Web Based Oscilloscope

Introduction

This app demonstrates the use of Python, Javascript, PHP, and HTML to implement a web based oscilloscope. An oscilloscope graphically displays an electrical signal with various user selectable parameters such as vertical attenuation, triggered sweep, and time base. This app also demonstrates how to hookup an I2C serial i/o device to the Raspberry Pi's GPIO.

Referring to figure 1, the user interacts with the Raspberry Pi web service via a browser running on computer or mobile unit. Whenever the user starts the oscilloscope, or interacts with one of the controls on the web page, a PHP script executes a Python script as a shell command running in the background as a process. The Python script collects data from the analog to digital converter (ADC) connected to the Raspberry Pi and periodically outputs the data to an output data file. At regular intervals Javascript running on the browser reads this file and graphically displays the signal waveform.

The oscilloscope application demonstrates distributed processing in the acquisition of data, and the presentation of that data. The application places the burden of data acquisition on the Raspberry Pi and the Python script, while Javascript running on the web browser takes the burden of graphically displaying the data by using the HTML canvas element.

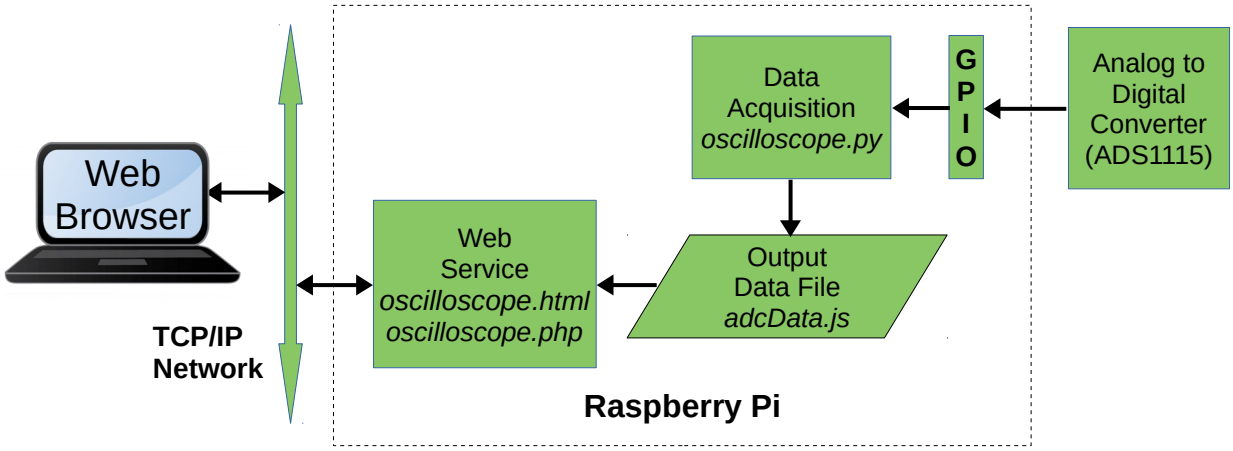

Figure 1. Block diagram showing logical relationship between web client, Raspberry Pi, and the analog to digital converter.

Hardware Hookup

Referring to figure 2, the ADS1115 breakout pin VDD is connected 3.3 Volts via header pin 1, while breakout pin GND connected to ground via header pin 6. Breakout serial clock (SCL) is connected to GPIO SCL, which is GPIO pin 3 (pin 5 on the header); and breakout serial data (SDA) is connected to GPIO SDA, which is GPIO pin 2, (pin 3 on the header). If you also have the function generator app hooked up, you can use the function generator as an input to the oscilloscope. The dotted line from breakout pin A0 shows where the input to the oscilloscope ADC should be connected.

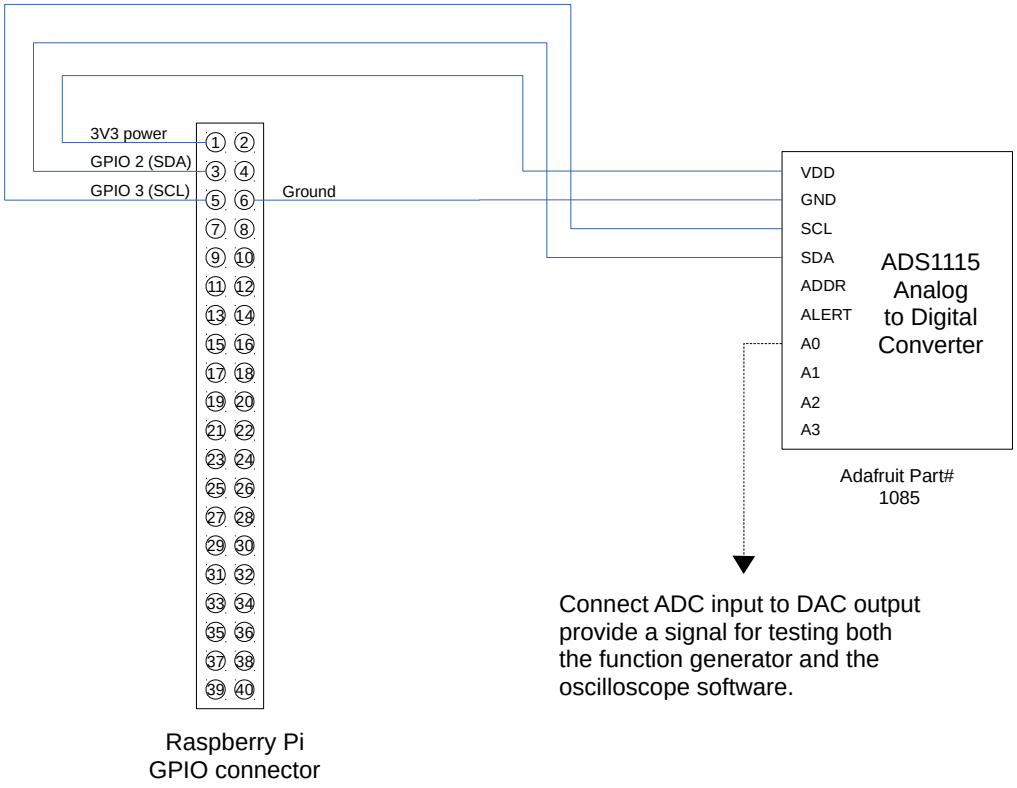

Figure 2. Schematic diagram of the analog to digital converter connected to the Raspberry Pi GPIO.

Software Description

The Python script *oscilloscope.py* is at the heart of the function generator app. The script can collect data samples from the ADC at different sample rates and sample sizes (the number of samples collected per acquisition cycle). The oscilloscope time-base setting determines both the sample rate and sample size. The command syntax is

```
oscilloscope.py [-kd] [-n SIZE] [-r RATE]
```

where

- -n number of samples collected per acquisition cycle;
- -r the rate of sample collection;
- -k kill all running instances of *oscilloscope.py*, including this instance
- -d run in verbose debug mode

The number of samples collected per acquisition cycle determines the number of samples that get written to the output data file *adcData.js*. The rate of sample collection determines the number of times per second samples are read from the ADC.

When started up, the script first kills any previously launched instances of the script. Once the script has been started in the background, it then continues to run in the background until killed. As long as the script continues to run, data gets acquired and written to the output data file, and the oscilloscope displays a waveform.

Executing the script with the -k option kills this previously running instance and then exits the current instance. That is, to stop data acquisition, execute the following on the command line

```
oscilloscope.py -k
```

When the user opens the HTML document *oscilloscope.html* in the browser, Javascript submits a form to be acted upon by the PHP script *oscilloscope.php*. When the user changes a control on the oscilloscope web page, the web page submits the run state and acquisition parameters to the PHP script. The PHP script executes the Python script *oscilloscope.py* in a shell with the appropriate arguments depending on the time base parameter submitted by *oscilloscope.html*.

The above procedure may be summarized in the following steps

1. Open the oscilloscope web page *oscilloscope.html* in the browser. Javascript submits the OFF runstate to the PHP script *oscilloscope.php*.
2. The PHP script executes the Python script *oscilloscope.py* as a shell command with the -k option to kill any previously running instances of itself.
3. If the user selects the start button, Javascript will submit the “run” state along with sample rate and sample size to the PHP script.
4. Else if the function generator is in the run state and the user changes the time-base control on the web page, Javascript submits new values for sample rate and size to the PHP script.
5. The PHP script executes the Python script as a shell command with the appropriate options depending on sample rate and size.
6. Return to step 3.

The ADS1115 Module

The *ads1115.py* module acts as a hardware abstraction layer providing a software interface between higher level calling routines and the ADS1115 device. The module defines a class containing all the necessary class methods to read and write the ADS1115 analog to digital analog converter. Class methods handle conversion and digital formatting of input and output data.

Software Installation

Important notes:

1. The oscilloscope application should be installed on a Linux host which meets the following requirements:

    - The server software should be installed on a recent distribution such as Debian or Raspberry Pi OS.
    - Apache 2 should be installed and configured to allow serving HTML documents from the user’s public_html folder.
    - PHP 7 should be installed and configured to allow running PHP scripts from the user pi public_html folder.
    - The Python I2C smbus library should be installed.
    - Python 3 usually comes per-installed in virtually all Linux distributions. Type “python3” at a command line prompt to verify Python has been installed.

2. See *Setting Up the Raspberry Pi for Internet of Things* for detailed notes on how to set up a Raspberry Pi server, and how to install and configure the components in note 1 above.

Software Inventory

The following software items from the repository need to be installed on the Raspberry Pi

oscilloscope html folder:

- index.html

- oscilloscope.html
- oscilloscope.php

oscilloscope bin folder:

- oscilloscope.py
- ads1115.py

Installing on a Raspberry Pi

Note that the following installation procedure assumes that the document root for HTML documents will be the user's public_html folder. Typically the full path name to this folder will be something like **/home/{user}/public_html**, where **{user}** is the name of the user account hosting node power. The following steps will assume the oscilloscope app is running under the **pi** user account. The **pi** account is the default account when Raspberry Pi OS is first installed on a Raspberry Pi.

1. Follow the instructions in *Setting Up the Raspberry Pi for Internet of Things* hookup guide to install and configure web services on your Raspberry Pi.

2. If it doesn't already exist, use **mkdir** to create a folder **public_html** in the **pi** user's home folder. The full path should look like **/home/pi/public_html**.

3. In the p**ublic_html** folder, use **mkdir** to create a folder called **oscilloscope** to contain the oscilloscope HTML and PHP files. The full path should look like **/home/pi/public_html/oscilloscope/**.

4. Move all the contents of the repository **oscilloscope/html** sub-folder to the **oscilloscope** folder created in step 3.

5. Move all the contents of the repository **oscilloscope/bin** sub-folder to the **bin** folder in the user pi home folder. The full path to this folder should look like **/home/pi/bin/**.

6. Create a folder in the temporary file system for dynamic content, and modify the folder's ownership and permissions to allow the Apache www-data user to have access.

   ```
   mkdir /tmp/oscilloscope
   sudo chown :www-data /tmp/oscilloscope
   chmod g+w /tmp/oscilloscope
   ```

7. The commands in step 6 may be placed in a startup shell script and run at boot up time by launching a startup script with the **su** command from **/etc/rc.local**. For example, place the above commands in a script **/home/pi/bin/startup.sh** and place the following line in the **/etc/rc.local** file.

   ```
   (su - pi -c "~/bin/startup.sh")&
   ```

   Whenever the host boots up, the **startup.sh** script will run the commands in step 6. Be sure that you grant **startup.sh** execute permissions by running

   ```
   chmod u+x ~/bin/startup.sh
   ```

8. In the **oscilloscope** folder created in step 3, create a symbolic link to **/tmp/oscilloscope** in the temporary file system by running

   ```
   ln -s /tmp/oscilloscope dynamic
   ```

9. Acting as superuser open **/etc/sudoers** in a text editor. Add the following line to the end of the file

   ```
   www-data ALL=(ALL) NOPASSWD: /home/pi/bin/oscilloscope.py
   ```

   Adding this line gives the Apache www-data user permission to execute the *oscilloscope.py* script in the background from the *oscilloscope.php* script.

10. Open the Raspberry Pi home page in a browser running on a computer connected to the same network as the Pi. Click on the “Oscilloscope” link to open the oscilloscope web page. You should be able to apply a signal to the ADC and see the waveform on the oscilloscope. You should be able to see the number of displayed cycles change when changing the time-base control on the web page. Likewise you should be able to see the amplitude change when changing the attenuation control on the web page.

This completes installation of the function generator software.