Punchbag project

This is a modification to the project originally posted here
https://www.sparkfun.com/news/2115 you can find a copy of the post with this documentation.
I am doing this documentation so that you can see the thought process which went into the project.

Step one was to visualize the data. I felt it would be better to do this as an overlay on the video. First I needed to create a file with the data in a set sequence, the original had gaps. I made a script to do that(convert.py) I created a python file (plottest.py) to create the moving plot. I only used the 3-77 data points for this. I then used RecordMyDesktop to capture the video and then OpenShot to superimpose the one on the other, Im not sure I got the sync quite right on the result (vidtest.mkv) For the plots I used the x,y,z and a total magnitude.

The first impression of the plots were that it should be possible seeing the activity at a punch being much higher than elsewhere. You can see the decay of resonance after each hit. After chatting with a Friend who knows more than me I consulted this wikipedia page.
https://en.wikipedia.org/wiki/Matched_filter

Correlation seems like it would be the best way to do this. I had three problems with this, I dont think I had a perfect punch to compare to, I doubt it will work on a 328 processor and I really did not have the time.

I figured that if there is plenty of activity when there is a hit what if I just used an average of a few data points and checked for a threshold. I guess this is a kind of low pass filter. I also just used the z data and just the difference between them. I also tried at a later stage to use the complete vector but it ended up with similar results.

After a few manual tries to find the right buffer size and threshold a thought why not try this in a brute force fashion this helped me get to the values I got to. Not getting perfect matches I thought to make the algorithm better. I then saw that the original was dividing the hits by 2. Interestingly enough I got similar results. I then took the sample code from the original, which Im not sure how it works, and put this into the brutforce system(bruteforce2.py) to see If I could get better results. This turned out to be a failure. I guess I was implementing it wrong.

If someone gets a better algorithm(Im sure its possible) Im guessing the brute force method might be a good way to calibrate it. My guess is different setups will result in different calibration values. Maybe a good way to calibrate a setup would be to run the brute force method on the unit itself.

I played around a bit with the program and changed something small in the end. Here are the results of the experiments with the algorithm used

Test1
Take an average over the last 79 data points (158ms) if its greater than 13 count it as half a punch. Also making sure not to count punches unless they are at least 158ms apart. 2.53% seemed like I could do better
Results:

file	result	error
3-77	77	0%
4-81	81	0%
5-93	93	0%

6-79	77	2.53%

Mysterydataset1 = 149

Mysterydataset2 = 150

Test2

Take an average over the last 76 data points (152ms) if its greater than 17 count it as half a punch. Also making sure not to count punches unless they are at least 152ms apart. 1.29% seemed like I could do better

Results:

file	result	error
3-77	78	1.29%
4-81	82	1.23%
5-93	93	0%
6-79	79	0%

Mysterydataset1 = 154

Mysterydataset2 = 153

Test3

Take an average over the last 156 data points (312ms) if its greater than 13, count it as a punch. Also making sure not to count punches unless they are at least 312ms apart. 2.15% seemed like I was going backwards

Results:

file	result	error
3-77	77	0%
4-81	81	0%
5-93	91	2.15%
6-79	78	1.26%

Mysterydataset1 = 151

Mysterydataset2 = 154

Test4

Take an average over the last 76 data points (152ms) if its greater than 16 count it as half a punch. Don't count the first and last one. Also making sure not to count punches unless they are at least 152ms apart. 1.26& seems like the best I've gotten so far

Results:

file	result	error
3-77	77	0%
4-81	81	0%
5-93	93	0%
6-79	78	1.26%

Mysterydataset1 = 153

Mysterydataset2 = 153

I modified the origional.ino file to do the above but I do not have the hardware to test it on.

I hope I did not do anything wrong in the licence or modifications. If so please point it out.

Any questions or suggestions please email me (plakkies at hooligans dot co dot za)