LESSON 16

CONTROLLING ROBOT MOTION WITH A WEB PAGE

OBJECTIVE

In this lesson, you will learn how to drive your robot in different directions using input from web page hosted on the Raspberry Pi. This web page will allow you to drive forward, reverse, and make left and right turns.

MATERIALS

- Finished mobile robot from Lesson D-11
- Wireless network for connecting multiple WiFi devices
- A computer or laptop with VNC client installed for accessing the Pi remotely

REVIEW CONCEPTS

If you do not feel comfortable with the following concepts, please review them before proceeding.

- Networking and Remote Access (Lesson C-1)
- Command Line Programs (Lesson C-5)
- Web Page Control (Lesson C-16)
- Driving Motors (Lesson D-10)

LESSON

There are many ways that a robot's motion can be controlled, like a joystick, pushbutton switches, or keyboard input. Another option for controlling a robot is to use a web page hosted locally on the robot.

In this lesson, you will create a web page to control the motion of your robot. This web page will be hosted by the Apache server that was installed as part of the RPi Cam Web Interface software that was used to stream camera video to a web page in Level C. The web page will be used to trigger multiple CGI scripts that will control the left and right motors and their speed. The steps used to accomplish this task will be similar to those in Lesson C-16 where you learned to control the color of an RGB LED with a web page. There will be some review in this lesson related to concepts covered in that previous lesson.

WEB PAGE OVERVIEW

The web page that we will create will use the HTML scripting language. Images of direction arrows will be downloaded from our GitHub repository and they will be used to create buttons on the page. When any of these buttons is clicked, a CGI script will be called that will drive the motors in the appropriate direction for that button.

Here is a table that lists the locations of some of the files that will be used for the web page:

Element	Path	Purpose
Web Root	/var/www	Holds files and folders for generating web pages
CGI Scripts	/usr/lib/cgi-bin	Location for CGI scripts for web pages

Any HTML files located in the **`/var/www`** directory will be hosted by the Apache server on the Pi and can be accessed by using the IP address of the Pi.

CGI SCRIPT REVIEW

CGI scripts will be used to run WiringPi commands that will adjust PWM motor speeds and drive the motors as required for forward, reverse, left, and right turns. To change the mode of a pin to an output, PWM, or back to an input we will use:

```
gpio -g mode pin_number new_mode
```

We can use the following command to drive the pin high or low once it's configured as an output:

```
gpio -g write pin_number 0_or_1
```

For PWM pins that need to have the duty cycle adjusted to control motor speed, we can use the following:

```
gpio -g pwm pin_number duty_cycle
```

For more information and examples on each of these commands, you can refer back to the section titled Controlling GPIO Pins with WiringPi in Lesson C-5.

You may remember that newly created CGI scripts won't be executable. They will need to be made executable by running the **chmod** command on the file to add the **+x** attribute:

```
sudo chmod +x /usr/lib/cgi-bin/forward.cgi
```

If you're already in the directory containing the CGI script you would like to modify then you can remove the path from the filename:

```
sudo chmod +x filename.cgi
```

This method will be used to make any CGI scripts that we create throughout this lesson executable.

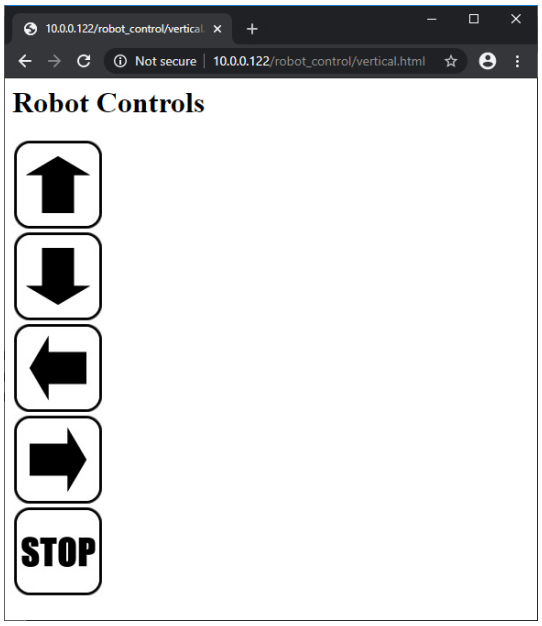
WEB PAGE ELEMENTS

The web page that we will build for controlling the robot will be very similar to the RGB LED control page that you built previously. There are a few new concepts you haven't seen before that will be used to build this page. These new concepts are HTML div containers and javascript onpointerup functionality.

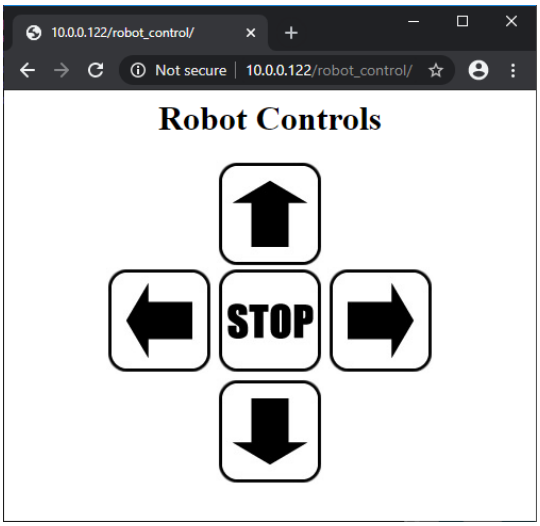
DIV TAGS

The RGB LED control page that you built previously used a series of links down the left side of the page. This left-justified formatting is the default display method used for web pages so it's very easy to create this type of page with very little code.

However, this layout doesn't work well for everything. We will be using five buttons for controlling the robot and this is how they would look if stacked down the left side of the page:



This layout will technically work, but it could be made a lot more user friendly by arranging the buttons more logically based on their direction:



This can be done by enclosing the heading and buttons in div tags. Div tags will create a division or section of the page that can apply styling to any elements inside the div. This div styling can be used for many things, but the example above applies a style called **`text-align:center`** to center everything inside the div.

The HTML code to accomplish this layout will look something like this:

```
<body>
  <div style="text-align:center">
    <h1>Robot Controls</h1>
    <img src="/forward.jpg">
    <br>
    <img src="/left.jpg">
    <img src="/stop.jpg">
    <img src="/right.jpg">
    <br>
    <img src="/reverse.jpg">
  </div>
</body>
```

The open div tag is used to set the alignment style for any elements in the div container. This will apply to the Robot Controls heading and all of the button images below.

The heading will be displayed on its own line. Next the forward image button will be displayed, and a break tag is used to create a new line. The left, stop, and right button images are centered together on the next line, and a break tag is again used to create a new line. The reverse button image is displayed and the closing div tag is used to close the div container.

The break tags must be used to create new lines, or else all of the button images would be centered below the heading on the same line.

JAVASCRIPT ONPOINTERUP EVENTS

In the previous RGB LED control program, we only used the JavaScript event named onpointerdown that called a function when a link was clicked. This worked to change the color of the LED as soon as the link was clicked.

With the direction controls for the robot, it will also be helpful to assign another event to each button that will call a function when the button is released. This will allow you to click and hold down on a button to keep going that direction and release the click when you would like to stop the robot. This will be accomplished by assigning both an onpointerdown event and an onpointerup event to the button:

```
<img src="/forward.jpg" onpointerdown="fwd()" onpointerup="stop()">
```

This button will have the following characteristics:

- The file named **forward.jpg** will be used as the image for the button.
- The function named **fwd()** will be called when the button is pressed.
- The function named **stop()** will be called when the button is released.

The functions named **fwd()** and **stop()** referenced by this button will need to be created earlier in the program to call the appropriate CGI scripts to control motor movements.

BUTTON FUNCTIONALITY OVERVIEW

You may remember from previous lessons related to WiringPi, that GPIO pins are inputs by default. They must be configured as outputs or PWM before they can be used as either an output that will be driven high and low, or as a PWM output. The following GPIO pins will need to be configured before they can be used to drive the robot:

- Right Motor
  - Enable – GPIO12 - PWM for speed control
  - Forward – GPIO24 - Output for high/low drive
  - Reverse – GPIO23 - Output for high/low drive
- Left Motor
  - Enable – GPIO13 – PWM for speed control
  - Forward – GPIO19 - Output for high/low drive
  - Reverse – GPIO26 - Output for high/low drive

This web page will have CGI files for forward, reverse, left, right, and stop that could be used to configure these pins. We could include the pin configuration code in each file, but that will add 6 lines of code to each file in addition to the lines required for driving the motors.

The direction arrow buttons will be configured to run the **`stop()`** function whenever they are released, calling the **`stop.cgi`** script. We can save some code in the direction CGI files by only including the pin setup information in the **`stop.cgi`** file. The pins will be configured every time the Stop button is pressed or any of the direction buttons are released. This means that there will be two ways to get the GPIO pins configured when you first open the web page:

- Press the Stop button and the pins will be configured and ready to drive the robot.
- Press and release a direction button. The first press will be ignored since the pins are not yet configured, but the release will run the **`stop.cgi`** script that will configure the pins. Subsequent presses of any direction buttons will result in movement of the robot.

ACTIVITIES

In the following activities you will:

- Download the images that will be used as buttons on the web page
- Build the CGI scripts that will use WiringPi commands to configure and drive the motors based on the button that is clicked
- Build the web page that will be used to control the robot

ACTIVITY #1 – DOWNLOADING AND UNZIPPING BUTTON IMAGES

In this activity you will download the arrow and stop button images from our GitHub repository so they can be used later to build the web page.

STEP #1

Since this program will take a little while to build, there is no reason to run the robot on batteries throughout the build process. Disconnect battery power input from the Pi by disconnecting the micro USB power connector. Connect the 5V wall power adapter to wall power and then to the micro USB power connection of the Pi. The Pi will power up and automatically connect to your WiFi network.

STEP #2

Connect to the VNC server of the Raspberry Pi using a desktop computer or laptop connected to your WiFi network. Once connected, you should be viewing the Desktop of the Pi just as if you had a monitor, keyboard, and mouse connected directly to the Pi.

STEP #3

Open a Terminal window by clicking on the Terminal icon in the upper-left menu bar.

Run the following command to change into the **/var/www** directory that's used to hold files and folders hosted by the Apache web server:

```
cd /var/www
```

STEP #4

You will now be located in the **/var/www** directory. Let's make a new folder called **images** that will hold the button images that we will use to build the web page later in this lesson.

Run the following command to create this folder:

```
sudo mkdir images
```

Your **pi** user will not have write privileges for this new directory by default. Run the command below to change the owner and group for the new folder to **pi** and **www-data**:

```
sudo chown pi:www-data images
```

Now that the **images** directory has been created, and you have changed ownership to give yourself write permissions, use the following command to move into the new directory:

```
cd images
```

STEP #5

Since you're now located in the **images** folder, download the .zip file containing the arrow and stop button images by running the following command:

```
curl
https://raw.githubusercontent.com/42electronics/level_d/master/lesson_16/arrow_images.
zip > arrow_images.zip
```

This is a very long command that cannot be displayed on one line above, but it should be run as one command in your CLI.

STEP #6

The .zip file contains button images for forward, reverse, left, right, stop, hello, and good-bye. Unzip the image files by running this command:

```
unzip arrow_images.zip
```

The button images will be inflated from the .zip file and stored in your current directory which is **/var/www/images**.

STEP #7

Use the **ls** command to list all of the files on your current directory to verify the images were unzipped properly. The output should look list the .zip file and seven unzipped images:

If the output from your ls command does not match the image above, go back to Step #6 and attempt to unzip the file again. If that does not help, repeat Steps #5 and #6 to download a new copy of the .zip file, and attempt the unzip process again. Verify your file listing matches the image above before proceeding to the next activity.

ACTIVITY #2 – BUILD REQUIRED CGI SCRIPTS

In this activity you will build the CGI scripts that will be used for configuring and driving the GPIO pins that will control the speed and direction of the motors.

STEP #1

The CGI scripts for the web page that we're building will be located in the **/usr/lib/cgi-bin** directory. Using the Terminal window from the previous activity, run the following command to move into the **/usr/lib/cgi-bin** directory:

```
cd /usr/lib/cgi-bin
```

STEP #2

The **stop.cgi** script will be the most complicated, so we'll start there. This script will perform the following actions:

- Establish the scripting language used in the file as bash
- Configure motor forward and reverse drive pins as outputs
- Configure motor enable pins as PWM outputs
- Pull all forward and reverse pins low to stop robot motion
- Return a Status 204 message to the Apache web server

Create and open a new file named **stop.cgi** by running the following command:

```
sudo nano stop.cgi
```

The new file will now be open in the Nano file editor. Add the content below to the **stop.cgi** file:

```
#!/bin/bash

#motor fwd/rev as outputs
gpio -g mode 19 out
gpio -g mode 26 out
gpio -g mode 23 out
gpio -g mode 24 out

#motor enable as pwm
gpio -g mode 12 pwm
gpio -g mode 13 pwm

#motor fwd/rev all low to stop
gpio -g write 19 0
gpio -g write 26 0
gpio -g write 23 0
gpio -g write 24 0

echo "Status: 204 No Content"
echo "Content-type: text/plain"
echo ""
```

If you're unclear on why the **/bin/bash** or **echo** lines above are being used, please refer back to the section titled Creating and Running CGI Scripts in Lesson C-16.

Use the CTRL-X keyboard shortcut to exit the file, press **y** to save changes, and press the Enter key to confirm the filename that will be written.

STEP #3

The **fwd.cgi** script will set the PWM speed of both motors to 1000 and drive both motors forward.

Create and open a new file named **fwd.cgi** by running the following command:

```
sudo nano fwd.cgi
```

The new file will now be open in the Nano file editor. Add the content below to the **fwd.cgi** file:

```
#!/bin/bash

#right motor speed
gpio -g pwm 12 1000

#left motor speed
gpio -g pwm 13 1000

#right motor forward
gpio -g write 24 1

#left motor forward
gpio -g write 19 1

echo "Status: 204 No Content"
echo "Content-type: text/plain"
echo ""
```

Use the CTRL-X keyboard shortcut to exit the file, press **y** to save changes, and press the Enter key to confirm the filename that will be written.

STEP #4

The **rev.cgi** script will set the PWM speed of both motors to 1000 and drive both motors in reverse.

Create and open a new file named **rev.cgi** by running the following command:

```
sudo nano rev.cgi
```

The new file will now be open in the Nano file editor. Add the content below to the **rev.cgi** file:

```
#!/bin/bash

#right motor speed
gpio -g pwm 12 1000

#left motor speed
gpio -g pwm 13 1000

#right motor reverse
gpio -g write 23 1

#left motor reverse
gpio -g write 26 1

echo "Status: 204 No Content"
echo "Content-type: text/plain"
echo ""
```

Use the CTRL-X keyboard shortcut to exit the file, press **y** to save changes, and press the Enter key to confirm the filename that will be written.

STEP #5

The **left.cgi** script will set the PWM speed of both motors to 850, drive the right motor forward, and the left motor in reverse.

Create and open a new file named **left.cgi** by running the following command:

```
sudo nano left.cgi
```

The new file will now be open in the Nano file editor. Add the content below to the **left.cgi** file:

```
#!/bin/bash

#right motor speed
gpio -g pwm 12 850

#left motor speed
gpio -g pwm 13 850

#right motor forward
gpio -g write 24 1

#left motor reverse
gpio -g write 26 1

echo "Status: 204 No Content"
echo "Content-type: text/plain"
echo ""
```

Use the CTRL-X keyboard shortcut to exit the file, press **y** to save changes, and press the Enter key to confirm the filename that will be written.

STEP #6

The **right.cgi** script will set the PWM speed of both motors to 850, drive the right motor in reverse, and the left motor forward.

Create and open a new file named **right.cgi** by running the following command:

```
sudo nano right.cgi
```

The new file will now be open in the Nano file editor. Add the content below to the **right.cgi** file:

```
#!/bin/bash

#right motor speed
gpio -g pwm 12 850

#left motor speed
gpio -g pwm 13 850

#right motor reverse
gpio -g write 23 1

#left motor forward
gpio -g write 19 1

echo "Status: 204 No Content"
echo "Content-type: text/plain"
echo ""
```

Use the CTRL-X keyboard shortcut to exit the file, press **y** to save changes, and press the Enter key to confirm the filename that will be written.

STEP #7

Use the **ls** command to confirm that **stop.cgi**, **fwd.cgi**, **rev.cgi**, **left.cgi**, and **right.cgi** are present in this folder. There will also likely still be other CGI scripts present from your work in Lesson C-16 when controlling the RGB LED.

At this point, the CGI scripts for this project have been created but the web server will not have execute permissions for the files. Add execute permissions to all files in that directory by running the following command:

```
sudo chmod +x *.cgi
```

The * will make the command apply to any file ending in .**cgi** in your current working directory. This will add execute permissions to any files that are not yet executable but will not affect the **.cgi** files from your previous project since they were already executable.

Leave this Terminal window open as it will be used in the next activity.

ACTIVITY #3 – BUILD THE WEB PAGE

In this activity you will build the web page that will use the button images from Activity #1 and the CGI scripts you created in Activity #2. The finished web page will allow you to control the movement of the robot by clicking the arrow buttons on the web page.

STEP #1

The web page will be located in a new directory named **/var/www/robot_control**. Run the command below to move into the /var/www directory:

```
cd /var/www/
```

Next, create the directory named **robot_control** using the command below:

```
sudo mkdir robot_control
```

Run the command below to move from your current directory into the **/var/www/robot_control** directory:

```
cd robot_control
```

Now that you're in the **robot_control** directory, create and open a new web page file named **index.html** using the following command:

```
sudo nano index.html
```

The new file will now be open in the Nano file editor.

STEP #2

This web page will follow much of the same format as the RGB LED page that you created previously. If you're unsure about any commands or HTML tags that are used in this and upcoming steps, please refer back to Lesson C-16.

The first step in building the web page will be creating the **html**, **head**, and **body** tags for the page. Add the following tags to the index.html file:

```
<html>
  <head>

  </head>
  <body>

  </body>
</html>
```

Make sure to use the forward-slash character to start the closing tag for each section. Using indentation and carriage returns to make your file match the code above will make it easier to add more code to the proper section later.

STEP #3

The web page now has the **html**, **head**, and **body** sections, but they are currently empty. In this step, we will add the script information to the **head** section. This script section will do the following:

- Specify the script language as **Javascript**
- Create a function named **fwd()** that will call **/cgi-bin/fwd.cgi**
- Create a function named **rev()** that will call **/cgi-bin/rev.cgi**
- Create a function named **right()** that will call **/cgi-bin/right.cgi**
- Create a function named **left()** that will call **/cgi-bin/left.cgi**
- Create a function named **stop()** that will call **/cgi-bin/stop.cgi**

Add the following highlighted block of code to your program between the **head** tags:

```
<html>
  <head>
    <script Language="Javascript">
      function fwd()
        {document.location="/cgi-bin/fwd.cgi";}
      function rev()
        {document.location="/cgi-bin/rev.cgi";}
      function right()
        {document.location="/cgi-bin/right.cgi";}
      function left()
        {document.location="/cgi-bin/left.cgi";}
      function stop()
        {document.location="/cgi-bin/stop.cgi";}
    </script>
  </head>
  <body>
```

Ensure all the curly braces, quotation marks, and semicolons match the code above exactly or the functions will not execute the CGI scripts properly when called.

STEP #4

The **head** section is now complete and it's time to build the **body** section of the page. We will start by adding a **div** that will **center-align** all of its contents and a heading that will display **Robot Controls** in the **h1** font size.

Add the following highlighted block of code to your program between the **body** tags:

```
    </script>
  </head>
  <body>
    <div style="text-align:center">
      <h1>Robot Controls</h1>

    </div>
  </body>
```

STEP #5

The **div** is now ready for the buttons. This is how the buttons will be arranged with their image file names, **onpointerdown,** and **onpointerup** behavior:

Row	Image File	Onpointerdown Event	Onpointerup Event
1	/images/forward.jpg	fwd()	stop()
2	/ images/left.jpg	rev()	stop()
2	/ images/stop.jpg	stop()	stop()
2	/ images/right.jpg	right()	stop()
3	/ images/reverse.jpg	rev()	stop()

Add the following highlighted block of code to your program between the heading and the closing **div** tag:

```
    <div style="text-align:center">
      <h1>Robot Controls</h1>
      <img src="/images/forward.jpg" onpointerdown="fwd()" onpointerup="stop()">
      <br>
      <img src="/images/left.jpg" onpointerdown="left()" onpointerup="stop()">
      <img src="/images/stop.jpg" onpointerdown="stop(event)" onpointerup="stop()">
      <img src="/images/right.jpg" onpointerdown="right()" onpointerup="stop()">
      <br>
      <img src="/images/reverse.jpg" onpointerdown="rev()" onpointerup="stop()">
    </div>
  </body>
</html>
```

Be sure to include the **
** tags to add new lines for the second and third rows of buttons. Without break tags, all the buttons would be displayed from left to right on the same line.

STEP #6

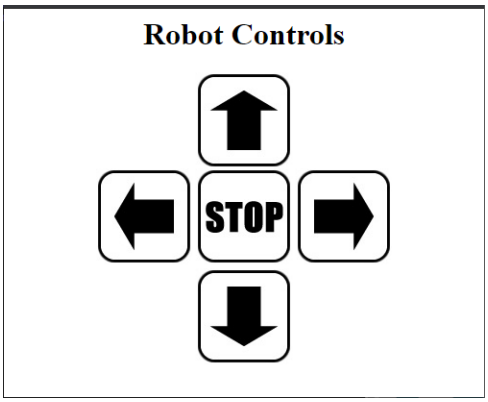
The web page is now complete. Use the CTRL-X keyboard shortcut to exit the file, press **y** to save changes, and press the Enter key to confirm the filename that will be written.

This page is now ready to use for controlling the movement of your robot from over the network. Before switching the Pi away from AC wall power, let's test the web page and CGI scripts to make sure they behave as expected. Using the web browser on another device on your home network, point the browser to the IP address of your Raspberry Pi, followed by **/robot_control**. Since the web page in this folder is named index.html, Apache will automatically load that file when this directory is requested.

If your Raspberry Pi is currently at IP address 192.168.1.100, then the full address to access this page would be:

```
192.168.1.100/robot_control
```

Once the web page is open, the heading and button layout should appear like this:



Click on each of the buttons to verify that no CGI or web page errors are displayed. The motors will not drive but the CGI scripts will still be running.

Here is a list of problems you might encounter and how to troubleshoot each of them:

- Page is not found – Verify that the IP address of the Pi is correct. Also verify the web page is named index.html and is located in the **/var/www/robot_control** folder.

- Page loads but layout is wrong – Double-check the contents of the div in your page. This code is responsible for the layout of the page, so missing elements or characters could break the formatting of the page.
- Internal server error when clicking a button – The web server does not have permission to run the CGI script associated with that button. Navigate back to **/usr/lib/cgi-bin** and use Step #6 of Activity #2 to make all scripts in that directory are executable.

Here is the full code of the web page in case you need to do any further troubleshooting:

```
<html>
  <head>
    <script Language="Javascript">
      function fwd()
        {document.location="/cgi-bin/fwd.cgi";}
      function rev()
        {document.location="/cgi-bin/rev.cgi";}
      function right()
        {document.location="/cgi-bin/right.cgi";}
      function left()
        {document.location="/cgi-bin/left.cgi";}
      function stop()
        {document.location="/cgi-bin/stop.cgi";}
    </script>
  </head>
  <body>
    <div style="text-align:center">
      <h1>Robot Controls</h1>
      <img src="/images/forward.jpg" onpointerdown="fwd()" onpointerup="stop()">
      <br>
      <img src="/images/left.jpg" onpointerdown="left()" onpointerup="stop()">
      <img src="/images/stop.jpg" onpointerdown="stop()" onpointerup="stop()">
      <img src="/images/right.jpg" onpointerdown="right()" onpointerup="stop()">
      <br>
      <img src="/images/reverse.jpg" onpointerdown="rev()" onpointerup="stop()">
    </div>
  </body>
</html>
```

Do not proceed to the next step until your web page is loading, the button layout is correct, and the buttons do not generate errors when clicked or released.

STEP #7

It's now time to drive the robot motors using battery power, so the robot must be switched from AC wall power over to battery power. Use your VNC connection to the Pi to complete a proper shutdown to avoid any data loss. Once the SD card activity LED is no longer flashing green, remove the micro USB power cable from the Pi.

Connect the micro USB power cable from the on-board 5V converter to the Pi. Next, connect the 9-volt battery pack to the 5V converter, powering up the robot.

Caution – When running on power from the battery pack, the motors will be powered up and ready to move. **Make sure that your robot is on the floor or in some location where driving off the edge of a desk or tabletop is not a possibility**, as this could result in damage to the robot that cannot be easily repaired.

STEP #8

As soon as the Pi is done booting up, the robot_control web page will be available. Using the web browser on another device on your home network, point the browser to the IP address of your Raspberry Pi, followed by **`/robot_control`**.

Click and hold each direction button to determine if each button behaves as expected:

- Clicking on a button should make the robot drive in that direction.
- Holding down a button should make the robot continue in that direction.
- Releasing a button should make the robot stop.

If the robot does not move in the expected direction when a button is pressed, then you should check the following:

- Check web page code to ensure proper function names are being called by that button and that function names are calling the correct CGI script.
- Make sure associated CGI script in **`/usr/lib/cgi-bin`** is performing the correct motor drive operations for moving in that direction. For example, **`fwd.cgi`** should be driving the forward lines on both motors high. If not, the robot will not drive forward when **`fwd.cgi`** runs. Also, double-check that PWM code is present in problematic CGI files. No PWM signal means that a motor will not be enabled, and it will not move.

STEP #9

We used PWM values of 1000 for the forward and reverse CGI files. Your robot may not be driving perfectly straight in forward or reverse, so the PWM motor values in **`fwd.cgi`** and **`rev.cgi`** may need to be adjusted. Adjust the values as needed to correct slight turns:

- Turning slightly left – Right motor is driving too fast. Use smaller value for right motor speed.
- Turning slightly right – Left motor is driving too fast. Use smaller value for left motor speed.

We used PWM values of 850 for the left and right CGI files. Your robot may be turning to quickly or slowly with these values, so the PWM motor values in **`left.cgi`** and **`right.cgi`** may need to be adjusted. Adjust the values as needed to obtain a good speed for turns:

- Turning too quickly – PWM value is too high. Try using a lower PWM value for the left and right motors.
- Turning too slowly – PWM value is too low. Try using a higher PWM value for the left and right motors.

STEP #10

When you're done driving the robot using the web page, make sure to properly shut down the Raspberry Pi using your VNC connection and disconnect the battery pack to avoid unnecessarily draining the batteries. Do not continue to run the robot if the red power LED on the Pi is flashing or turning off. Discontinue use of the robot immediately and charge or replace the batteries. Running the robot in a low battery pack could result in corruption or data loss on the SD card in the Pi.

QUESTIONS FOR UNDERSTANDING

1. What is the name of the event that's used to assign behavior to the release of a clicked button?

2. Will the heading below be left justified, centered, or right justified in the div container below?

```
<div style="text-align:center">
  <h1>Robot Controls</h1>
</div>
```

3. What tag is used to separate the button images in the web page from Activity #3 into three rows so they can be displayed in a direction-friendly layout?

Answers can be found on the next page.

ANSWERS TO THE QUESTIONS FOR UNDERSTANDING

1. *What is the name of the JavaScript event that's used to assign behavior to the release of a clicked button?*

   ANSWER: The **onpointerup** event can be used to assign an action that will be performed when a button is released.

2. *Will the heading below be left justified, centered, or right justified in the div container below?*

```
<div style="text-align:center">
  <h1>Robot Controls</h1>
</div>
```

   ANSWER: The heading displaying **Robot Controls** will be centered in the div container. The style of **text-align:center** will be applied to all elements within the div.

3. *What tag is used to separate the button images in the web page from Activity #3 into three rows so they can be displayed in a direction-friendly layout?*

   ANSWER: The break tag or **
** is used to create new lines for the button images:

```
<img src="/images/forward.jpg" onpointerdown="fwd()" onpointerup="stop()">
<br>
<img src="/images/left.jpg" onpointerdown="left()" onpointerup="stop()">
<img src="/images/stop.jpg" onpointerdown="stop(event)" onpointerup="stop()">
<img src="/images/right.jpg" onpointerdown="right()" onpointerup="stop()">
<br>
<img src="/images/reverse.jpg" onpointerdown="rev()" onpointerup="stop()">
```

CONCLUSION

In this lesson, you learned how to create an HTML web page that uses CGI scripts to drive your robot in different directions. This same web page concept will be used again in upcoming lessons where more functionality will be added to the web pages used to control the robot.

In the next lesson, you will add video to a web page used to control the robot. This will enable you to control the movement of the robot while monitoring its movements using the video feed from the onboard camera.