remotesDB *concepts 1*

This is my tentative definition of a 'good' Data Base for IRP storage, to work with `irp_classes` and to be a start point to build 'good' user-oriented IR home automation applications.
I think to specialized applications, with advanced user interfaces, like a media center: you can choose the TV channels by channel icon, or to schedule air conditioner on week basis etc., beyond the simple replication of strandard IR remote controls (as in this demo). All this using smart-phone on WIFI or Internet.
*As a first try of **remotesDB** capabilities, here a demo application to perform a simple replica of standard IR remote controls.*
*You can see it as an example of using **remotesDB** and the included php libraries you can use in your IR applications.*

`remotesDB:`

- *Stores IRP and commands in four formats: 'HEX' (0x003E20), protocol and device 'DATA SET' ('{F=2}') and RAW-1.*
- *Because it stores RAW commands, you can implement 'learn and repeat' strategies in case of unknown IRP.*
- *Can handle IR 'universal' controls, using idRemote + 'code' as PK.*
- *Can handle real or virtual remote controls, status less or with status.*
- *Can handle static and dynamic keys.*
- *Stores the commands that a device can receive.*
- *Stores UI oriented informations, like remote controls layouts, key translations, tooltips.*
- *Stores also generic reference data, like photos and documentation or external urls.*
- *Uses views and stored procedures to make it easy to write custom applications.*

IR remote controls

modal multifunction keys

IR remote controls can have some multi-function keys (the key 'UP' is also 'PLAY'), and some keys are for internal use only (the key 'SET'). And a device can receive only a sub-set of keys of a remote control (e.g. a TVset, and a remote control for TV + DVD combos).
How to define such key? KEY_UP or KEY_PLAY? How to handle it?
The solution of all this problems is to explicit the *modal* behavior.
In one 'mode' a key has a name and triggers an action, in a different 'mode' the same key has a different name and triggers a different action maybe using a different IR code.

Implementation

In `irp_remotes.modes` you set all modes for a remote control like this:
`T=TV|X=TXT|V=VCR|S=self`
In `irp_remkeys` you creates records for all required keys, with key position and mode (the definition can be done also importing a remote sheet, see "Update remoteDB using lirc or sheet files" tool):

```
-----NAME--------------------------ROW-----COL-----MODE--
    KEY_UP                      5      2      T,X
    KEY_PLAY                    5      2      V
    KEY_LEFT                    6      1      T,X
    KEY_BACK                    6      1      V
```

In `irp_devrem,` where you associate remote controls and devices, you can specify in `irp_devrem.mode1, irp_devrem.mode2` and `irp_devrem.mode3` up to three modes (key sets) that the device can accept.
note: default `A, A=all`

Example: my TV HITACHI CL32WF740AN is now being used with a external DVB-S, so all TELETEXT commands are not useful with this TV set.
I just set `irp_devrem.mode1 = 'T'` (TV).

Now in remoteDB you have the info structure to handle all situations easily. You can create a *modal* view of a (real or virtual) remote control that only enables keys required by the target device.

status and stateless remote controls

Many remote controls are *stateless*, meaning that by pressing a key the code sent to a device is always the same, other Remote Controls have a *status*, and the keys modifies the status and/or sends the status (total or partial) to the device.
So we must:

1. show the status to user
2. make the status permanent
3. associate status modifications to keys
4. send the status

implementation

1. The field `irp_remote.phpgui` can contain the name of an include php file (in `/extra/` dir) that build the HTML code required for a custom status UI on the standard simulated remote control, that is built in this demo by `usr_simremote.php`.
   This UI for aircon example is built by `./extra/aircon_std.`
2. A special fake key (in `irp_remkeys`) is used as storage space.
   The key is unique to *remote+code+device* and has `row=0` and `col=0`.
3. The code used to change the status is in 'onClick' (or others) key attributes.
   The HTML+javascript snippet is defined in the `irp_remkey.clickAction` field.
   These keys are defined *dynamic keys*, as opposed to standard *static keys*. A dynamic key is without associated IR code. (the definition can also be made importing a *remote sheet*, see the "Update remoteDB using lirc or sheet files" tool).

```
KEY_POWER2     1 1 F   onClick='sendOnOff($url);'
KEY_UP         1 2 F   onClick='document.getElementById("tempplus").click();'
KEY_MODE       1 3 F   onClick='nextValue("amode");'
```

4. The code (javascripr) required to perform some complex actions (e.g. *nextValue()* or *sendOnOff($url)*) and to save, restore and send the status is in a custom include php file in `./extra/` dir, defined in `irp_protocols.phpadapter`, on protocol basis.
   This example uses `./extra/Fujitsu_Aircon_adapter.php`

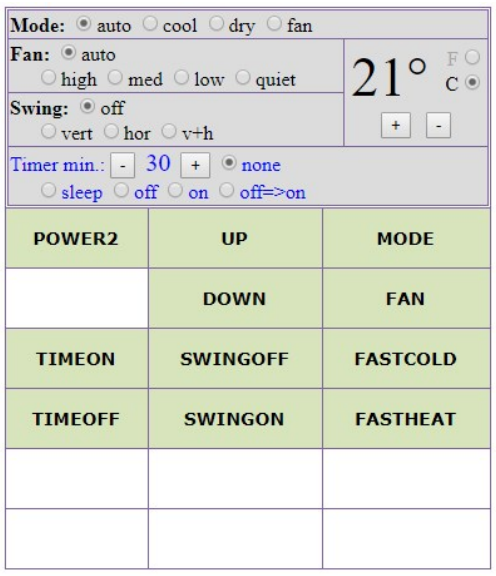
For the *logical schema* of remoteDB see `remoteDB-concepts2.pdf.`

multi-code IR remote controls

Some remote controls can send more than one single protocol. *Example: the remote control for my DVB-S top box (i-CAN 11105V) can also send IR commands to the TV set.*
There are also 'universal' multi-code remotes controls, with hundreds or thousands of codes. The control and the key set are always the same, but the key-IRstream association changes.

implementation

based on `irp_devrem.code` (association remote-device) and `irp_remcommand.code` (association remkey-stream).
note: default `code` is '0', but the numeric codes must start with an alpha character: not '0032' but 'K0032' (or whatever letter you like). You can also use names.



Mode: [✓ auto] cool dry fan
Fan: [✓ auto] high med low quiet
Swing: [✓ off] vert hor v+h
21° F [✓ C] + -
Timer min.: - 30 + [✓ none] sleep off on off=>on

POWER2	UP	MODE
	DOWN	FAN
TIMEON	SWINGOFF	FASTCOLD
TIMEOFF	SWINGON	FASTHEAT