# What do we need?

At the minimum, we need the following:

* A barcode scanner
* A microprocessor device like Raspberry pi. This could be done with a microcontroller like Arduino, but that could mean that more work be done on our end.
* Screen preferable 7 – 10 inches that has touch screen capabilities
* An application that can process inputs given to it by the user. This application will need some sort of GUI.
* A database to store the information into. The database will need to track a part number, serial number, quantity, user’s name, aircraft it was charged to, the date the transaction happened, and optionally a Boolean value whether part has been processed or not.
* A way for a client to connect to the raspberry pi server.

# In Both Cases:

We will need to configure the raspberry pi to auto run a program on boot up using the Raspbian OS. You will also need to configure the raspberry pi to auto log in every time. In both cases we will want the loading boot screen to have a splash screen.

# Approach 1:

Here I can use a Raspberry pi as the computer server. The raspberry pi 3 B+ can be configured to boot with a custom load/splash screen and then bring you to a URL that cannot be changed by the user. The URL can be changed through SSH tunneling from a client PC to the server raspberry pi.

What happens after the custom splash screen? This approach would boot into a Linux distro and run a browser called Chromium and this would allow us to limit the end user’s access only to the site that we want them to use. Once the application is loaded the user can then proceed to start to use the system.

This approach means that we would have to make a custom web application to process the barcode reader. This can be done using python and the Django or Flask framework. By doing this with a web application, we could treat the application like a website. We can host it on the internet or within our local network and our computers could interact with it as well. We could also expand this application easily to take on more online based tasks. The main benefit is that anyone with a browser could have access to this site, assuming they have proper permission and credentials to do so.

Where would the database be held? Locally on the raspberry pi? If that’s the case, then we will need a way for the data on the database to be accessible to a client PC. We can make a page that displays all parts that have not been processed. We could add to the charge out table a column named Procced and it can be a Boolean value that reflect whether a part has been processed by the client or not. Then the client user can change these values in the database stored on the raspberry pi to reflect the parts has been processed. This would then be reflected on the client’s parts to charge out page. A download link could also be given on this page.

Possible draw backs, that this may be way more than what we need. Also could be a lot of work in the future trying to set up a connection between these two devices.

# Approach 2:

This approach would use a client/server set up with standalone applications. This approach does not limit us to making a python program, so if we wanted to make a c++ program or java program we could have that load instead of making a Django/flask web app through the chromium browser. With the standalone application ,we would need to establish the client – server relationship by creating sockets that connect each other. The socket will allow data transfer between the two systems.

The main advantage to this approach is that we can make a self contained system that will work without having to be dependent upon a central database and a website UI. Reason is, in the future, the plan is to distribute this same system to other places. To reduce the maintenance and complexity, the self contained approach allows the user to use the raspberry pi server to enter data, and then allow a client PC user to manipulate and retrieve the data from the database that resides in the raspberry pi server. If anything needs to be fixed, it can largely be done by the user and they shouldn’t need much help after the initial installation.