BrewPi Remix

Release 0.8.0

Jul 12, 2022

Contents:

1 Basic Configuration of your Raspberry Pi **3**
- 1.1 Installing Raspbian on your Raspberry Pi - Easy Mode . . . 3
- 1.2 Going Headless - No Monitor, Mouse or Keyboard Needed . . . 3
- 1.3 Using your System . . . 4
- 1.4 Your RPi's Name . . . 4
- 1.5 Updating programs . . . 5
- 1.6 Cleaning Local Repositories . . . 5
- 1.7 Updating firmware . . . 5
- 1.8 Work Complete . . . 5

2 Automated Installation of BrewPi **7**
- 2.1 Running the Install Script . . . 7
- 2.2 After the Install . . . 7

3 Configuring your Devices **9**
- 3.1 Your Devices After Programming . . . 9
- 3.2 Receiving the Device List . . . 9
- 3.3 Installing new devices and assigning them to a function . . . 12
- 3.4 Uninstalling a device . . . 13

4 Specific Gravity Tracking **15**
- 4.1 Tilt . . . 15
- 4.2 iSpindel . . . 15

5 BrewPi Remix Manual Installation **17**

6 Programming your Arduino **19**
- 6.1 Programming via Command Line . . . 19
- 6.2 Programming via Web Interface . . . 20
- 6.3 Troubleshooting . . . 20

7 Running Multiple Chambers on the Same RaspberryPi **21**
- 7.1 Clean Install Required . . . 21
- 7.2 Multi-Chamber Installation . . . 21
- 7.3 Arduino Selection . . . 22
- 7.4 Adding Additional Chambers . . . 22
- 7.5 Multi-Chamber Index . . . 22

8 Updating BrewPi **23**
8.1 Scripts After Update . . . 23
8.2 Updating Multi-Chamber . . . 24

9 BrewPi Remix API **25**
9.1 Arduino Commands . . . 26
9.2 Arduino Message Types . . . 27
9.3 BEERSOCKET Commands . . . 27
9.4 Config.cfg Settings . . . 29

10 Calibrating your Probes **31**
10.1 Calibration Steps: . . . 31
10.2 Setting Calibration Adjustment . . . 31
10.3 Alternative method - relative calibration . . . 32
10.4 Accuracy . . . 32

11 How to Compile the Code from Source **33**
11.1 Disclaimer and Limitation . . . 33
11.2 Setting up the Development Environment . . . 33
11.3 Clone the Repository . . . 34
11.4 Compile the Source Code . . . 34

This documentation is written in reStructured text format and is part of the BrewPi Remix Project. If you see something that could use some updating, please let me know. Even better: Fix it yourself and send me a pull request on GitHub.

This documentation is intended for the beginner, who has to find a teenager just to change his or her wallpaper. It should also contain ample "just the facts" information for more experienced users. A deficiency in the documentation (i.e. if you get lost) is considered a bug, and you should bring it to my attention.

CHAPTER 1

Basic Configuration of your Raspberry Pi

This step-by-step guide will help you to set everything up on your Raspberry Pi and Arduino. Current technology is assumed throughout. Testing is done during development on the most recent platform available. At the time of the last update to this section, that was a Raspberry Pi 4, with the latest version of Bullseye installed. No intentional differences exist between this platform and others, especially older configurations, but changes exist. If you have issues, you should always upgrade to the Raspberry Pi website's latest Raspberry Pi OS version. If you deviate from that, you should know enough to help translate these instructions for your own use.

Please also note that while BrewPi Remix will probably work with other *nix flavors (almost assuredly Raspbian Desktop on a PC, likely Debian, and probably Ubuntu;) these are not tested. You will undoubtedly have issues I've simply not tested for and am not sure I want to worry about. Raspbian on the Raspberry Pi is the only officially supported OS.

1.1 Installing Raspbian on your Raspberry Pi - Easy Mode

The easiest cross-platform way to install Linux on your Raspberry Pi is using the NOOBS installer. There is no amount of documentation I could add here which would be better than that which already exists, so please visit the Setting up your Raspberry Pi page at RaspberryPi.org.

1.2 Going Headless - No Monitor, Mouse or Keyboard Needed

Many folks running BrewPi Remix intend to embed the physical computer in their project and view the interface remotely on a web browser from another computer. This is a great way to create a nice clean setup, and it's pretty easy to do.

Full instructions are included in the article Headless Raspberry Pi on the BrewPi Remix website.

1.3 Using your System

From this point you will either execute the BrewPi Remix installer via the Terminal window within the Raspbian Desktop (a black icon with a '>_' in it), or via SSH if you are headless. Nearly all testing is done via SSH which should be the simplest of all scenarios.

SSH is disabled by default on the contemporary Raspbian distros. This is done because the Raspbian OS is shipped with a well-known password ('raspberry') and leaving it open via SSH is what we call A Bad Idea™. You will be presented with an opportunity to change the default password if it is set during the BrewPi installation process. SSH will be enabled if you follow the Headless article referenced above. If not, you can enable it with two simple commands:

```
sudo systemctl enable ssh
sudo systemctl start ssh
```

1.4 Your RPi's Name

In the past, it was very common to instruct people to give their Pi a static IP address so they could access it via their local network. This is still possible, but a rather dated way to approach it. You can still do this by editing the Interfaces configuration, but a better way is to simply use the name. There is a function called "mDNS" embedded in computer systems which allows this to happen.

Note that Microsoft appears to be lagging a bit in their mDNS support. There is a way to enable it for classic programs in Windows 10, but it's not very straightforward. By far the easiest way to do it is to install Bonjour from Apple. Apple uses it to detect printers and such on the network so it's proper name is "Bonjour Print Services for Windows." You can download it here, it's free, and a very lightweight install.

The stock Raspberry Pi starts out life named 'raspberrypi'. You would access it over the network using the special domain suffix '.local' so in your SSH software or your web browser you would the name 'raspberrypi.local'.

If you insist on having a static IP address, the configuration has moved to the file /etc/dhcpcd.conf. To create the static configuration, run:

```
sudo nano /etc/dhcpcd.conf
```

In this file, look for a block that has something like this:

```
# Example static IP configuration:
#interface eth0
#static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
#static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1
```

At the very least you will have to uncomment the `interface`, `static ip_address` and `static routers` lines by removing the '#' sign. A minimal configuration may look like this:

```
# Example static IP configuration:
interface eth0
static ip_address=192.168.0.10/24
#static ip6_address=fd51:42f8:caae:d92e::ff/64
static routers=192.168.0.1
#static domain_name_servers=192.168.0.1 8.8.8.8 fd51:42f8:caae:d92e::1
```

Finally, you will need to restart your network interfaces for these changes to occur:

```
sudo service networking restart
```

Because this is much less straightforward than it used to be, and assumes knowledge of what's called CIDR notation and other networking skills, I sincerely recommend NOT using a static address unless you really need it and/or know what you are doing. I'll answer questions as I can on the forum about this, but just keep in mind I've given you the very sincere recommendation NOT to do it.

1.5 Updating programs

Keep your Pi's programs up to date with these commands:

```
sudo apt-get update && sudo apt-get upgrade -y
```

1.6 Cleaning Local Repositories

Occasionally you may want to clean out old or unused repository files which take up space on your SD card. To do so, use the following commands:

```
sudo apt-get clean && sudo apt-get autoclean
```

1.7 Updating firmware

Make sure you also have the latest firmware version, and stay up to date using the rpi-update tool. Firmware updates will often fix instability issues, so make sure you keep this up to date. To run, execute the following command (it will reboot after completion):

```
sudo PRUNE_MODULES=1 RPI_REBOOT=1 rpi-update
```

1.8 Work Complete

If you have followed along, you now have a perfectly functioning, up to date, Raspberry Pi on your network, capable of running BrewPi Remix. Congratulations, this was probably the toughest part!

CHAPTER 2

Automated Installation of BrewPi

Part of the "remix" in the BrewPi Remix project was to create all new scripts and methods to install and update BrewPi. These scripts are part of the brewpi-tools-rmx repository on GitHub. The install script will install all dependencies, set up users and permissions, download the latest code base and setup a few system daemons to keep things running well.

Behind the scenes there are actually four GitHub repositories required to run BrewPi. None of that will be visible to most users. Most people will only need to follow the instructions they are presented to get a perfectly running system.

2.1 Running the Install Script

The initial install is handled by a bootstrap script, which will execute, provide for dependencies, and call the rest of the scripts needed. Behind the scenes; GitHub repositories will be cloned to your RPi, apt packages will be downloaded and installed, and configuration items created.

If you are security conscious, naturally inquisitive, or would just like to know what's going on, have a look at this note about security.

Use the following command to begin the installation, no other preparation or commands are necessary. The script is automatically downloaded for you:

```
curl -L install.brewpiremix.com | sudo bash
```

Now just follow the instructions on the screen.

2.2 After the Install

If the installation was successful, your last step will be to *to set up your devices in the device manager*.

CHAPTER 3

Configuring your Devices

Since BrewPi 0.2, the hardware setup is dynamic and flexible: you can install and uninstall hardware from the web interface. This is all done from the *Device Configuration* tab in the web interface. From the device manager you can assign hardware (temp sensors, SSRs, etc.) to functions.

3.1 Your Devices After Programming

If you have just uploaded a HEX file to your Arduino and the EEPROM was reset, no devices are installed by default.

3.2 Receiving the Device List

Even with no devices configured, you must first click *Refresh device list* to receive an updated list of installed and detected devices an slots from the Arduino. To be able to receive the device list, the BrewPi script has to be running.

3.2.1 Detected Devices

The detected devices list shows all devices that are automatically discovered by BrewPi, these include:

- All OneWire devices (temperature sensors and DS2413 OneWire switches)
- All pins for which there is an assigned terminal on your shield

3.2.2 Installed Devices

All devices that are assigned to a function are found under *Installed devices*.

3.2.3 Device Properties

Each device has the following properties:

Device setting/property	What it does
Device slot	A device is installed into a device slot. This is a unique number used to identify the device. When configuring your devices, make sure there are no 2 devices with the same slot.
Assigned to (chamber)	Each device is assigned to a Chamber. Currently there is only Chamber 1, but we are preparing for future multi-chamber support. Select *Chamber 1* for all your devices.
Assigned to (beer)	In each chamber, each device is either a *Chamber device* or assigned to a beer. Currently all supported devices are chamber devices, except for the beer temperature sensor which should be assigned to *Beer 1*.
Function	**This is the most important setting for your device. Here you can set wh** • **Chamber temp**: the sensor in the fridge (chamber device). • **Beer temp**: the sensor in your beer (beer device). • **Room temp**: measures any temperature you want, but is not used in the algorithm, just for logging (chamber device). • **Chamber cooler**: The output that controls your fridge compressor (chamber device). • **Chamber heater**: The output that controls your heater (chamber device). • **Chamber light**: This output is activated when the door is opened (see chamber door) and can also be used as heater by enabling *light as heater* in advanced settings. • **Chamber door**: an input that detects when the fridge door is opened.
Device type	Not user configurable, set automatically based on assigned function.
Hardware type	Not user configurable, set automatically (*Temp Sensor*, *Digital pin* or *OneWire actuator*).
Device type	Not user configurable, set automatically based on assigned function.
Pin type	Here you can set whether the output/input should be inverted. Because the shields have a transistor that inverts the signal behind each output with a terminal, you should set this to *inverted*. For devices you add yourself, without a transistor, do not invert the signal. For the door switch, it depends on the type of switch you have.
Arduino Pin	The pin the device is configured for. You can only set this yourself when defining a new device. The OneWire pins are hardcoded, you can not add your own OneWire pins. Just connect your sensors to an existing OneWire pin and refresh the device list to detect it.

3.3 Installing new devices and assigning them to a function

You can install a device by changing the properties to a correct configuration and hitting *Apply*. If the values are accepted by the Arduino, your device will show up under *Installed Devices* after refreshing the list.

Please refer to the screenshot below for a reference configuration with all devices installed currently supported by BrewPi. Just leave out any devices you don't have.

3.4 Uninstalling a device

To uninstall a device, just set it's function to *None*, hit *Apply* and refresh your device list.

CHAPTER 4

Specific Gravity Tracking

BrewPi Remix will incorporate readings from a gravity device, currently the Tilt, Tilt Pro or iSpindel. The Tilt uses Bluetooth Low Energy (BLE) and the iSpindel uses WiFi. Because of this difference, they will connect to BPR in different ways. You may have one or the other attached to a chanber, and only a single device.

Between teh two, I find the Tilt and Tilt Pro to be the most hassle-free choice, but they do come at a premium price. For me it's worth it, but you will have to make up your own mind.

4.1 Tilt

To configure the Tilt or Tilt Pro, you may choose the color during the install. Doing so puts a configuration line in the *config.cfg* designating this color such as:

```
tiltColor = Yellow
```

You may also choose to clamp the readings to prevent wide swings which can be created by moving or shaking the fermentor:

```
clampSGUpper = 1.175
clampSGLower = 0.970
```

If you failed to add the Tilt on the original install, or if you wish to change the color, edit the file and restart the script.

4.2 iSpindel

To add iSpindel, you must have the iSpindel configuration set in your config.cfg in the format:

```
tiltColor = iSpindel-Yellow
```

The name or color used is configured within the iSpindel configuration portal. Point your iSpindel to the brewpi-api.php file as its "HTTP" service type on port 80. More information may be found in the iSpindel documentation.

If you are using a multi-chamber configuration, point the iSpindel at the brewpi-api.php in the directory corresponding to the chamber you wish to serve.

As with the Tilt, you may choose to clamp the readings to prevent wide swings which can be created by moving or shaking the fermentor:

```
clampSGUpper = 1.175
clampSGLower = 0.970
```

If you failed to add the iSpindel on the original install, or if you wish to change the name, edit the file and restart the script.

CHAPTER 5

BrewPi Remix Manual Installation

Coming soon(ish.) For now you can view an article on the BrewPi Remix Website.

CHAPTER 6

Programming your Arduino

6.1 Programming via Command Line

Currently the web-based firmware update method is not very dependable. I sincerely recommend using `updateFirmware.py`, which is located in your BrewPi Utility directory. This method has the advantage of providing a menu to download the correct firmware package. When you install BrewPi Remix you will be presented with the opportunity to flash your Arduino. There are cases where you may want to re-flash, or change your currently connected Arduino.

Run it with:

```
sudo /home/brewpi/utils/updateFirmware.py
```

The script will pick the serial port currently configured for BrewPi. With a single chamber install, this defaults to 'auto' and will choose the first Arduino it finds if there are multiple controllers plugged in. In multi-chamber mode, the script will flash the Arduino corresponding to the local installation's configuration file.

The script will guide you through the process, presenting you with a choice of firmware versions which are available and appropriate for your shield type. Generally you will want to choose the latest version.

Finally, the script will offer to back up and restore your settings and configured devices if it was previously configured. CHoose according to your preferences.

When complete, the script will attempt to restore the state of your system; if it was running before you started it will attempt to restart BrewPi.

6.1.1 Selecting a Shield

If your Arduino has never been flashed before, you will first be prompted with the opportunity to select a shield type. The available types are:

- **RevA**: One of the original BrewPi Arduino shields. Very few of these are in the wild, and it is only supported through firmware version 0.2.10.

- **RevC**: The successor to RevA (no idea what happened to RevB), this is the most common type. It includes support for a parallel LCD screen display and a rotary encoder which can be used to make manual adjustments. Nearly all shields right now are RevC. *Even if you do not use a shield* you should select the RevC for most purposes.
- **I2C**: The newest shield variant which supports an I2C/TWI LCD display. Technically speaking this firmware version does not need a shield in order to allow complete functionality, unlike the parallel LCD versions which require some additional circuitry. If you are not going to use a shield at all however you want to use an I2C LCD you should choose this shield type. Be aware that this firmware shield version moves the OneWire sensor to A0 from A4.

For the most part, the shield type designation for the firmware is really about the capabilities of the firmware more than if you have a shield plugged in or not. Nearly all guides will default to the RevC pinouts.

If you have an Arduino that is currently flashed with one shield type and you wish to change, you can use the *-h* or *–shield* argument like so:

```
sudo /home/brewpi/utils/updateFirmware.py --shield
```

6.1.2 Beta Versions

If you would like to be presented with possible beta versions of firmware as a choice, use the *-b* or *–beta* argument like so:

```
sudo /home/brewpi/utils/updateFirmware.py --beta
```

6.2 Programming via Web Interface

Note: The web-based firmware flashing is not dependable in the current version and lacks some of the functionality of the command-line tool. It is recommended to avoid the web-based interface for flashing right now. This documentation will be updated when the web update functionality is restored.

6.3 Troubleshooting

- If saving devices in the device manager does not work, your EEPROM was probably not reset properly. Try reprogramming without settings and devices restore.
- Check whether it is not a hardware problem by programming your Arduino using the Arduino IDE. Just open the *blink* example and click *upload*.

CHAPTER 7

Running Multiple Chambers on the Same RaspberryPi

You may wish to run more than one chamber on a single Raspberry Pi. As of release 0.5.1 this is supported with the normal install tools.

You will need an Arduino per chamber, configured appropriately.

7.1 Clean Install Required

Installing a multi-chamber system is only possible on the initial install. If you have a current single-chamber setup you must remove it and reinstall from the beginning. If you need to uninstall you can either start over with a clean SD card and fresh install of Raspbian (preferred) or try the uninstaller:

```
curl -L uninstall.brewpiremix.com | sudo bash
```

Follow the instructions on the screen.

7.2 Multi-Chamber Installation

Starting with a clean system, run the standard installer:

```
curl -L install.brewpiremix.com | sudo bash
```

Now just follow the instructions on the screen. The important choice comes during this part of the install:

```
If you would like to use BrewPi in multi-chamber mode, or simply not use the
defaults for scripts and web pages, you may choose a name for sub directory and
devices now.  Any character entered that is not [a-z], [0-9], - or _ will be
converted to an underscore.  Alpha characters will be converted to lowercase.
Do not enter a full path, enter the name to be appended to the standard path.
```

(continues on next page)

(continued from previous page)

```
Enter device/directory name or hit enter to accept the defaults.
[<Enter> = Single chamber only]:
```

This is where you must enter a directory name to use, for instance "chamber1". No spaces or special characters may be used. After this entry you will be asked for a friendly name for your chamber to be displayed in menus and on the web page:

```
Now enter a friendly name to be used for the chamber as it will be displayed.
Capital letters may be used, however any character entered that is not [A-Z],
[a-z], [0-9], - or _ will be replaced with an underscore. Spaces are allowed.

[<Enter> = $CHAMBER]:
```

You may use spaces and capitalization here such as "Chamber 1" however special characters are not allowed.

7.3 Arduino Selection

During the install, if there is only one Arduino the script will create a udev rule for that controller. If more than one is present, a list of Arduino serial numbers connected will be displayed.

Keeping track of serial numbers may be somewhat unwieldy. To avoid this, plug a single new Arduino in for each install pass. An Arduino which has previously been configured for a different chamber will be ignored.

7.4 Adding Additional Chambers

For each additional chamber desired, simply plug in a new Arduino and re-run the install.sh script in the tools directory:

```
sudo ~/brewpi-tools-rmx/install.sh
```

7.5 Multi-Chamber Index

A link to a multi-chamber index will be created in the root of the web directory. The chambers will be in a sub-directory underneath the root. The multi-chamber index will allow a terse view of each configured chamber's status, and you can view the full chamber web page by clicking the corresponding LCD.

CHAPTER 8

Updating BrewPi

The script will stop the BrewPi script when it executes. To run the update script, use the following command:

```
sudo /home/brewpi/doUpdate.sh
```

The script will actually download the update script currently on the GitHub abd re-execute that in order to take advantage of any updates to the script itself.

If merging the updates fails, the script will ask you to stash your local changes. This commits your changes to the git stash and bring your repository back to its original state. You can get your changes back with 'git stash pop', but be warned that your changes could be incompatible with the latest updates.

8.1 Scripts After Update

After updating, the updater calls several other scripts from the Scripts area. These are:

Script	Function
doDepends.sh	**Checks for updates to the following:** • apt packages • pip packages **Removes the following if they are installed:** • PHP5 files • nginx files
doCleanup.sh	**Removes:** • *.pyc files • Empty directories
doDaemon.sh	Install and/or update the daemon unit files which run BrewPi and optionally the WiFi Checker
doPerms.sh	**Sets permissions on the following:** • Script directories • Web directories • Bluetooth stack

8.2 Updating Multi-Chamber

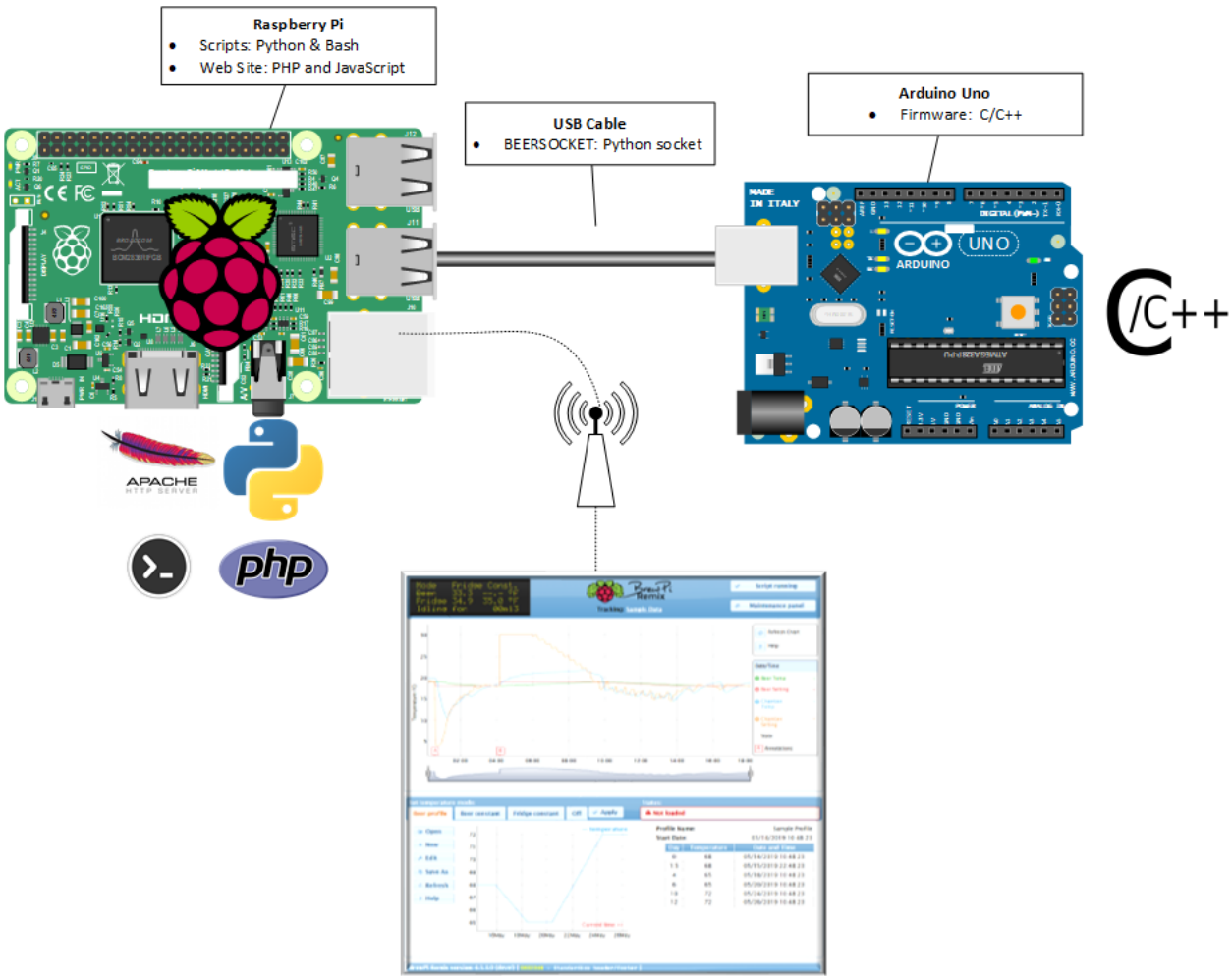
You will need to run the *doUpdate.sh* script from each chamber's instance in order to upgrade all of them.

CHAPTER 9

BrewPi Remix API

BrewPi Remix functions via interopeation of three different parts of the project:

- Firmware - This is the compiled application running on the controller
- Scrips - Python and shell scripts designed to receive and make available data from the controller, and send commands back to the controller
- Web Pages - A web application written in PHP and JavaScript



Communication takes place in the following manner:

- Firmware <-> Script: The Python Script on the Raspberry Pi opens serial communications over the USB cable with the Arduino Controller. Two way communication is possible via specially formatted JSON messages.
- Web Server <-> Script: A PHP socket called *BEERSOCKET* is created within the filesystem. This socket allows two-way communications between the web server and the Script via specially formatted messages.
- End User <-> BrewPi Remix - The end user opens a web page served by Apache 2 running on the Raspberry Pi. The web pages are written in PHP and JavaScript. The web server allows commands to be sent to the Script via the *BEERSOCKET* and the Script in turn will communicate with the Controller as needed. Data to display on the web page is sent to the web server throgh *BEERSOCKET* as well as filesystem information.

9.1 Arduino Commands

The following commands are issued by the Script to the Arduino Controller. If desired, these commands may be sent to the Controller through a terminal emulator.

- A - Alarm on
- a - Alarm off
- t - Request temperatures

- C - Set default constants
- S - Set default settings
- s - Request control settings
- c - Request control constants
- v - Request control variables
- n - Request firmware version information
- l - Request LCD contents
- j - Set settings as JSON
- e - Request contents of EEPROM
- E - Initialize EEPROM
- d - Request devices in EEPROM order
- U - Update device
- h - Request hardware
- Z - Zap EEPROM (in DEBUG mode only)
- R - Reset controller

9.2 Arduino Message Types

The following characters define the JSON message type sent to and returned by the Controller. They are hte first character of the JSON message.

- T - Temperature info
- D - Debug message
- L - LCD content
- C - Control constants
- S - Control settings
- V - Control variables
- N - Version number
- h - Available devices
- d - Installed devices
- U - Device update confirmation

9.3 BEERSOCKET Commands

The following messages are sent through the *BEERSOCKET* from the web server to the Script. These may also be send manually using the *{web_root}sockettest.php* page.

- ack - Acknowledge request (test message)
- lcd - Request LCD contents from script

- getMode - Request mode setting from script
- getFridge - Request fridge temperature setting from script
- getBeer - Request beer temperature setting from script
- getControlConstants - Request control constants from script
- getControlSettings - Request control settings from script
- getControlVariables - Request control variables from script
- refreshControlConstants - Request control constants from controller
- refreshControlSettings - Request control settings from controller
- refreshControlVariables - Request control variables from controller
- loadDefaultControlSettings - Reset control settings to default
- loadDefaultControlConstants - Reset control constants to default
- setBeer - Set new beer constant temperature
- setFridge - Set new fridge constant temperature
- setOff - Set mode to OFF
- setParameters -
- stopScript - Stop script, write semaphore
- quit - Quit but do not write semaphore
- eraseLogs - Erase stderr and stdout logs
- interval - Set new logging interval
- startNewBrew - Set new beer name
- pauseLogging - Pause logging, may be resumed
- stopLogging - Stop logging, may not resume
- resumeLogging - Resume logging
- dateTimeFormatDisplay - Change date time format
- setActiveProfile - Set a new beer profile
- programController or programArduino - Reprogram controller
- refreshDeviceList - Request devices from controller
- getDeviceList - Request device list from script
- applyDevice - Create device settings
- writeDevice - Configure a device
- getVersion - Get firmware version from controller
- resetController - Erase EEPROM

9.4 Config.cfg Settings

These settings control how the application behaves. They are set in the *{app_home}/settings/config.cfg* file. Not all are mandatory.

- altport - Checks this port definition if the controller is not found on "port"
- beerName - Name of the beer currently being logged
- dataLogging - Defined whether data loogging is active, paused, or stopped
- interval - Time period between data points
- iSpindel - Defines use of iSpindel (not currently in use)
- logJson - Log every received line from the controller if True, False only logs 'New JSON received.', when not defined, JSON messages are muted
- port - Port at which the script will communicate with the controller. If 'auto' it will connect to the first controller found on the USB bus. May also be explicit such as /dev/ttyACM0 or /dev/chamber1
- scriptPath - Path where the brewpi.py script may be found
- tiltColor - Color of currently connected Tilt
- wwwPath - Path to current chamber's website
- useInetSocket - Windows only, set to true to allow Inet socket use
- socketPort - Windows only, set to port for socket (default 6332)
- socketHost - Windows only, set to IP address for socket (default localhost)
- arduinoHome = Set to location of arduinoHome, defaults to /usr/share/arduino
- avrdudeHome = Set to location of avrdude, defaults to arduinoHome/hardware/tools/
- avrsizeHome = Set to location of avrsize, defaults to empty string because avrsize is in path on Linux
- avrConf = Set to location of avrdude.conf, defaults to avrdudeHome/avrdude.conf
- boardType = Defaults to 'arduino', no longer used

CHAPTER 10

Calibrating your Probes

The DS18B20 probes are factory calibrated and are ±0.5°C accurate. If they are within factory specifications, the maximum difference between two sensors can be 1°C (1.8°F.) If you are within this range, no calibration is necessary. The temperatures should be consistent, not exact.

If you are convinced you need to calibrate your sensors, you will need to use direct serial commands.

10.1 Calibration Steps:

- Connect the sensor to calibrate
- Use `h{v:1}` to list the hardware with values and verify the Arduino can see the sensor. (It's easiest if just one temp sensor is connected.)
- Place the sensor and another calibrated thermometer in a large glass of water. Leave for 1 minute to settle.
- Take three readings from each sensor at 30s intervals.
- Average the three readings from each sensor
- The calibration adjustment value is the average value of the calibrated thermometer minus the sensor's average value.

Note:

- The `h{v:1}` command always prints uncalibrated, raw values from the sensor.
- The `d{v:1}` command always prints values from the sensor that include calibration adjustment.

10.2 Setting Calibration Adjustment

The `U` command has the `j` parameter for setting the calibration adjustment on temp sensors `{h:2}`. Valid values are in the range +/- 7.9°C. (You will always calibrate in degrees C.)

For example, after defining a temp sensor at index 3, you could set its calibration adjustment value like this:

```
U{i:3,j:-0.3}
```

That command will subtract 0.3°C from the sensors readings if the sensor reads higher than a calibrated thermometer.

10.3 Alternative method - relative calibration

Using all available sensors, place them all in a large body of water. From all sensor readings, determine the most likely temperature. Then calibrate all sensors to display this temperature.

While it may not be perfectly accurate, it will be close enough; it will be consistent, so all sensors report the same value given the same temperature.

10.4 Accuracy

Calibration adjustment values are stored to the nearest 1/16th of a degree to match the precision of the DS18B20s.

CHAPTER 11

How to Compile the Code from Source

These notes are just that; notes. It will be enough for a reasonably technical person to create the environment in which they can fiddle to their heart's content. If however you are here as a complete neophyte and have dreams of conquering your Arduino; this is not the right project for you.

Elco and others created this project many years back and by Elco's admission he would (and has) do it differently if he had to do it again. The project team also crammed about as much as possible into the relatively tiny Arduino Uno. By way of providing scale: When I added the I2C code, even though it was a compile-time choice (therefore not adding unnecessary code), I ended up searching through the code to find 26 letters in the text strings to remove to make room enough that it would not crash. It is that tight.

The original project was created in Atmel studio. That sent more than a few otherwise sane people screaming and looking for the exits. At the recommendation of another Brewing-related developer, I moved the project to PlatformIO and have not looked back.

You are free to do you - but I can't help you if you diverge from this path. And that brings up another sensitive point:

11.1 Disclaimer and Limitation

I will likely not provide support for anyone related to the source code. It's just too big a mess, too many choices to make, too many ways it can go wrong. If there's a one-liner question you want to ask in the forums I might be able to help (or someone else might) but there is no way I can retain my sanity if I try to support people going feral in the source code.

11.2 Setting up the Development Environment

I will make heavy use of existing documentation. There's no sense in me trying to re-write someone else's work. If you have an issue with a particular step, your best bet is usually seeking help with that tool's vendor or support methods specifically. If you need more help and see a link, click it and go get help there.

1. Install Microsoft's Visual Studio Code (hereafter "VS Code".)

2. Install PlatformIO as an extension within VS Code.

At this point, you have a very serviceable development environment. I encourage you to view the various documentation. A good place to start is PlatformIO's tutorial on building a cross-platform "Blink" application.

11.3 Clone the Repository

One need not use any Git functionality, however, I will use that functionality here as an example of how to obtain the source code. You may use your method; of course, you are on your own down those other paths.

Git is available and sometimes built into OS distributions. For Windows, I recommend downloading the Git for Windows available directly from the Git website.

Once that is installed, open Git Bash and navigate to your intended source code destination. i.e. If you want it in a subdirectory if C:\MyCode, open that folder in Windows Explorer, right-click on "MyCode" and select "Git Bash Here" from the context menu.

At this point you can clone any software into its directory with the simple command:

```
git clone {url to git repository}
```

For BrewPi Remix, the full command is:

```
git clone https://github.com/brewpi-remix/brewpi-firmware-rmx.git
```

The entire repository will be cloned to a directory named "brewpi-firmware-rmx."

11.4 Compile the Source Code

Open VS Code and PlatformIO will load. Use the File > Open Folder workflow to open the root folder of the repository (i.e. C:\MyCode\brewpi-firmware-rmx).

If you have followed along correctly and used the "Blink" example above, you now know how to compile the firmware and upload it to your controller.

11.4.1 Project Configuration

The BrewPi firmware's compile-time options are controlled by the `Config.h` file. Most importantly, the choice of the shield is set with the `BREWPI_STATIC_CONFIG` item.

I recommend you take some time reviewing this file and the code instances to understand the options available.

Looking through that file you will also see `VERSION_STRING` and `BUILD_NAME`. These macros, coupled with the shield type set with `BREWPI_STATIC_CONFIG` above, are responsible for some of the in-application information.

There are also two Python scripts, `git_rev.py` and `name_firmware.py`. These scripts set up the compilation environment variables which feed the above macros, as well as define the naming of the firmware file when created.

11.4.2 Other Environments

PlatformIO will also run within the Atom environment for its IDE, and it is capable of operating as a stand-alone CLI environment within Linux. I will leave that as an adventure for the reader.