First thoughts on the order of development (March 27, 2015):

1. Get OS loaded on SD card and up and running (DONE. RASPBIAN OS on a 32 GB micro SD card, WIFI working OK)
2. Get SSH running (just to make development easier).
3. Write demo program to read button presses. (simple use of Phidgets example code)
4. Write program to capture IR from a TV remote and save to a named file. (extension to Phidgets example code)
5. Write program to replay captured IR from a file. (extension to Phidgets example code)
6. Write a demo program that incorporates items 3, 4, & 5, perhaps pressing each button changes to a different channel on a TV.
7. Install CEPSTRAL Personal voice and get it running.
8. Install KODI and get it running.
9. Investigate various ways of getting song and movie meta-data out of KODI.
10. Investigate various ways of telling KODI to play a particular song/album/movie.
11. Design configuration and file format that will drive the list program. XML?
12. Based on knowledge of all the preceding items decide on the programming language to implement the main nested list processing program (NLPP). Presumptively it is Python.
13. Write a first iteration of NLPP that runs off file definitions and integrates buttons presses, IR output, and Cepstral voice output. No volume control at this point (just a manual analog control).
14. Write a program to get movie meta-data out of KOI and present it in ordered lists within NLPP.
15. Write the Movie presentation function of NLPP.
16. Write a program to get song meta-data out of KODI and can present it in an ordered list in NLPP.
17. Write the Music presentation function of NLPP.
18. Evaluate the 3 options identified for rotary encoders as well as the option of using the TC Level Control as an input control to the PI so it can control the volume rather than as an output analogue volume control. Come up with a volume control and audio strategy. A solution that uses the Cirrus Audio card is acceptable.
19. Write the volume control interface and integrate it.
20. Full demonstration of system using the Movie and Music modules.
21. Testing by someone other than the developer. Fixes, changes and improvements coming out of this.
22. Testing by target user. Fixes, changes and improvements coming out of this.
23. Build second set of hardware so it can left with the target user.
24. Test, fix and make first release to the target user. Celebrate!
25. Write the Internet RADIO function. Test, fix and release to the target user.
26. Write the recorded audio function. Test, fix and release to the target user.
27. Write the Live RADIO function. (maybe, depends how well internet radio works, requires additional hardware)
28. Write the “place a phone call function”. Small list of fixed names and numbers. Require additional hardware.
29. Write the “answer a phone call function”.

Testing the system:

1. Ideally the tester hasn’t been involved in the development of the system so that they have no prior knowledge of who it is supposed to work.
2. simulate slight cognitive impairment (stay up all night or drink two beers (volume needs to calibrated to the user)).
3. Sit in a dimly lit room, put on sunglass that have been smeared heavily with Vaseline, and pretend you can’t read any text or numbers. All this is to simulate the visual agnostia. You can place a small spot light on the red button as the user can see red about 100 times better than any other color.
4. Put on thick ski gloves to simulate part of tactile agnosisa a.k.a. clumsiness
5. Now try to navigate to and play the movie/music you’d like. Play it for a bit, and then try going to another item.