The Oswestry Pullometer (How it Works)

**Disclaimer**

This system was developed in response to John Harrison’s challenge (The Ringing World 9/12/16) and proved to be a learning experience in many areas. I freely acknowledge that Pete Neil made the 3-pulley box and without his input none of this would have been possible. I claim no expertise in terms of hardware or software related to the project and the development took several false turns. As a result, the software incorporates some code that is now redundant and some unwanted hardware is still used to save large scale rejigging of the project. In short, there are almost certainly better ways of doing everything, but this worked. Feel free to make your own modifications.

**Basics**

The system is based around the Raspberry Pi computer for the simple reason that I had been experimenting with one when the challenge was issued. I have tested the software and hardware on Models 3B and Zero and on laptops running Windows XP and Windows 10 although I have not tried wireless communications.

One Pi is based in the belfry, mounted on a convenient frame member, taking measurements which are passed, using an ethernet cable, to a computer in the ringing room (another Pi or a PC) recording the information and displaying output.

**Sensors**

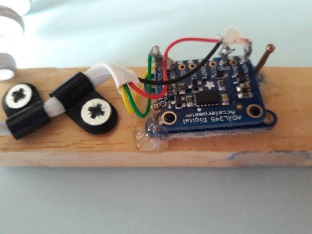
Although the challenge related to pull I was already convinced that it was important to monitor bell position as well. Initially I thought a digital accelerometer (ADXL345) would be useful but vibrations from the bells ruled this out. Instead, I used a digital gyroscope (adafruit L3GD20) mounted on one of the wheel spokes close to the centre of rotations. Since the sensor moves with the bell but the computer is fixed, a means of connecting them is required. I use a coiled 4 core telephone cable which wraps around the gudgeon as the bell reciprocates. Although I view this as a replaceable item it has been in use at Oswestry for over 9 months without any problems. I hope that the extensibility of the cable will accommodate the additional motion should a stay break.

The pull is measured by a 3-pulley system mounted just below the main ground pulley. An upper and lower pulley are positioned to be in line with the rope path, but a central pulley slightly deflects this path. Tension in the rope tries to displace this pulley and the resulting force is measured by the system. Pulleys tend to be rather expensive so banks of three 19mm diameter roller bearings were used in place of each pulley (MOTIONCO BR08M-19) at a total cost of less than £10. The force is sensed by a load cell (Active Robots 3133\_0 Micro Load Cell [0-5kg] CZL635) connected, via a long 4 core cable, to a digital amplifier (Active Robots Load Cell Amplifier - HX711). We used pivot centres of 160 mm top to bottom with the centre pulley offset by 18mm. This produced acceptable output from the 4th at Oswestry although I would recommend making this offset adjustable. The physical design is such that in the event of a stay breaking the sally can pass through without restriction.

**Pulley block details (Pete Neil)**

The pulley block was made from 12mm sheet material, either plywood or MDF would be suitable, with a spacer of 21x12mm close grained softwood. The bearings were mounted on M10 threaded rod with spacing washers between. Four pieces of the sheet material, one 50mm wider than the others to allow for mounting, were clamped together and drilled to take the pulley axels. The pieces were then separated and the outside leaves centre holes drilled out to 12mm to allow for movement of the sensor pulley block. The inside leaves were then screwed to the softwood spacers down the outside edges and then the outside leaves were screwed in place. The whole block was then separated and the inside leaves channelled out to allow a sally to pass through. This will leave four inside leaves, two which hold the fixed pulleys and two that hold the pulley connected to the sensor. The sensor side was then re-assembled, suitable holes drilled to accept the set screws which hold the strain gauge in place and a section holding the pulley separated with two saw cuts. This separated piece was sanded down slightly to allow for free movement between the outside leaves. A slot was cut down the outside leaf of the box to allow the bell rope to be slid into the block and the whole block was reassembled, Strain gauge first, then sensor pulleys and the remaining wooden parts glued and screwed together using the previously made screw holes inserting the idler pulleys in place as progress was made.

**Connections**

The ADXL345 (if used) and L3GD20 boards can be mounted on a small board ready to be strapped to a spoke on the wheel. They both need power and a pair of communication (I2C) wires making a 4-core telephone cable quite suitable. Connect wires to the points marked SDA, SCL, GND and 3vo, carefully noting which is which. The connections are made in parallel between the two boards. I cut the pre-moulded connectors off the cable and simply soldered the wires in place. Although this has worked for over 9 months without failure, the cable will need replacing from time to time so using a socket on the board will make this easier.

*N.B. since the ADXL345 board is no longer used you can save money by omitting it but you will need to modify the code to remove reference to it. This will mean you need to substitute dummy data to be transmitted, or modify all the ringing room code.*

The other end of the cable makes connection with the GPIO pins on the Raspberry Pi (again a socket will make changing cables more convenient). Please check a guide to the GPIO pin numbers to trap any errors I make, but I believe they should be as follows: -

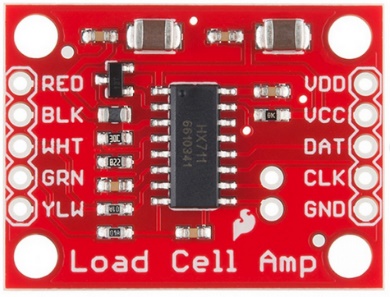
3vo - Pin 1

SDA - Pin 3

SCL - Pin 5

GND - Pin 6

Now turning to the load measurement, the belfry layout will determine where the 3-pulley block is mounted. It must be below the ground pulley but as high as possible (although friction will be quite low you need as much weight of rope below the system to keep the rope moving freely). At Oswestry we mounted the unit on the ceiling of the clock room. The sensing element is a shear beam load cell which, until recently, tended to be expensive but now can be obtained for less than £10. The type I have located are marked ‘Home use’. I’ve no idea what this means but take it to indicate that you need to sort out calibration for yourself (you will need to do this in any case).

The transducer consists a rectangular beam with holes carefully designed to make the attached strain gauges sensitive to the applied shear force. The strain gauges are wired to form a Wheatstone Bridge and therefore four colour coded wires are provided for the power supply and output signal. A 4-core cable can be used to connect this to the digital strain gauge amplifier, (again I have used telephone cable and tests would indicate that cable length need not be a problem). IMPORTANT: On the underside of the board you will see two small pads with the word RATE written beside them. Take a sharp knife or scriber and firmly cut between these pads to break the connection. Without doing this you will reduce the sampling rate by a factor of 10 and the results will be useless. Then simply connect the correct colours to the connections shown on the left-hand side. The YLW connection can be ignored but if you have shielded cable the shield can be connected here with the potential of reducing noise (my experience suggests that noise is on the digital side rather than analogue). Now connect the board to the GPIO pins on the Raspberry Pi as follows: -

GND - Pin 9

CLK - Pin 11

DAT - Pin 13

2.7-5V - Pin 17

For the system to be useful the information must go somewhere and an ethernet cable is used to send this to the ringing room. If you are using a full-sized pi then a network socket is provided. When working with the pi zero use the USB port and an USB to ethernet converter. The other end of this cable needs to be connected to a computer which is powered up for the software to function.

At this stage you should have a working system that can monitor output from the force gauge and the gyroscope, but it is difficult to test it when you are in the belfry and any output is in the ringing room. Two methods to check function are provided. If you can get a monitor with an HDMI input into the belfry then a simple plot of force output plotted against time is provided. This needs quite a lot of effort, so a simpler system makes use of a LED mounted between pins 37 and 39 (ground). Please be careful to get polarity right and use a suitable current limiting resistor. This LED will flash rapidly if there is no ethernet connection and slowly if the system is functioning correctly.

Should you wish to monitor a switch, provision is made between pins 33 and 30 (ground). The state of this switch is transmitted to the ringing room, but my software does not use it.

**Belfry Code**

The memory card for the pi in the belfry will need to create a folder called BellPull to which you download :-



The main code is datalogger, and this needs the support of L3GD20 and bitOps. These support files I obtained from <https://github.com/mpolaczyk/Gyro-L3GD20-Python/blob/master/L3GD20.py> but I advise using the version I have provided since the original contains a bug which has not been corrected.

The code to read the force gauge came from <https://github.com/dcrystalj/hx711py3> but is incorporated in datalogger .

The adxl345 code came from <https://github.com/adafruit/Adafruit_Python_ADXL345> and will be needed unless you have recoded.

Since you are unlikely to have keyboard or mouse in the belfry I intend that this code should run as soon as the pi powers up. You will need to modify the start procedure for this to happen but there are online guides describing how you do this.

**Ringing Room Code**

There are two coded components for the ringing room, one giving you control of when files are recorded from the ringing and one giving various display options for any saved files. To work with a PC you will need to download and install Python2.7 (free download). Although this is not the most recent version of Python all the code will run on this. This version is preferred since more recent versions are not compatible with Windows XP which I expect is common in towers.

On the PC or the card on the ringing room Pi crate a folder named BellPull to which you download the following files: -



and Church\_sketch.jpg which displays the tower of St Oswald’s, Oswestry revealing the origins of this project. You can substitute your own picture if you wish.

Within the main folder you should create 4 additional folders named Setup, Data, Inter and Final.

Within the Setup folder use something like Notepad to create the following files:

* NextFileNumber.csv containing the single number “1” followed by rtn.
* Calibration.csv containing “1,0” followed by rtn.
* Setup.csv containing “1,1,0,0,0,0,-1,1,0,1” followed by rtn.

**Operation**

The Pi in the ringing room will constantly monitor the sensors until power is removed. The results are streamed via the ethernet cable to any machine willing to read them.

BellPull-Grab-3 will continuously read the arriving data. The raw data from the force gauge is shown on a thermometer like display which can provide confidence that the system is working. This is raw data and noise on the signal may cause this to jump a little even when no force is being applied. Left clicking the mouse within this window will start and stop data collection. As soon as the stop command is given you should initiate data processing using the display software BellPull-Overlay-Gn. This should be completed very quickly and this will allow viewing any selected file(s) in the preferred format.

**Data Structure and Processing**

*Belfry*

Data is sent from the belfry in the following, comma separated, format : -

1. Raw force data
2. X accelerometer
3. Y accelerometer
4. Z accelerometer
5. X *gyroscope*
6. Y gyroscope
7. Z gyroscope
8. Time
9. Counter (used during development phase)
10. External switch.

This information is sent out repeatedly at about 70 cycles per second.

*Ringing room data capture*

When commanded to capture data the first action is to access the calibration file, read the contents, and write this information into the output file. This ensures that the wrong calibration cannot be applied if files are reprocessed. Then the 10 items listed above are repeatedly read and written to the output file until the stop command is received. The resultant file is named BellPullRaw*xxx.*csv, where *xxx* represents a sequential file number.

When requested, the captured data file is processed through a number of stages with intermediate files written into the Inter folder at each stage. The stages are : -

1. Apply the calibration factors and try to trap short burst digital noise.
2. Apply general smoothing.
3. Derive acceleration from gyroscope data.
4. Integrate gyroscope data to derive position. (N.B. This assumes the bell starts form being set and you need to measure the set angle and insert this into the code.)
5. Apply corrections for gyroscope drift (N.B. The assumes that the bell is set at the end of data collection.)
6. Derive events such as TDC, BDC, direction reversal.
7. Note times per stroke and whole pull.

The derived information is written into the Final folder and is ready for plotting or other analysis.

The Inter files are all left so each stage of data processing can be checked if problems arise. If storage space is limited these files may be deleted.

**Improvements/Extensions**

I’m aware of many improvements that could be made but I need to concentrate on other projects for now. Examples include: -

Elimination of all reference to the accelerometer.

Modifying code in the belfry so that it will loop until a network connection is made.

Modifying the ringing room code so that starting and stopping data collection using the mouse is more reliable.

Processing the data files as soon as capture is completed (this would make the linking with the calibration file more secure).

Modifying the ringing room code so that data collection can be started and stopped from the bell force/motion. I have made a start but it has bugs.

Adding a calibration routine.

Producing a ‘Stay protector’. The Gyro can detect small bounces on the stay if the bell is not set very carefully. Given how some ringers seem to casually set their bell this feature could be turned into a competitive game with each ringer seeing how carefully they can set the bell (eliminating bouncing).

Merging the two ringing room files into a single application. I separate them because one seemed to work best using Pygame and the other Tkinter. I suspect that with careful coding the functions could be combined.

**Richard Major** February 2018