**Cube board test jig**

Problem:

There was a need to test a large quantity of sensor printed circuit boards that were vital for some of the systems being created by the company. The problem arose from the time needed to test an individual board and the potential for human error as each board required 14 values to be sampled and recorded by hand. This process took on average around 6 minutes.

Solution:

The LCR meter used to retrieve the values from each node on the board seemed to have a USB interface on it. Because of this I believed I could develop a solution to both eliminate human error and significantly decrease the amount of time needed to test each board using script.

Initial difficulties:

After researching the LCR meter being used at the time I was able to uncover an English version of the originally Chinese user manual, and it referenced a method of querying the LCR meter from a computer using VISA and the scpi protocol. However, after investigating this protocol and the visa software I realised that the preloaded drivers for the current LCR meter were not compatible with VISA. After attempting to create custom drivers and being blocked by windows 10 we decided it was worth getting a more modern LCR meter with compatible drivers.

Method:

From the research into the visa software I found a reference to a python library that handles usb serial communication with SCPI, the core of the app is built on this.

As this application was going to be used by

others I needed a GUI, to avoid the need for extra software I decided to create a web dashboard hosted on the raspberry pi that drives the test jig. This dashboard allows a user to control the test jig from an internet browser on the local network. I employed a custom input validation function and jQuery to seamlessly connect the jig which is controlled by a node webserver with the front end and the python jig driver script.

Front end

Web server

Database manager

Jig driver python script

Blue: test board instructions

Orange: instantaneous results

Green: individual board information stored as json

Black: database as a csv file passed downloaded by front end

The python jig driver script loops through and queries each node on the subject PCB (using GPIO to control relays, and SCPI to query the connected LCR meter) and serialises the Inductance and resistance values for each in json format which is used as an interface between the driver script and the database manager. On driver script completion, the webserver calls the database manager along with the board number from the front end. The database manager takes the json data and parses it into the database. It will also alert the front end if the board failed, what the values were, and which node it failed on. On completion, the front end becomes active again (you can enter new values) and the value iterates by one automatically if the corresponding checkbox has been clicked. From here the user can either test a new board or download the database in the form of a CSV file. Finally, I added a physical off button to the jig itself as to shut down the raspberry pi safely without having to access the ssh.

Database retrieval issues:

Due to difficulties with asynchronous nature of JavaScript the for loop I was using to format the retrieved results from the database into csv became confused and I would end up with missing line and bad data. Because of time restraints I initially created a recursive function with call-backs to force the loop into executing in the right order. After successfully testing the current batch of boards I went back and replaced the recursion function with a promise loop which is something I had never used before, coming from a background in Java. This new function is far more streamlined and works quite a bit faster than the recursion function which, due to the limited processing power of the pi is extremely lucrative.

Conclusion:  
After testing the application and jig I have found it to be a reliable and helpful machine, apart from the complete removal of human error, each board takes about 40 seconds to test rather than 6 minutes, most of this time can be attributed to the raspberry pi taking its time to open the python script. Throughout this process I have developed my skills in postgres sql, javascript, the express node framework, raspberry pi and linux operating systems, as well as GPIO and USB interfacing.