piWebCAT

website: http://piwebcat.g3vpx.net

email: piwebcat@g3vpx.net

support group: piwebcat at groups.io

Fig 1 piWebCAT control window configured for Yaesu FTdx101D with the Hamlib API.

Introduction

piWebCAT is a web based rig control and station logging system.
The webserver is a Raspberry Pi 4 computer which connects to the rig via USB, RS232 or Icom CI-V.
Control is via wifi, wired LAN or internet using a web browser on a desktop, tablet or even by mobile phone.

The system is highly configurable. Controls have user specified features. For example: Button controls include position, colour, caption, grouping, behaviour and rig control codes. Sliders display an adjacent text value with user defined scaling, units and optional lookup tables. Meters have optional 20 point calibration tables with linear interpolation.
System configuration is facilitated by prebuilt configuration examples and extensive instructions in the manual.

A preconfigured Raspberry Pi micro SD card is available for purchase at cost price.

The initial development was ASCII text based control on a Yaesu FTdx101D. This is applicable to text based control systems on modern Yaesu and Kenwood rigs.
Icom CI-V was then added and then the older Yaesu 5-byte system (eg: FT920)
Then, in mid 2020, I discovered the Hamlib API which supports 250 radios. piWebCAT now has a fully featured Hamlib rig control system. Thus piWebCAT has four control modes: ASCII, YAESU5, CIV and HAMLIB.

I can quickly switch piWebCAT control of my FTdx101D between a direct ASCII configuration or a HAMLIB configuration using rig #1040 (the Hamlib code for the FTdx101D).

Configuration – MySQL database

piWebCAT configuration FTdx101D | Control | Cat config. | Meter cal. | radio | FTdx101D

Select table for edit: buttons | catcodes | sliders | lookups | meter | timings | radios | settings

buttons definition data . . . FTdx101D

Id	rig	description	color	caption	btnno	active	code	action	von	voff	numchar	nset	nans	chset	chans
11	FTdx101D	Select 10m band	indigo	10m	11	Y	BAND	G	0	0	N	9	77		
12	FTdx101D	Select 6m band	indigo	6m	12	Y	BAND	G	0	0	N	10	77		
13	FTdx101D	Select 4m band	indigo	4m	13	N			0	0	N	0	0		
14	FTdx101D	Select 2m band	indigo	2m	14	N		G	0	0	N	0	0		
15	FTdx101D	Seelect 70cm band	indigo	70cm	15	N		G	0	0	N	0	0		
17	FTdx101D	VFO A operation	maroon	VFOA	16	Y	VFO	G	0	0	N	0	0		
18	FTdx101D	VFO B operation	maroon	VFOB	17	Y	VFO	G	0	0	N	1	1		
23	FTdx101D	Roofer wide	teal	roof 12k	31	Y	ROOF	G	0	0	C	1	1	1	6
24	FTdx101D	Roofer SSB	teal	roof 3k	32	Y	ROOF	G	0	0	C	2	2	2	7
25	FTdx101D	Roofer CW	teal	roof 600	33	Y	ROOF	G	0	0	C	3	3	4	9
26	FTdx101D	Roofer narrow	teal	roof 300	34	Y	ROOF	G	0	0	C	4	4	5	A
27	FTdx101D	MOX	#C00000	MOX	59	Y	MOX	T	1	0	N	0	0		
28	FTdx101D	Tune on/off	indigo	Tune	75	Y	TUNE	T	2	2	N	0	0		
29	FTdx101D	Tuner on/off	indigo	Tuner	76	Y	TUNR	T	1	0	N	0	0		
32	FTdx101D	Copy VFO A > B	#003000	A > B	80	Y	ATOB	S	0	0	N	0	0		
33	FTdx101D	Copy VFO B > A	#003000	B > A	81	Y	BTOA	S	0	0	N	0	0		
34	FTdx101D	Swap VFOs	#003000	Swap A/B	79	Y	SWAP	S	0	0	N	0	0		
35	FTdx101D	Split VFOs	#003000	Split	82	Y	SPLT	T	1	0	N	0	0		

Fig 2 Fdx101D - buttons editing

piWebCAT is very highly configurable. The configurations are stored in a MySQL database tables in the RPi server.
Configurations are provided for the FTdx101D and IC7000 (both direct and Hamlib) and for the FT847 and FT920.
In addition, there are two set of learning 'rigs' , each having three versions with progressively added features. The website/manual uses these as part of a teaching exercise in how to configure piWebCAT for your rig.

A spreadsheet-like grid based configuration editor is provided with direct editing onto the grid. See Fig 2.
This displays configuration data only for the currently selected rig.
Where appropriate, data values are selected by simply clicking a grid cell and selecting from the resulting drop list.

The MySQL configuration database is also accessible using MySQL Front, HeidiSQL or other SQL editing applications.
I provide a suite of SQL scripts with functions which include cloning and renaming of rig configurations.
The use of MySQL Front or HeidiSQL and these scripts provides for total or partial configuration backup and potential sharing of configurations between users who have the same rig.

Project development – CAT control traffic issues

Rig control from a web browser has a more complex and potentially slower data path when compared to direct control from a PC via USB or serial. The two most critical areas were: (1) provision of a good S-meter presentation, and (2) an adequate tuning rate. Both need adequately fast data transfer repetition rates between the client and the radio. All data transfers are initiated from the web browser. The data path is:

Browser - javascript code > LAN transfer > RPi server PHP code > USB (or serial) > rig.
(Followed where necessary by data return on the reverse path.)

The two-way transaction time using Hamlib is in fact only 20 or 25 mS which offers potentially good data rates. However, piWebCAT has a number of concurrent processes which are all competing for data bandwidth.
The final very adequate result was achieved by the use of a carefully designed prioritised queuing system and user definable repetition rates for each type of data.

- Tuning action from piWebCAT needs a fast data rate to giving smooth useable tuning action.
- Tuning action from the rig needs to update the piWebCAT frequency display but this can be at a much slower rate.
- S meter (and Tx meter) reads need a fast rate to give a good meter display. I actually display meter rate on the web page footer as a performance indicator. A rate of 530/min is typical using my PC and an RPi 4.
  The rate slows after an hour or so due browser memory leakage and other well documented issues. (piWebCAT is firing 20 URLs / second at the server … most browsers send nothing until you hit a key!) The reset button restores full speed without any operational change to the rig)
- Slider and button controls are all synchronised at start up to their corresponding rig values. This may take four seconds. The same happens when changing band. The controls are unresponsive during this four second interval. However, piWebCAT remembers the settings per VFO and so the four second delay is not encountered on VFO swapping.
  Each button and slider control sends a single command to the rig when operated on piWebCAT. These actions have minimal data bandwidth demand.
  Of more significance is the updating these controls in piWebCAT in response to changes on the rig.
  There are 90 buttons and 27 sliders. To update all 117 controls once per second would not be possible.
  Each control therefore has a user definable option to be synchronised to the rig (by setting **active = S**).
  The web client builds a list of synchronised controls at startup and then reads them cyclically. The list stepping rate is user definable. Thus the user can selectively achieve the syncing of important controls in a manner which does not compromise the fast S meter and tuning needs.
- The MOX button is handled separately. piWebCAT needs to respond rapidly to PTT Rx/Tx switching.
  Here again, the rate is user definable.

Tuning

Supported bands are: 160m, 80m, 60m, 40m, 30m, 20m, 17m, 15m, 12m, 10m, 6m, 4m, 2m and 70cm.

Tuning is by mouse drag or mouse thumbwheel rotation on the blue horizontal tuning band as shown in Fig 1.
The tuning band has three lanes: slow, medium and fast. The tuning rates for each lane are user configurable in the MySQL database. (ie: Hz/pixel for mouse drag and Hz/click for thumbwheel)
On a touch screen tablet or phone, horizontal finger drag can be used.
(Finger dragging usually moves the whole window – but this is suppressed on tuning lanes and on slider controls)

Beneath the tuning lanes is a frequency scale. Each of the 14 bands has a scale designed to fit the available width.
The scale has a red marker which moves in response to frequency changes made on piWebCAT or on the rig.
Course frequency changes can be made by simply clicking at a point on the scale.

piWebCAT provides tuning by UP and DOWN buttons for VHF/UHF channels, eg: +12.5kHz and -12.5kHz
This is one of a small number of fixed coded functions
- but you still retain control of frequency shift, button choice and captions and colour.

Memories

Rig channel memories are configured on the rig.
They can then be selected for use by piWebCAT.

Any button of your choice can be configured to select a specific memory channel.

In addition, a button can be configured with code=MPAD to popup a memory selector numeric keypad to select memories by number.
The keypad can be dragged to a convenient position within the browser window.

On clicking OK, a command is sent to select the memory.

Fig3 memory keypad

Button controls

The control window has **66 button controls.** Their positions and sizes are fixed. See Fig 1.
24 additional buttons are optionally available on a moveable popup window.

A few of the control window buttons are of fixed function (eg: VFO switching, MOX).
The rest have their function totally defined in the database.
Most users will not wish to move the band and mode buttons on the right of the display but it can be done.
Unused band buttons (eg: 4m, 2m and 70cm) can be allocated to other functions.

All buttons have configurable captions and colour. Unused buttons are set inactive and show a light grey.

Buttons have an **action** field option which includes:

- **S** Single action
- **T** Toggled (eg: on/off)
- **G** Grouped – participating in a mutually exclusive group with other buttons with the same **code** field.

Any button can be configured as a **MORE** button which launches the popup window with **24 extra buttons**, all similarly configurable.

Any button can programmed as a **MPAD** button which launches a memory numeric keypad.

The **active** field in the button configuration tables has options:

- **N** The button is disabled and is shown only in a pale grey.
- **Y** The button is active and controlling the rig.
- **S** The button is active, controlling the rig and synced to the rig.
  This means that the button's state will follow changes on the rig as described above.
- **L** LED button. The button does not control the rig. It is simply an indicator of the corresponding control on the rig. The button is black with red text. The configured colour is a light colour that shows the ON state.

Slider controls.

The control window has **27 slider controls**.
Some are predefined and some are spare for user allocation ... but you can redefine them all.
The slider configuration defines its function (the rig control codes) and matches the range of the numeric data returned from the rig to the full range of slider movement.
Each slider has an adjacent value display (with scaling, decimal point, units and optional lookup table).

Fig 4 slider controls

Sliders are in three groups:

- COMP, Mic Gain, Vox gain, RFpower, AF gain and squelch (probably unchanged) + 2 spare.
- Nine sliders with a reset to default button. The default value is defined by you in the database.
- Nine sliders with an adjacent button: mainly on/off.
  The association of button and slider is database defined and optional.

Using the *Firefox* web browsers, sliders can be slowly moved with the mouse wheel (allowing effective RIT tuning).

As with button controls, sliders can optionally have **active = S** so that they remain synced to their rig control.

Default settings of controls and maintaining synchronisation with the rig.

At start up, frequency and mode and settings for buttons and sliders are loaded from the rig.
Thereafter, frequency, band, mode and MOX update on piWebCAT, if changed on the radio.

In addition, every button and slider control has the option to set **active=S** (S = sync instead of just Y = yes).
All such **sync** controls are then continually polled in a circular list and updated from their current rig setting.
Sync status is thus restricted by user choice to selected controls to minimise the data bandwidth.
The polling interval for the list is user configurable.
Any such control's update interval = polling interval / total no of sync controls.

A 'Reconnect' button is provided to reload the ALL the settings.
However, this is only needed if changes are made on the radio:
If you use piWebCAT to change band or VFO on the radio, the radio responds immediately and piWebCAT's controls are updated automatically within 5 seconds.

VFO change facilities

piWebCAT remembers the most recent settings for each VFO and so the 5 second loading of the settings does not need to occur on repeated swapping between VFO A and VFO B.

There is an option on VFO swapping to automatically mute the background receiver and switch the foreground receiver to its last audio level. This facility is for dual receiver rigs such as the FTdx101D which have two separate audio gain controls which are not automatically adjusted on VFO change.

Metering

Fig 5 S meter

On receive, the meter is an S meter with an appearance similar to the FTdx101D.
On transmit, the meter displays one of five button selectable options.
The Tx metering options and their CAT controls are defined in the database for your radio.
I provide five Tx meter backgrounds for Power, ALC, Compression, PA ID and SWR.
These are simple 250 x110 bitmap images ... you can change them (please keep the arc !!)
Accuracy of display is achieved for each of the six installed metering functions by the provision of optional 20 point interpolated calibration tables in the database (and an easy calibration procedure)

Database editing.

piWebCAT has a spreadsheet-like editor grid for each of the database tables. See Fig 2
Editing is directly onto the grid. Only records for the selected radio are presented.
The grids are produced using phpGrid for which I have purchased an 'Enterprise' license which allows allows OEM distribution of the code. (It is of course illegal for users to use the code for other projects.)
phpGrid makes extensive use of jQuery and jqGrid which are free to use open source products.

The database tables can be exported to a CSV file for import into Excel. They can then be printed.

The database is standard MySQL (MariaDB) and so can be edited by other database editor/toolkits.
For most of the development, I used the free MySQL Front. This can export radio configurations and so facilitates the exchange of configuration data between users. (See section 14.1 Downloads)
More recently, I have used HeidiSQL: another excellent, free database editor / toolkit.

Rig duplication - SQL scripts

I provide SQL scripts which include duplicating a whole rig configuration (with a different name!).
This allows you to preserve my existing configurations whilst experimenting on a copy.

Using the Hamlib rigctld API:

Each radio has a Hamlib identifying number.
Typing **rigctl -ls** in the Raspberry Pi terminal window will list all the supported radios with their Hamlib number.
You create an entry for your rig in the **rigs** table and set: **catcomms=HAMLIB** and **hamlib=nnnn** (rig number)

Hamlib's **rigctld** translates a common set of commands into CAT commands for your selected radio.
eg: If FTdx101D (#10400) is selected, then Hamlib: *set_freq VFOA 3744000* will translate to Yaesu: *FA003744000;*

An important early configuration decision with Hamlib is whether or not to use **–vfo mode**.
--vfo mode allows separate control of the rig's two VFOs (or two completely separate receivers, eg: FTdx101D)
If –vfo mode is used, then ALL Hamlib commands must contain the VFO parameter (eg: VFOA), even if only as a dummy and not relevant to the command. (eg: \set_level VFOA RFPOWER 0.6)

piWebCAT can display the frequency of both VFOs, one as current and one as background. However, it has only one set of slider and button controls and so, where relevant, the commands from the control to the server must be automatically directed to the current VFO/receiver.
In piWebCAT, this is done by the control being configured with **vx=V** in the client (web browser) data.
V translates to current VFO selection of **A** or **B** in the client > server message. This directs the message to one of two **A** and **B** server CAT command records identified by **abx=A** or **abx=B**.
For commands which are not VFO specific, **vx=X** and the single corresponding server record has **abx=X**.

I have not used –vfo mode with Icom rigs as they do not appear to have access to the background VFO.

An early decision is not needed when using this **vx** and **abx** system with piWebCAT's non-Hamlib configurations.
The provision of separate A and B server CAT control records can be made on an individual control basis.

Amateur radio logging system For auto-time entry this needs internet connection or an RPi real time clock (£5)

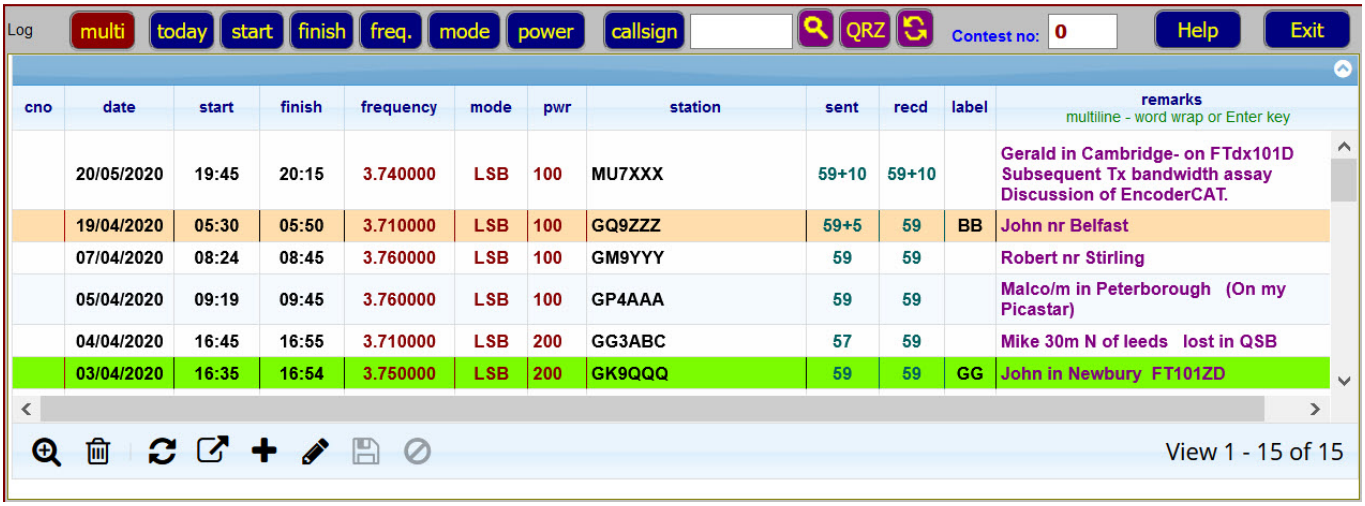

Fig 6 piWebCAT's log window

The log is stored in a table in the RPi MySQL database. The logging system has the following features:

- One single button click enters: date, start time, frequency, mode, power and optional contest number.
- Date picker and time sliders are provided for 'manual' data entry.
- Optional multiple lines of text with word wrap in the remarks column (automatically expands the row).
- Searchable label column with three specific label options to highlight the row in a background colour.
- Search button.
- QRZ.com button for the displayed callsign.
- Export to CSV button (will launch in Excel for printing etc)
- Backup and restore of log using third party MySQL editor (eg: MySQL Front ...free download)
- Logging system can be run separate from piWebCAT (No auto-insertion of freq, mode nor power)

Software

piWebCAT uses a 16 Gbyte microSD card

I can supply this ready configured.
The website / PDF manual gives detailed instructions on how to configure your own card.

The card is configured with:

- Raspberry Pi OS,
- Apache2 web server,
- MariaDB database server,
- PHP 7 server code,
- Pure-FTPd server for FTP file transfer - This is accessed from a PC by an FTP client (eg: FileZilla) and also by Microsoft Expression 4 web development system (or alternative).
- phpMyAdmin database tools.
- RealVNC server for keyboard + mouse + monitor control from a PC.
- Hamlib API which supports 250 radios in piWebCAT's HAMLIB mode.
- piWebCAT websever code (Located in the standard folder: /var/www/html)
- Ten preconfigured rig configurations including two sets of three progressive learning configurations.
- A copy of piWebCAT website as built in Help.
- Other standard Raspberry PI applications + Libre Office.
- Mumble VOIP server and client configured for Rx and Tx audio (using a USB audio adapter from piHut)

The piWebCAT 'website' code uses PHP server code and client javascript..

Client javascript makes extensive use of jQuery (open source).
Communication with the server is by jQuery Ajax commands.
The editing grids use phpGrid - a purchased product that is licensed to me for OEM distribution.
phpGrid makes extensive use of javascript jqGrid which is open source.

piWebCAT code was built using Microsoft's Expression 4 - a free downloadable web development package.
The design was done 100% in code. This kind of web page is too complex for WYSIWYG graphical layout.

The piWebCAT data path is:

Javascript in the web browser < LAN > PHP code in RPi webserver < serial OR USB > rig.

Documentation

The website is http://.piwebcat.g3vpx.net
The same website is on the microSD card with access from piWebCAT's Help button.
It was generated using HelpNDoc.
HelpNDoc also generates a printable PDF with no extra effort other than positioning page breaks:
http://piwebcat.g3vpx.net/files/piWebCAT.pdf

Hardware

For USB connectivity, only the Raspberry Pi 4 computer + microSD card is needed.

A Serial Pi Zero GPIO module can provide RS232 connection.

My G3VPX piWebCAT PCB has RS232 and Icom CI-V connectors.
I can supply as a bare board or as a kit of parts. Contact piwebcat@g3vpx.net

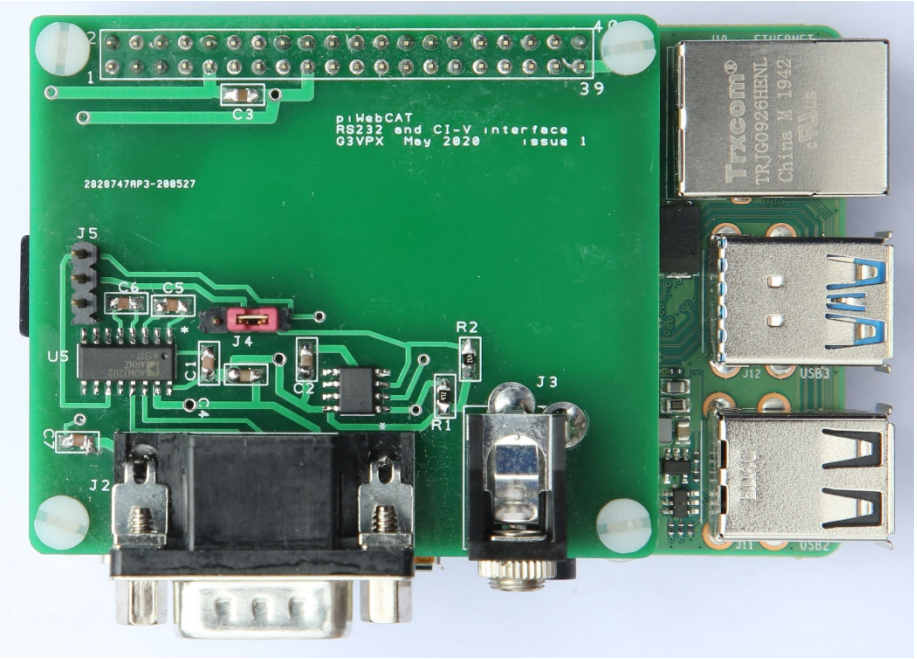

Fig 7 piWebCAT PCB on a Raspberry Pi 4

Mumble

Mumble is a free, open source, low latency, high quality voice chat application using Voice Over IP (VOIP).
It is not part of piWebCAT. Its installation adds the transmission of Tx and Rx audio to and from the RPi.
This allows operation from a location remote from the transceiver using the combination of CAT control and audio.

It is one of a number of VOIP applications that could operate in parallel with piWebCAT.
It appeared to be the most suitable package for this purpose.

I have used it in QSOs with piWebCAT using my laptop and a USB microphone.

My preconfigured microSD card has mumble client and server configured and working with a Raspberry Pi USB audio adapter purchased from piHut for 5ukp.

The website /PDF manual gives detailed setup instructions for Mumble.

Fig 8 USB audio adapter