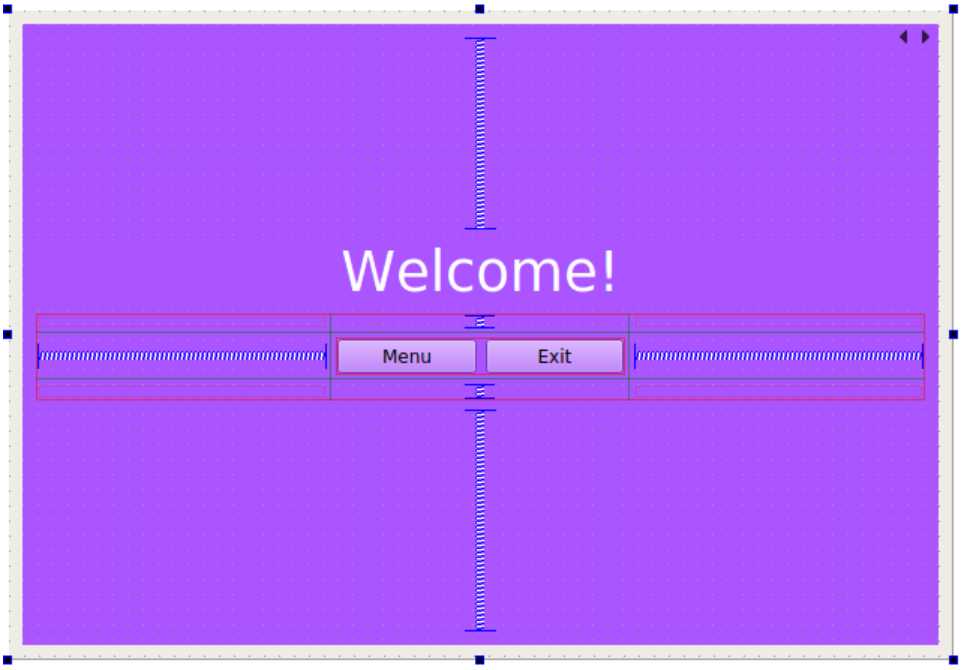

Figure 58. Adding Vertical Spacers Above and Below Push Buttons

12. Deploy your application to view it on the EVM. You will see that the Welcome Label and two push buttons are occupying the screen in a better way now (see Figure 59).

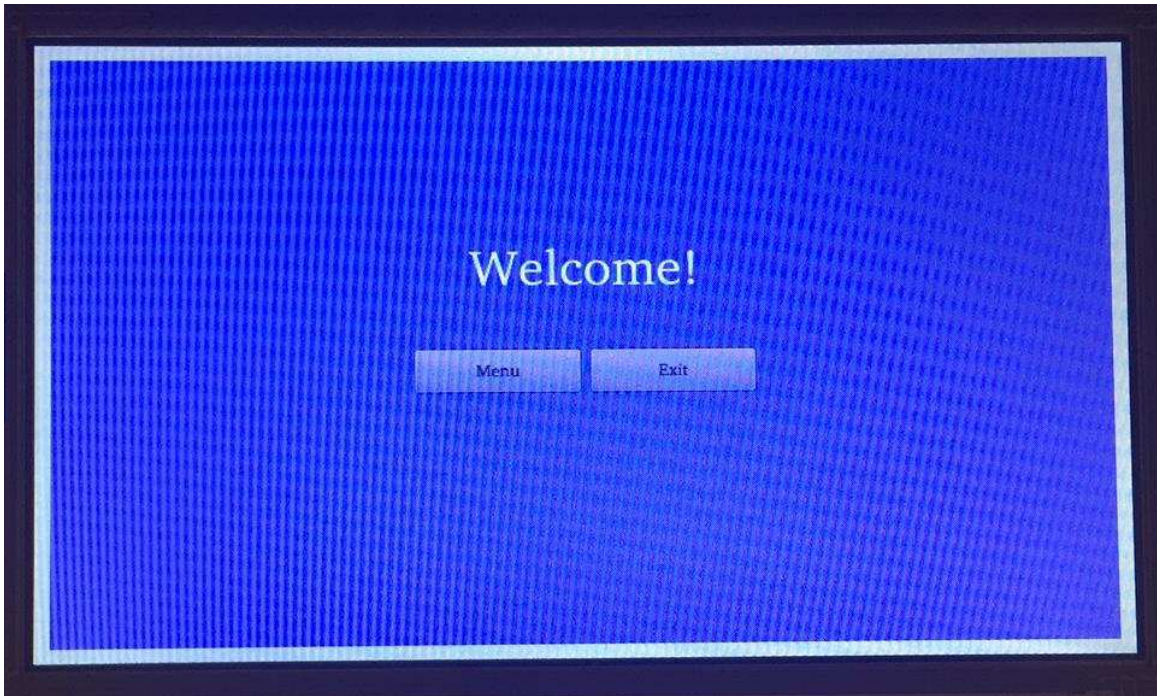

Figure 59. demoGUI Deployed With Vertical Spacers Above and Below Push Buttons

4 Adding Image to Qt GUI

This section covers how to add an image to your Qt GUI application.

1. Open the demoGUI project created in **Section 2.3.** If you did not create the demoGUI project, you can create a new project and follow the steps of this tutorial.
2. Right click on the demoGUI project name and click Add New (see Figure 60).

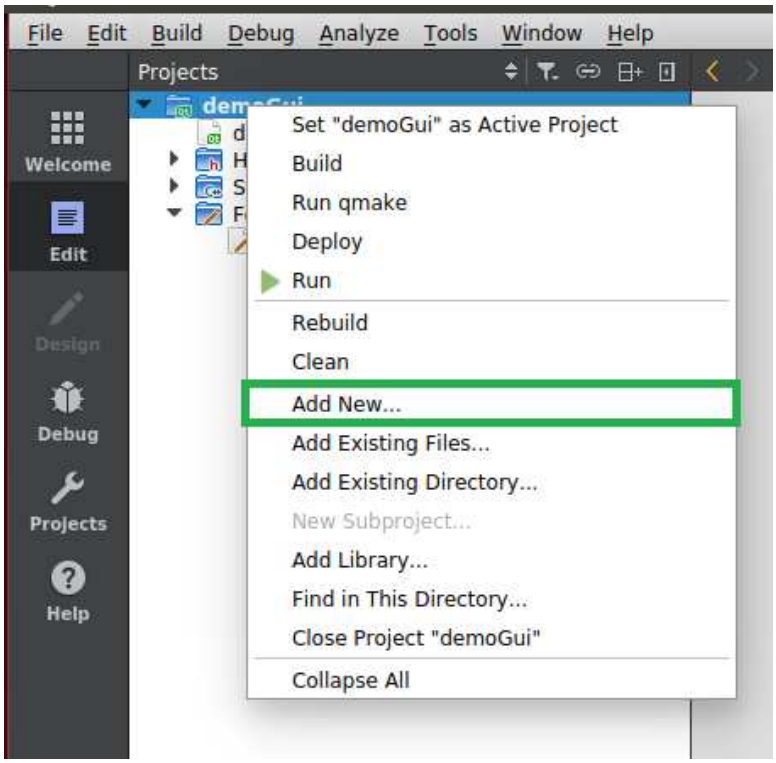

Figure 60. Add New… Option

3. Select Qt, then Qt Resource File, and Choose (see Figure 61).

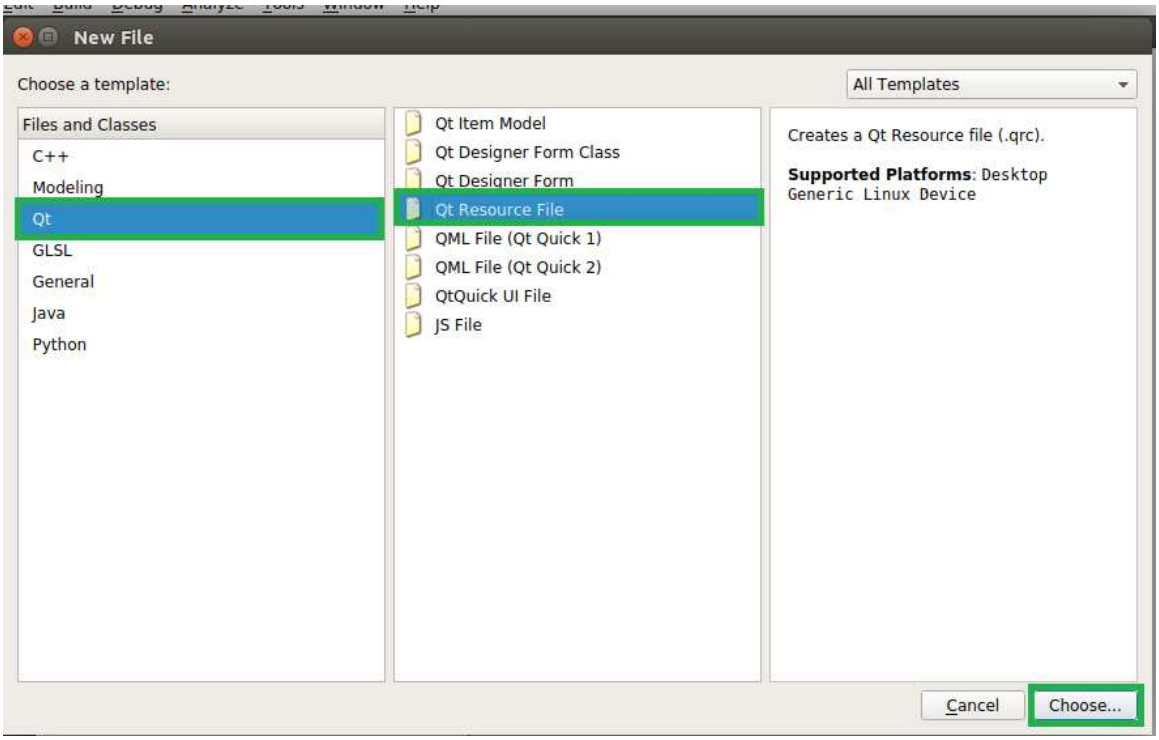

Figure 61. Creating Qt Resource File for demoGUI Project

4. Type myMedia for the name field and click Next (see Figure 62).
5. Click the Finish button.

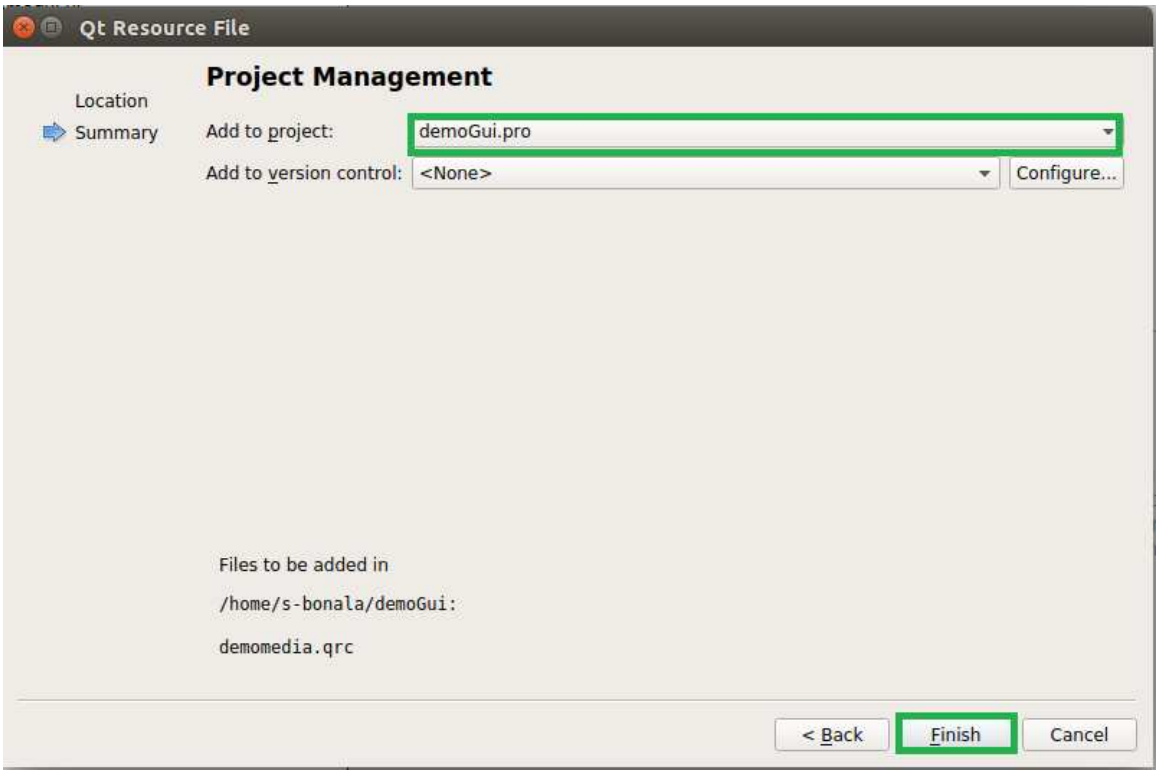

Figure 62. Adding Qt Resource File to demoGUI Project

A Resources folder is added to your project (see Figure 63).

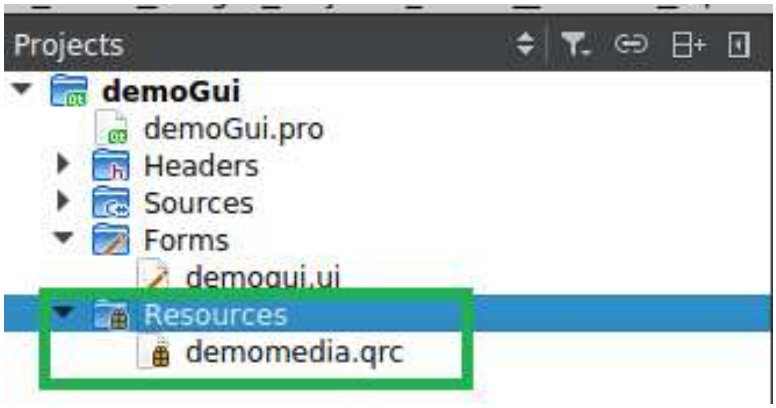

Figure 63. Resource Folder Added to demoGUI Project

Next, add an image to the demoGUI project directory from your computer. You will insert this image onto the second page of the demoGUI project.

6. Click on demomedia.qrc to open the resource file.
7. Click Add → Add Prefix (see Figure 64).

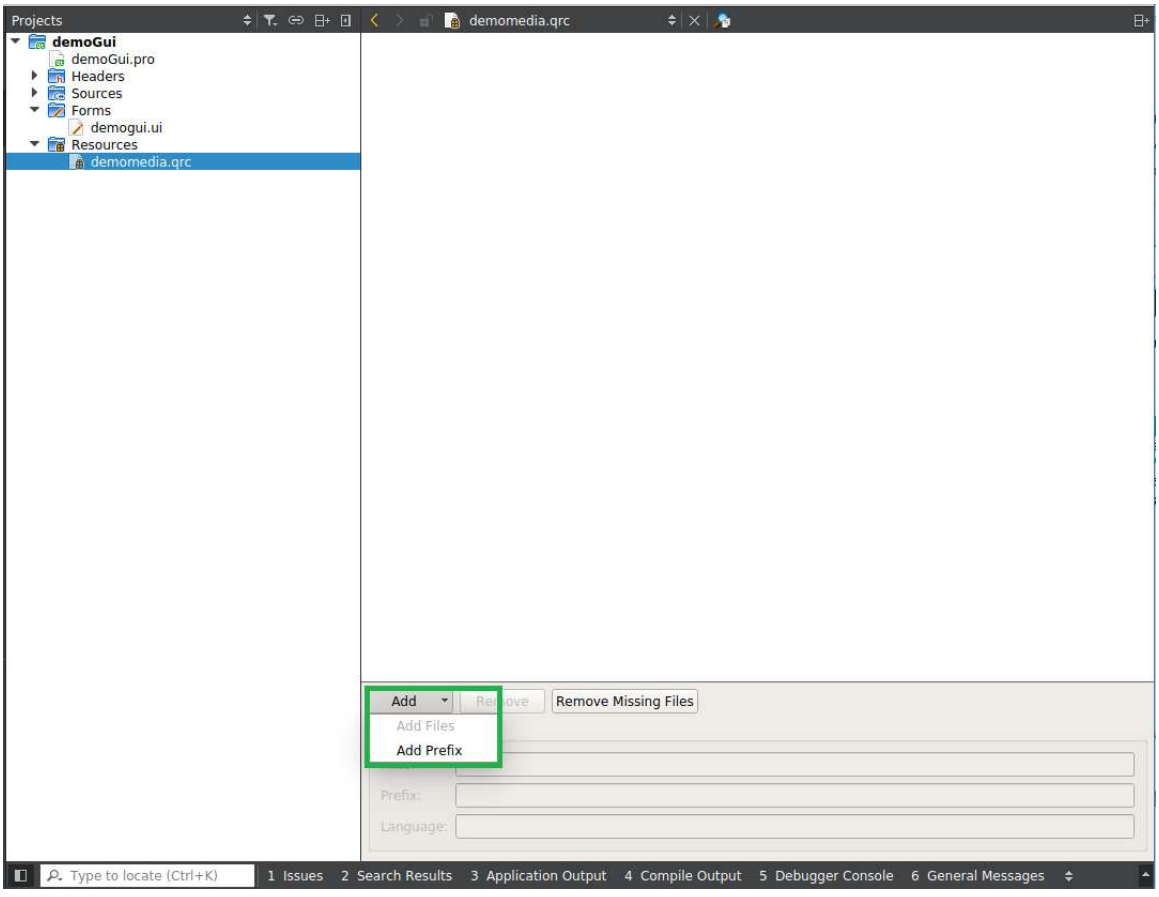

Figure 64. Adding a Prefix

8. Click Add → Add File (see Figure 65).

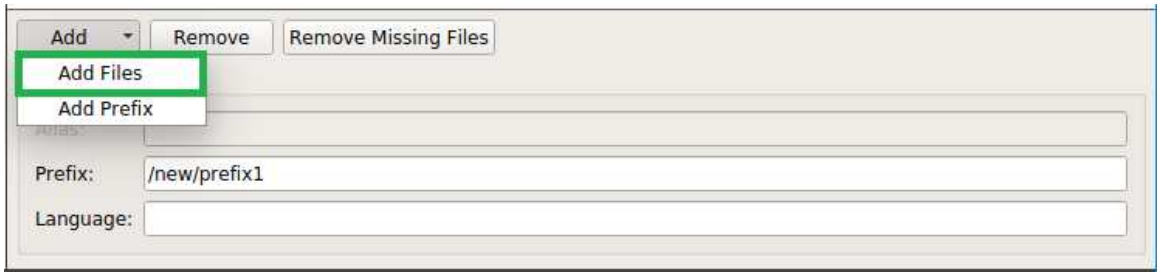

Figure 65. Adding a File

9. Choose your image and click Open (see Figure 66).

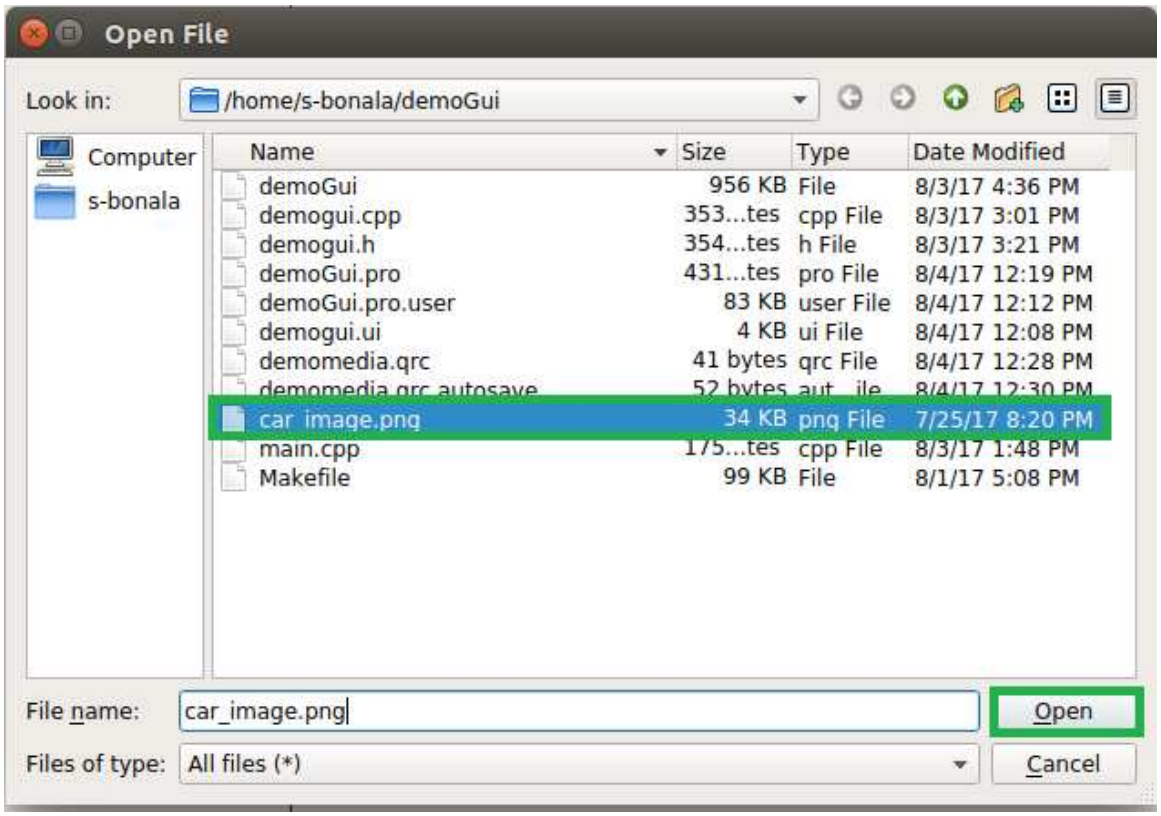

Figure 66. Choosing an Image

10. Change the Prefix field from /new/prefix1 to /pictures (see Figure 67).

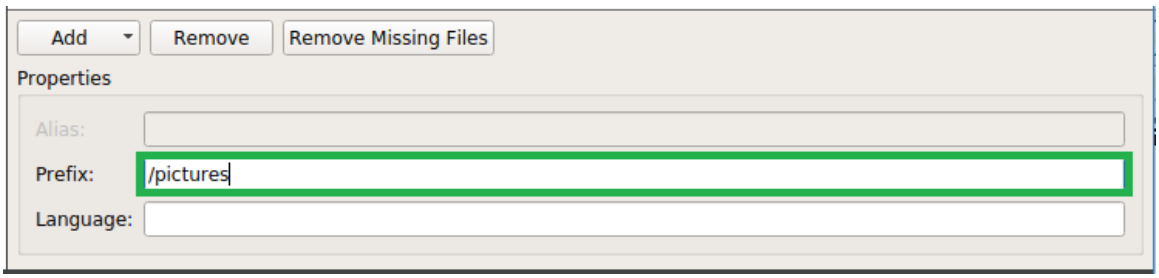

Figure 67. Changing Prefix Name

11. Open the demoGUI.ui file in Qt Creator.

12. Use the right arrow at the top right corner of the GUI to navigate to page two (see Figure 68).

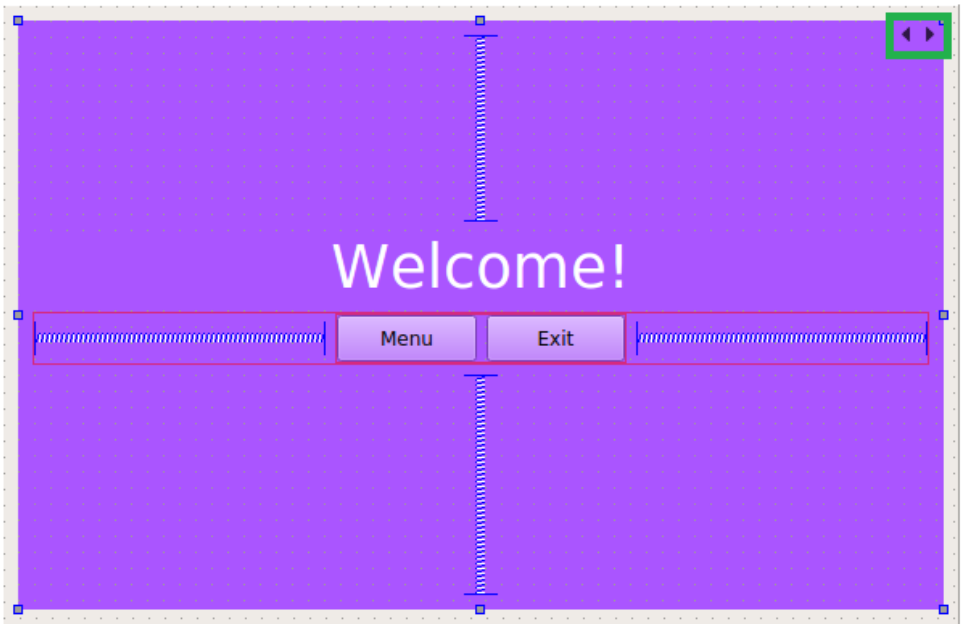

Figure 68. Stacked Widget Navigation Arrows

13. Drag and drop a label onto page two of the GUI.
14. In the Property Editor scroll to the pixmap row within the QLabel section.
15. Click on the right column and select the down arrow (see Figure 69).
16. Select the Choose Resource option.
17. If Resource is blank, then choose File and browse to the picture.

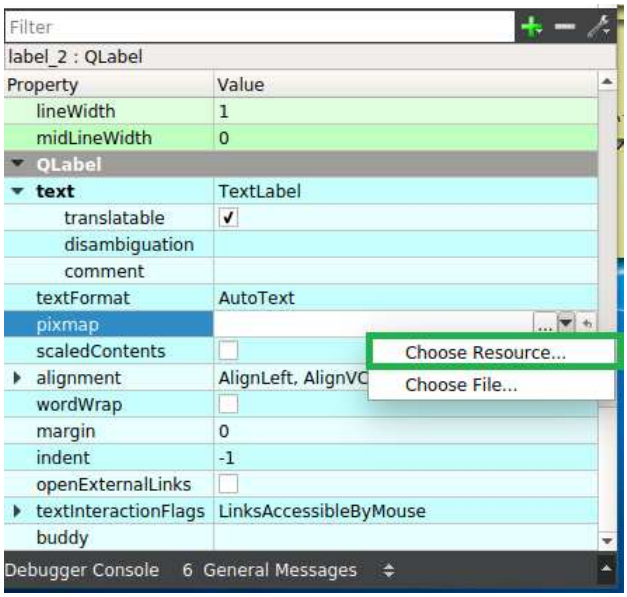

Figure 69. Choose Resource… Option

18. Select the image file you added to your project and click Ok (see Figure 70).

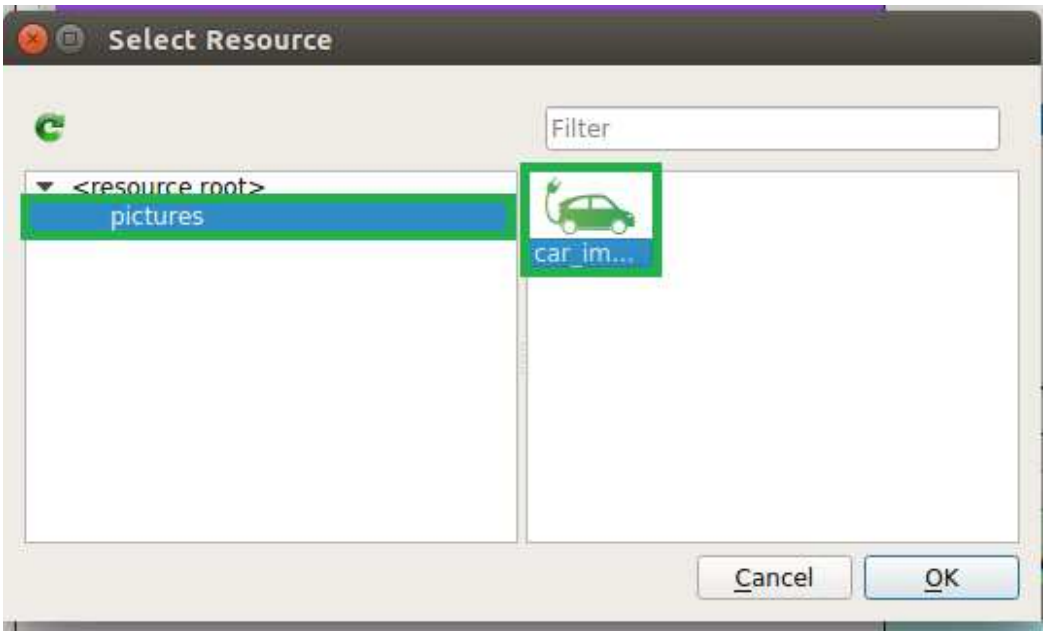

Figure 70. Selecting Resource for demoGUI Project

19. Navigate to the pixmap row again. Under pixmap there is a scaledContents row, check the box next to it (see Figure 71).

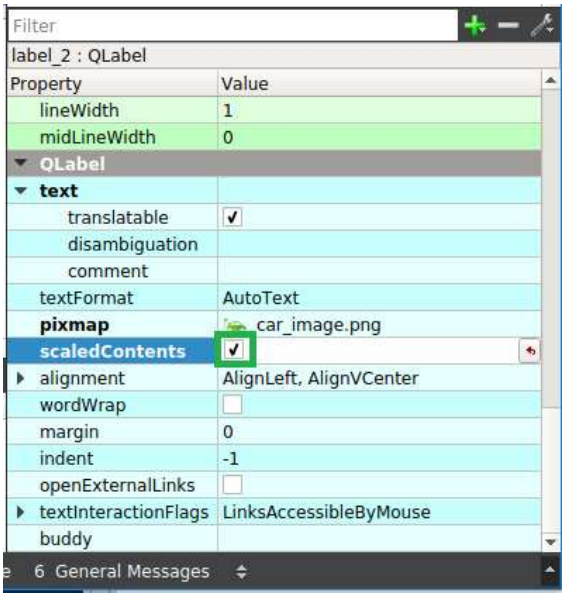

Figure 71. scaledContents Option

The image you selected now appears inside the label you added to the second page of the GUI.

20. Apply a layout to the second page of the GUI, the picture will now occupy your entire screen (see Figure 72).

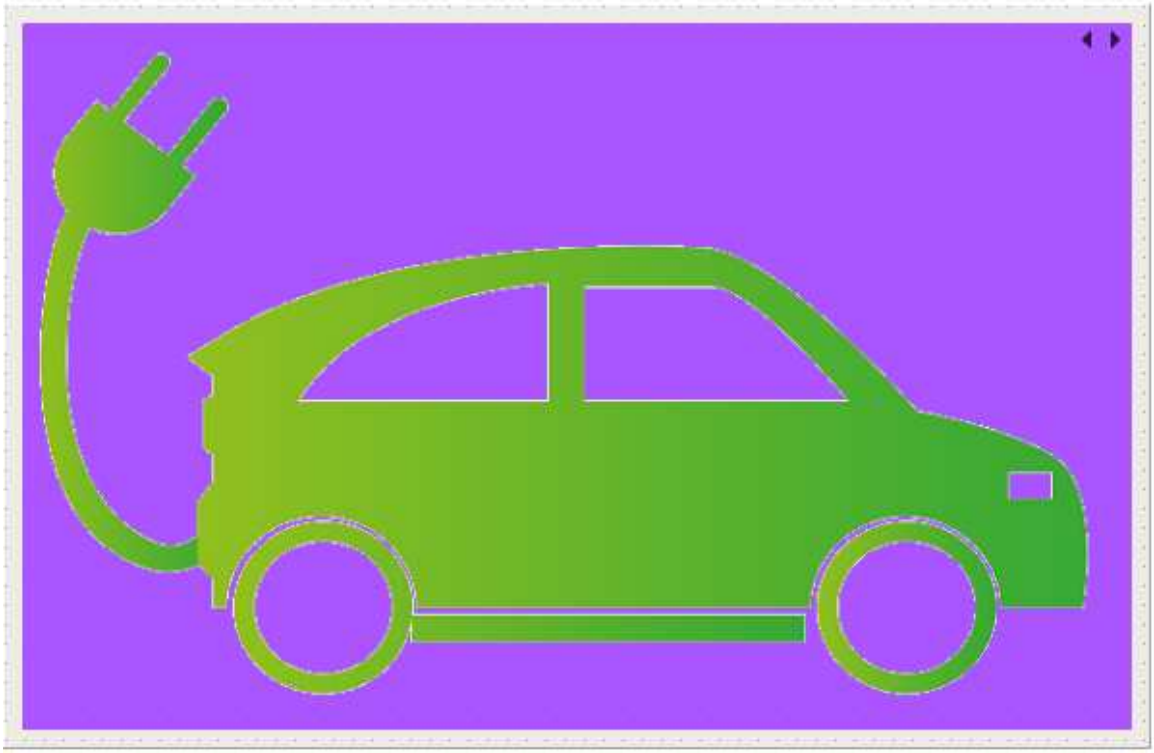

Figure 72. demoGUI Page Two With Image

21. Deploy your GUI to see what it looks like on the EVM.

NOTE: When you build a multipage GUI, it will deploy from whichever page it was on when you pressed the build button. So, if you want to navigate to page two from the home screen, build your application while it is on the welcome page.

5 Customizing Qt Buttons With Pictures

This section covers how to customize Qt buttons by adding pictures to them.

1. Open the demoGUI project.
2. Break the layout of page two of your application.
3. Search the internet to find a picture of a home icon and save it as home-icon.png in the demoGUI project folder.

   Which picture you choose is not important. For demonstration purposes this application report uses Figure 73.

Figure 73. Home Icon

4. Right click on the /pictures folder under Resources/demomedia.qrc.
5. Click Add → Add Existing Files (see Figure 74).

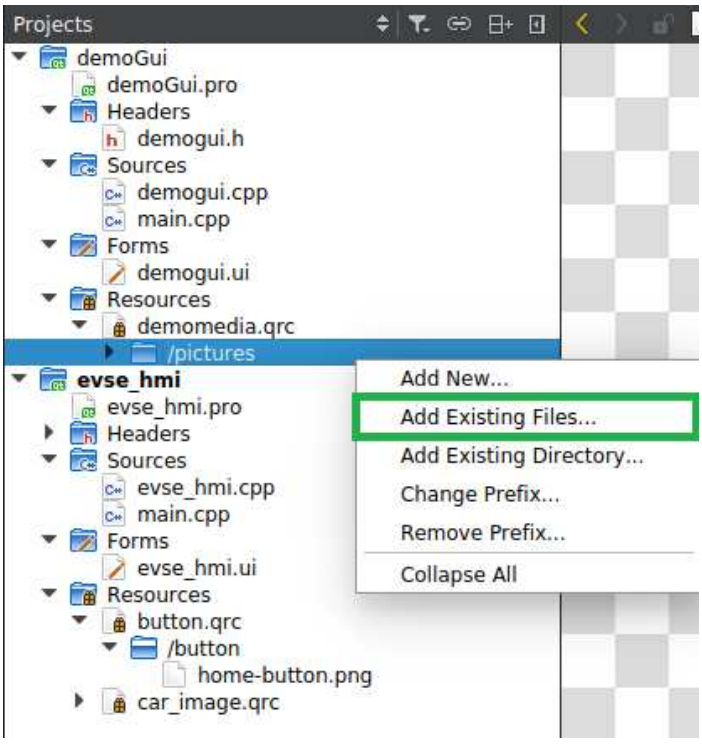

Figure 74. Add Existing Files… Option

6. Select home-icon.png.
7. Navigate to the ui and break the layout of page two of your GUI.
8. Drag and drop a button onto page two of your GUI.
9. Right click on the button and select edit styleSheet.

10. Paste the following code into the style sheet (see Figure 75):

```
QPushButton {
    border-image: url(:/pictures/home-button.png);
    background:none;
}
```

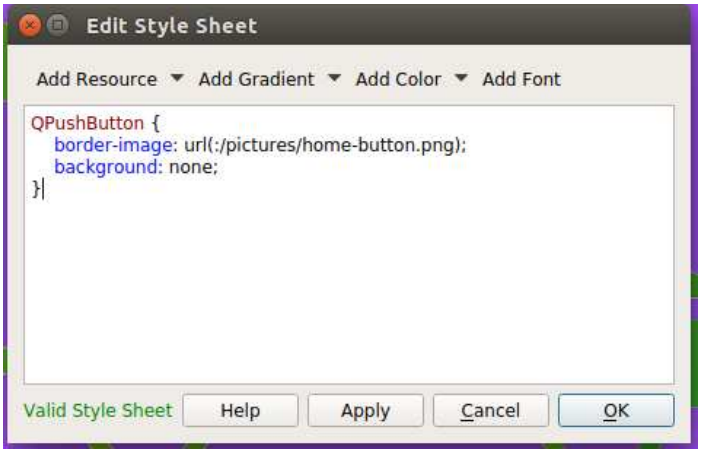

Figure 75. QPushButton Syle Sheet

11. Now the button looks like a home icon you would find in a sophisticated GUI application (see Figure 76).

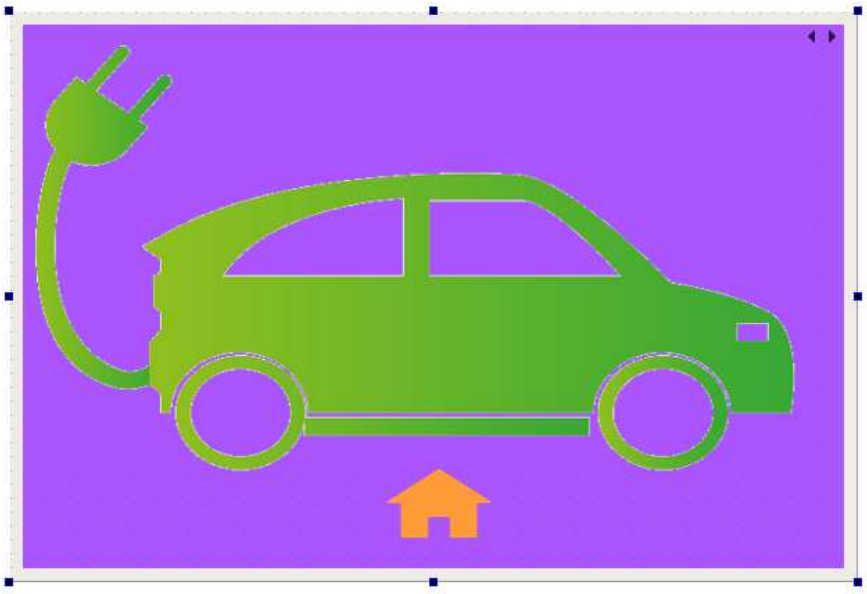

Figure 76. demoGUI Page Two With Home Button

To add functionality to the home icon you just created, see Section 3.1.5.

To make this page resizable, see Section 3.

6 Conclusion

Conflagrations! After following these steps, you should now know how to create a multipage, resizable GUI containing pictures and customized icons.

IMPORTANT NOTICE FOR TI DESIGN INFORMATION AND RESOURCES

Texas Instruments Incorporated ('TI") technical, application or other design advice, services or information, including, but not limited to, reference designs and materials relating to evaluation modules, (collectively, "TI Resources") are intended to assist designers who are developing applications that incorporate TI products; by downloading, accessing or using any particular TI Resource in any way, you (individually or, if you are acting on behalf of a company, your company) agree to use it solely for this purpose and subject to the terms of this Notice.

TI's provision of TI Resources does not expand or otherwise alter TI's applicable published warranties or warranty disclaimers for TI products, and no additional obligations or liabilities arise from TI providing such TI Resources. TI reserves the right to make corrections, enhancements, improvements and other changes to its TI Resources.

You understand and agree that you remain responsible for using your independent analysis, evaluation and judgment in designing your applications and that you have full and exclusive responsibility to assure the safety of your applications and compliance of your applications (and of all TI products used in or for your applications) with all applicable regulations, laws and other applicable requirements. You represent that, with respect to your applications, you have all the necessary expertise to create and implement safeguards that (1) anticipate dangerous consequences of failures, (2) monitor failures and their consequences, and (3) lessen the likelihood of failures that might cause harm and take appropriate actions. You agree that prior to using or distributing any applications that include TI products, you will thoroughly test such applications and the functionality of such TI products as used in such applications. TI has not conducted any testing other than that specifically described in the published documentation for a particular TI Resource.

You are authorized to use, copy and modify any individual TI Resource only in connection with the development of applications that include the TI product(s) identified in such TI Resource. NO OTHER LICENSE, EXPRESS OR IMPLIED, BY ESTOPPEL OR OTHERWISE TO ANY OTHER TI INTELLECTUAL PROPERTY RIGHT, AND NO LICENSE TO ANY TECHNOLOGY OR INTELLECTUAL PROPERTY RIGHT OF TI OR ANY THIRD PARTY IS GRANTED HEREIN, including but not limited to any patent right, copyright, mask work right, or other intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information regarding or referencing third-party products or services does not constitute a license to use such products or services, or a warranty or endorsement thereof. Use of TI Resources may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

TI RESOURCES ARE PROVIDED "AS IS" AND WITH ALL FAULTS. TI DISCLAIMS ALL OTHER WARRANTIES OR REPRESENTATIONS, EXPRESS OR IMPLIED, REGARDING TI RESOURCES OR USE THEREOF, INCLUDING BUT NOT LIMITED TO ACCURACY OR COMPLETENESS, TITLE, ANY EPIDEMIC FAILURE WARRANTY AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

TI SHALL NOT BE LIABLE FOR AND SHALL NOT DEFEND OR INDEMNIFY YOU AGAINST ANY CLAIM, INCLUDING BUT NOT LIMITED TO ANY INFRINGEMENT CLAIM THAT RELATES TO OR IS BASED ON ANY COMBINATION OF PRODUCTS EVEN IF DESCRIBED IN TI RESOURCES OR OTHERWISE. IN NO EVENT SHALL TI BE LIABLE FOR ANY ACTUAL, DIRECT, SPECIAL, COLLATERAL, INDIRECT, PUNITIVE, INCIDENTAL, CONSEQUENTIAL OR EXEMPLARY DAMAGES IN CONNECTION WITH OR ARISING OUT OF TI RESOURCES OR USE THEREOF, AND REGARDLESS OF WHETHER TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

You agree to fully indemnify TI and its representatives against any damages, costs, losses, and/or liabilities arising out of your non-compliance with the terms and provisions of this Notice.

This Notice applies to TI Resources. Additional terms apply to the use and purchase of certain types of materials, TI products and services. These include; without limitation, TI's standard terms for semiconductor products http://www.ti.com/sc/docs/stdterms.htm), evaluation modules, and samples (http://www.ti.com/sc/docs/sampterms.htm).

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265