

**Year 2 Project**

**Wheel Truing System**

**By: Connor Brizell (201275492), Daniel Fox (201278002), Chinemerem Duruaku (201317167)**

**Supervised by: Professor Jeremy Smith**

**Department of Electrical Engineering and Electronics**

**25 March 2019**

# Abstract

The report written on this Year 2 Project report looks at how a Wheel Truing System is used to resolve the problems cyclists face for being able to correct the buckle and ovality. The approach to this problem was to design an automatic mechatronic system which can reduce the difficulty cyclist face when truing a bike wheel. The purpose of the project was to make the process easier and faster than manually truing a bike wheel. The project was split up into three teams of three students, each focused on different parts of the system. This report is on the image processing team which was to produce a program which can detect the location of the buckle and ovality when tracking the bike wheel. The image processing team were successful in programming the buckle and ovality locations but unfortunately, due to time constraints sending the data to the other teams was unsuccessful this is discussed further in this report.

**Declaration**

**I confirm that I have read and understood the University’s definitions of plagiarism and collusion from the Code of Practice on Assessment. I confirm that I have neither committed plagiarism in the completion of this work nor have I colluded with any other party in the preparation and production of this work. The work presented here is my own and in my own words except where I have clearly indicated and acknowledged that I have quoted or used figures from published or unpublished sources (including the web). I understand the consequences of engaging in plagiarism and collusion as described in the Code of Practice on Assessment (Appendix L).**

Contents

[Abstract 2](#_Toc4432223)

[Acknowledgements 6](#_Toc4432224)

[1.0 Introduction 7](#_Toc4432225)

[1.1 Aims and Objectives 7](#_Toc4432226)

[2.0 Research 9](#_Toc4432227)

[3.0 Materials and Methods 14](#_Toc4432228)

[3.1 Materials 14](#_Toc4432229)

[3.2 Methods 15](#_Toc4432230)

[4.0 Results 21](#_Toc4432231)

[4.1 Error Analysis 22](#_Toc4432232)

[4.2 Multimedia 24](#_Toc4432233)

[5.0 Discussion 25](#_Toc4432234)

[5.1 Analysis of Results 25](#_Toc4432235)

[5.2 Limitations 27](#_Toc4432236)

[5.3 Applications 28](#_Toc4432237)

[5.4 Ethical Considerations 28](#_Toc4432238)

[5.5 Commercialisation and Intellectual Property 29](#_Toc4432239)

[6.0 Conclusion 31](#_Toc4432240)

[6.1 Future Work 31](#_Toc4432241)

[6.2 Conclusions 31](#_Toc4432242)

[References 33](#_Toc4432243)

[Appendices 35](#_Toc4432244)

[Appendix A 35](#_Toc4432245)

[Camera 1: Detecting the Buckle 35](#_Toc4432246)

[Appendix B 38](#_Toc4432247)

[Camera 2: Detecting Ovality 38](#_Toc4432248)

[Appendix C 40](#_Toc4432249)

[ASCII conversion 40](#_Toc4432250)

[Appendix D 41](#_Toc4432251)

[Example of Sending messages to raspberry pi 41](#_Toc4432252)

List of Figures

[Figure 1:Wheel Diagram 8](#_Toc4432150)

[Figure 2:Greyscale colour values 9](#_Toc4432151)

[Figure 3:Sobel convolution 10](#_Toc4432152)

[Figure 4:Sobel operation of bike wheel 10](#_Toc4432153)

[Figure 5:RS232 example 11](#_Toc4432154)

[Figure 6:RS232 Logic 12](#_Toc4432155)

[Figure 7:Flow diagram 14](#_Toc4432156)

[Figure 8: Resolution x pixel diagram 15](#_Toc4432157)

[Figure 9:Original histogram 16](#_Toc4432158)

[Figure 10:Filtered Histogram 17](#_Toc4432159)

[Figure 11: Image of white background histogram 17](#_Toc4432160)

[Figure 12:Buckle and ovality camera setup 18](#_Toc4432161)

[Figure 13:ASCII conversion 19](#_Toc4432162)

# Acknowledgements

All members on the project are currently studying mechatronic and robotic systems, this project was a great learning experience. Please see Appendix E for project contribution towards the report. We would like to show our appreciation to our supervisor Prof. Jeremy Smith in his contribution of academic support and advice towards the competition of the project.

# 1.0 Introduction

The purpose of this project is to create an automated wheel truing system using mechatronics to improve the process of truing a bike wheel. The problem many cyclists face when their bike wheel is out of true is that it can cause issues when riding such as wobbling on the brakes which could potentially cause a crash, hence it’s important for bicyclists to make sure their bike wheels are properly trued. Bike wheels out of true are commonly caused from crashes and riding over obstacles which can become common for cyclists. There are manual ways to true a bike wheel by adjusting the spoke tensions with a tension tool however people find it tedious and time consuming as it can take a long time to figure out how much tension is needed to tighten the spoke and reach a perfectly trued wheel.

The project was created to get rid of this problem by creating a process which is much easier for cyclists, an automated system would make sure the wheel is perfectly round and straight with the perfect tension with little effort on the user’s part, the system is designed to improve the time and the struggle in truing a bike wheel. This system also demonstrates a perfect example of how well mechatronic systems can work, using electrical, mechanical and computer science together in one system.

The project was split into three sub groups which included three students in each group, splitting the team up into three sub groups helped speed things up for the five-week deadline. The sub teams split consisted of groups of three people, working on different fields, this report is on the image processing team;

* Group 1: Image processing team
* Group 2: Spoke detection team
* Group 3: Data processing team

## 1.1 Aims and Objectives

The overall aim of this project was to create an automated wheel truing system to make a bike wheel perfectly trued.

Each group have been instructed to create a report depending on their group, this document is a report on group 1 – the image processing team. The objective was to create a working program which detects the buckle and ovality of the bike wheel by tracking the location of the wheels movement from left to right, then storing this data and sending it to the data processing team using serial communication. The main objectives can be seen below:

* Create a program to determine the location of buckle and ovality
* Send data to data processing team using serial communication methods(e.g RS232, Bluetooth)

# 2.0 Research

At the beginning of the project, research was conducted before any program design started.

**Buckle and ovality**

At first to understand what was to be programmed, research was carried out to give a brief understanding how buckling and ovality effect a bike wheel and why it was important to have a perfectly trued bike wheel.

Buckling is caused when the wheel is under compressive stress, if the maximum weight of the rim is exceeded then the bike wheel can most likely buckle also resulting in an ovality which causes the circular shape of the bike wheel to deform. The wheel can buckle due to factors like accidents, riding over rough terrain as well as general use. The ovality of the wheel is altered when the spokes on the wheel are loosened causing the shape to deform [1].

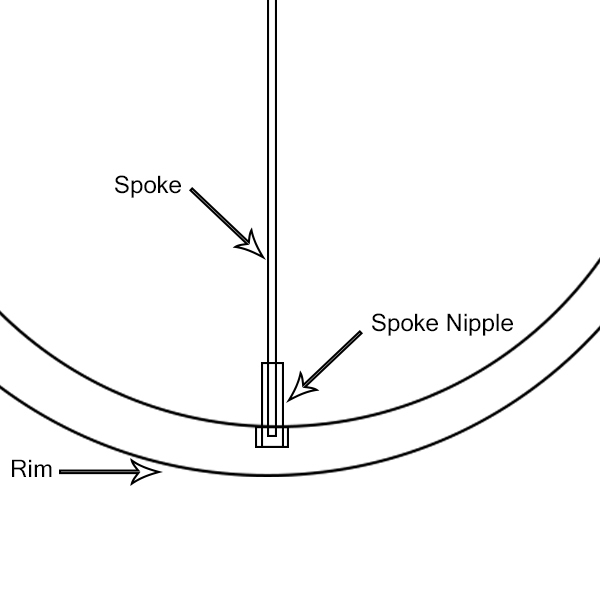


Figure 1:Wheel Diagram

**Programming Language**

Choosing a programming language was the first decision the group had to make, this decision was between python and c++ , these are two languages the group had no experience with, the decision was to go with c++ this is because the group had previous experience with c programming. C++ was similar to C which helped when designing the program as there isn’t much difference between the two languages.

**Image processing libraries**

C++ was the choice in programming language however external libraries where needed to be implemented as this process made it easier to code. OpenCV is one source which included libraries made specifically for image processing which became really useful, OpenCV was the main choice as this is one of the most commonly used sources for image processing.

OpenCV needed to be researched due to inexperience working with this library, the source has programming functions aimed at real time computer vision operations, it includes built in algorithms which can be used to detect and recognize faces objects in a camera, track movements, extract data from images and much more.

**RBG to Greyscale**

RBG to greyscale conversion was an important tool to use when image processing, as RBG contains lots of unnecessary data which is not required for this program. When the image is converted to greyscale it’s much easier to remove the noise and other factors in the image as it only has 1 channel to track the intensity value , whereas RBG has 3 channels consisting of 24- bits and since there is 3 channels to track lots of data is required to store making it much harder to manipulate the image. [2]

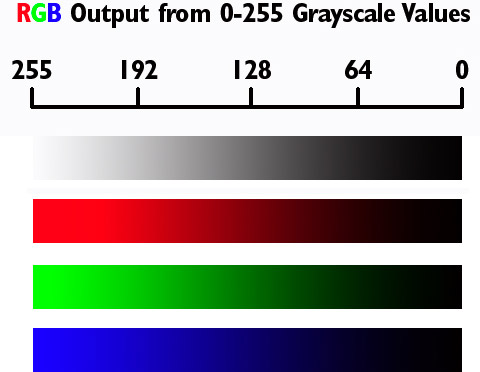


Figure 2:Greyscale colour values

A greyscale image consists of one single byte for each pixel, one byte can store a value from 0-255. The greyscale holds a two-dimensional array of bytes this size of the array being the size of the image (height and width). The array is known to be a channel thus a greyscale image has one channel, represented by the intensity of white colour [3].

**Sobel Operator**

The sobel operator was one method found when researching how to track the buckle and ovality of the bike wheel, this method was advanced compared to other operations. The reason to use the sobel operator would be to use it with edge detector algorithms where it creates an image emphasising edges, this can be extremely useful when tracking objects in an image. The sobel works by calculating the gradient of an image intensity at each pixel within the image, taking the x and y direction separately, with this the x and y axes can be combined to give the strength of the edges by using the absolute value.

When using this operator it’s much easier to convert from RBG to greyscale images, as greyscale only has 1 channel to focus on whereas RBG has 3 channels (red,blue,green).Then kernel convolution can be used, the image below shows the kernel these are derived from signal calculations which can be found online [4].

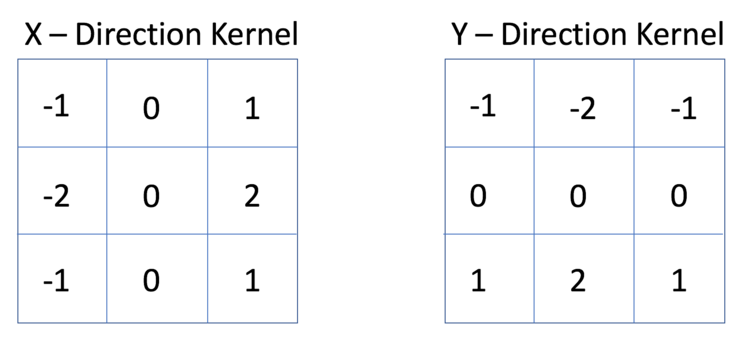


Figure 3:Sobel convolution

The above shows the kernels used to calculate sobel operations more information on this is found online [5].

These kernels can be combined or kept separately, when combined an image would be displayed like the one below;



Figure 4:Sobel operation of bike wheel

**Histogram**

A histogram can be used to track lines on an image, which can be useful when determining the ovality and buckle. As a histogram can be programmed to focus on a line in the image and once movement occurs it can spike and track the movement. A histogram shows the frequency of pixel values in an image which is useful to determine movement of the bike wheel.

**Serial communication (Protocol)**

**RS232**

The program needed a way to send the information to the data processing team, the decision to use RS232 was decided as its compatible with many devices as well as it’s a common communication protocol to use , which made it easier to research. RS232 standard logic is represented at logic high (1) which represents a negative voltage anywhere between -3 to -25V while logic level low (0) transmits positive voltage that can be from 3 to 25V, Most PCs alternate around -13 to +13V.

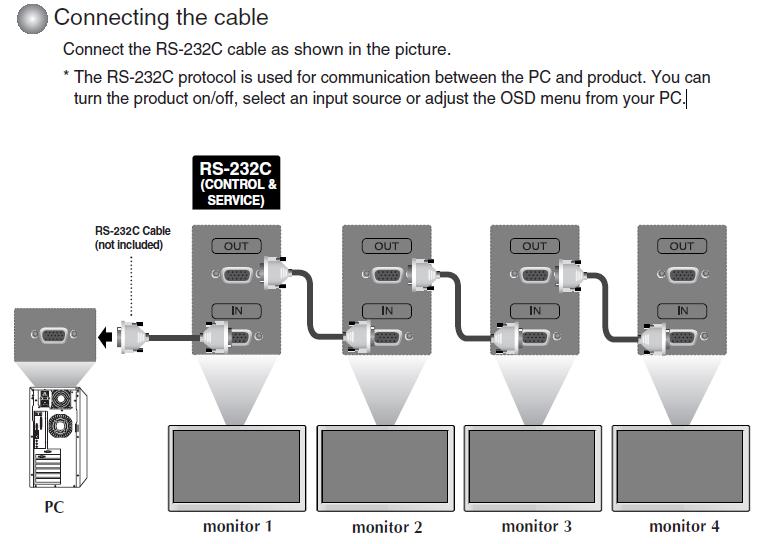


Figure 5:RS232 example

RS232 is common with many devices the example above shows a RS232 cable connecting to the pc and to the monitor to send data and read to each device [6].

The RS232 is a communication port which would be able to send data from a computer to its peripheral device allowing serial data exchange. To transfer the data through the serial port it is best to use ASCII code to send one bit of data at a time. [7].

**DTE and DCE** – These are the types of equipment at either end of the RS232 cable. DTE stands for Data terminal equipment and DCE stands for Data communication equipment. A computer is known as a DTE whereas a raspberry pi is known to have DCE equipment.

**Baud rate –** This is used to measure speed of transmission, this is described by the number of bits passing in one second.

Data Bits in a Communication Packet

Figure 6:RS232 Logic

The image above shows a typically RS232 waveform in logic which consists of a 1 start bit, 1 stop bit and a total of 8 data bits. The transmission starts with a start bit then followed by the data bits, LSB (Least significant bit) is sent first and the MSB (Most significant bit) sent last and then ending the transmission with a stop bit.

The mark (1) is the voltage logic “1” which is between -3VDC to -15VDC, while the logic “0” known as space (0) is between +3VDC to +15VDC.The RS232 connects the ground of two different devices together which results in a unbalanced connection which is more susceptible to noise resulting in a low distance limit more information on this procedure can be found online(look at reference [8]).

Figure 6 shows a simple 8 data bit communication packet which can be used in this project. As this is the amount of data needed to send ASCII messages to the receiver. Standard ASCII code consists of 0-127 characters, thus making 7 data bits enough, if the data is extended ASCII code then 8 data bits would be required (128 to 255).

**Bluetooth**

Another way to send the data over to the processing team is to use Bluetooth by connecting the two devices, the raspberry pi connected to the computer has Bluetooth support which can be linked with the data processing teams raspberry pi, this would allow the data to be sent over instead of using a RS232 cable.

**ASCII**

ASCII is known as American Standard code for Information Interchange based on the English alphabet, it is a 7-bit character set containing 128 characters, numbers containing 0-9 and upper to lower case letters from A to Z with also some special characters (e.g %). ASCII is used in most computers to represent text, which makes it possible to transfer the data needed from one computer to another computer or device.

# 3.0 Materials and Methods

## 3.1 Materials

**List of materials;**

* Two raspberry pi (3B+).
* Two raspberry pi cameras (480p resolution).
* Visual studios 2017.
* Cable connection (DVI to HDMI cable).
* Bicycle wheel.
* Black box.
* White screen.

**Raspberry pi \* 2 (3B +)**

* Raspberry pi\_1: This device helped to implement the ovality program using the camera\_1
* Raspberry pi\_2: This device was used to implement the buckle code using camera\_2 that checks for buckle.

**Camera’s \* 2 (480p resolution)**

* Camera\_1: this inputs data which is then used to measure the buckle
* Camera\_2: this inputs data which is then used to measure the ovality.

**Visual studio 2017**

* This is the application used to write the code which is in turn used to measure the ovality and buckle.

**Cable connection (DVI to HDMI)**

* Used to connect the camera output to the monitor.

**Bicycle wheel**

* This item was the basis for which the experiment was conducted. While the wheel turns the cameras, raspberry pi’s and the program all work together to output values which can then be used to measure the buckle or ovality.

**Black box**

* This was used to cover and protect the circuit.

**White background**

* This was used to block out the back-ground noise, so the program can track the line used to measure the coordinate more accurately.

## 3.2 Methods

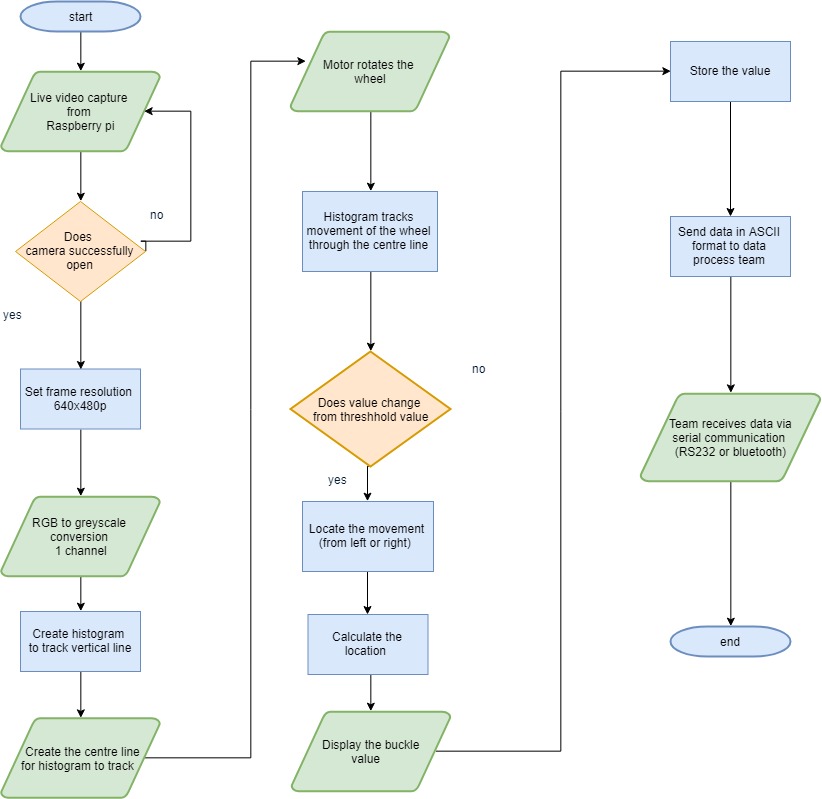


Figure 7:Flow diagram

Step 1: The first step involved setting up the system, this involved placing the wheel stand in a position for the two cameras to track the ovality and buckle. The two cameras are placed on separate stands which where set at equal level so the two cameras can track the same spoke at the same time.

Step 2: After the initial setup the first part of the code involved opening and editing the camera using code. This was done using the function “videocapture(0)” which opened the camera when the program was run. The second part involved changing the camera from RGB to greyscale colour to allow tracking to be easier. This was done using the function “CV\_BGR2GRAY” this function converts the image to greyscale, see appendix A and B for more code and comments.

Step 3: The next part involved deciding on a resolution and pixel value for the camera and since the cameras where set at 640x480p the decision was to keep the resolution and pixel value capped at 640x480p since it would give the best results. As the program runs and the camera opens the window for the video capture would be set at 640x480p. Since the resolution and pixel values are known, the video capture can be edited with code as OpenCV can track pixel values on the screen.

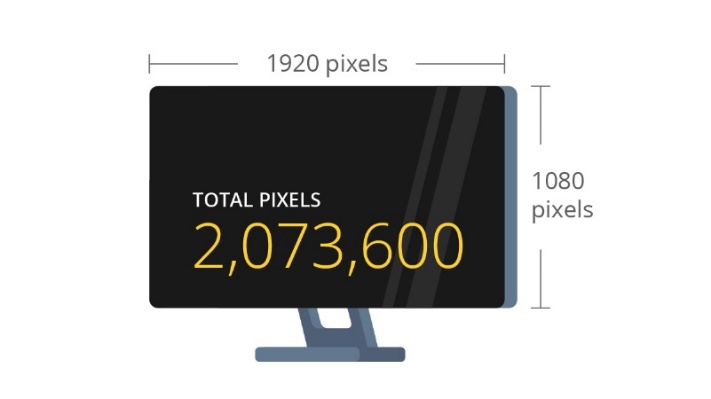


Figure 8: Resolution x pixel diagram

Step 4: Create a program which will display a histogram, which will be used to track the buckle and ovality. This is done by tracking the intensity line across the centre of the screen, then as the bike wheel rotates it will show a spike in the histogram if the wheel has moved from left to right. The histogram width matches the camera width which equals 640, the height is also set at 256 this value was chosen so the as a preference when displaying the histogram window. The histogram will track the vertical intensity line down the centre of the screen, if the bike wheel moves left or right this will show in the histogram as it would spike. The steps for this are shown below;

* Create histogram width and height using width at 640 and height at 256 which will then create a window at 640x256.
* The mat function is used to clear the background for the histogram, using this line of code “Mat histImage(hist\_h, hist\_w, CV\_8UC1, Scalar(255, 255, 255))” The 255 values make the background of the histogram white as the colours at 255 for greyscale are white. “CV\_8UC1” is the channel number in this case its 1.
* The final step in creating the histogram is to create the intensity lines which will show if the bike wheel has changed due to there being buckle or an irregularity with the ovality of the wheel.The line function is used to create the tracking line for the histogram; this function will create a centre line that will be used to track the intensity values. If the bike is buckled for example, then the line plotting the histogram will have a significant peak, which will then be used in an algorithm for determining its displacement.The intensity line is created by pointing to the width value (640) to assign the width of the line, then the height value is determined by taking the set histogram height (256), and then taking away the new width value (hist[i]). This then allows the histogram to track any change along the centre line due to the change in the displacement of the wheel’s buckle or ovality.The scalar colours are set at 0, so it distinguishes the intensity tracking line on the histogram from the white background. 0 will make the histogram intensity line grey. See appendix A and B which shows the code used to create the histogram.

Step 5: Once the histogram was created, it would be called in the main function to display the histogram and track the centre line.

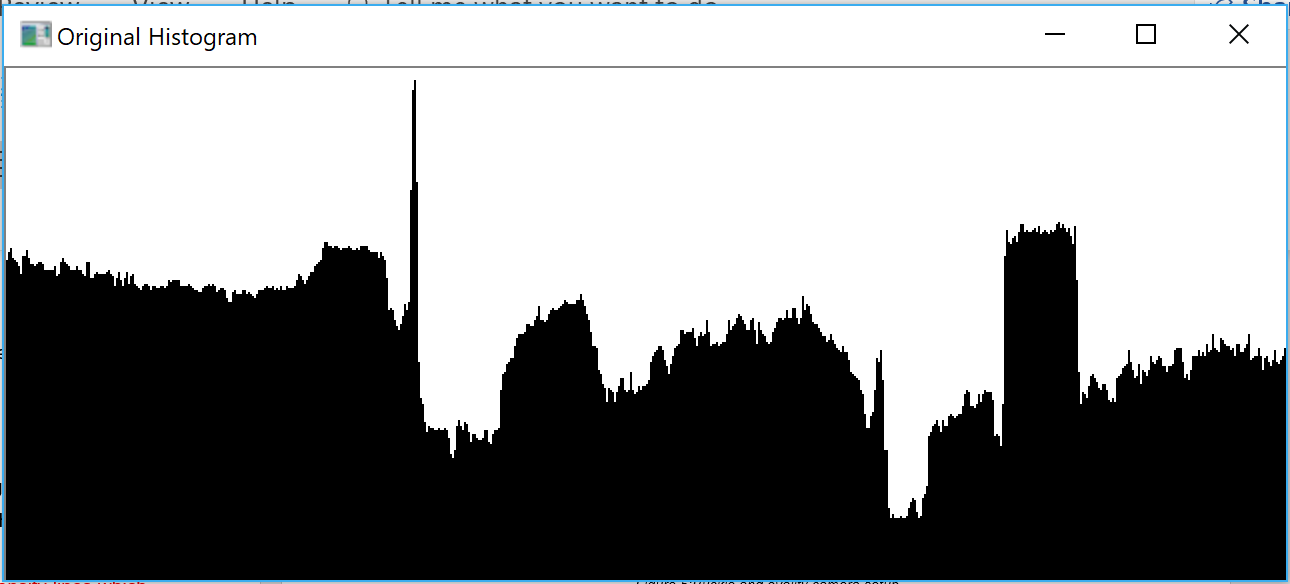


Figure 9:Original histogram

Step 6: Create a filter intensity variable, which makes the histogram intensity lines smoother to allow more accuracy

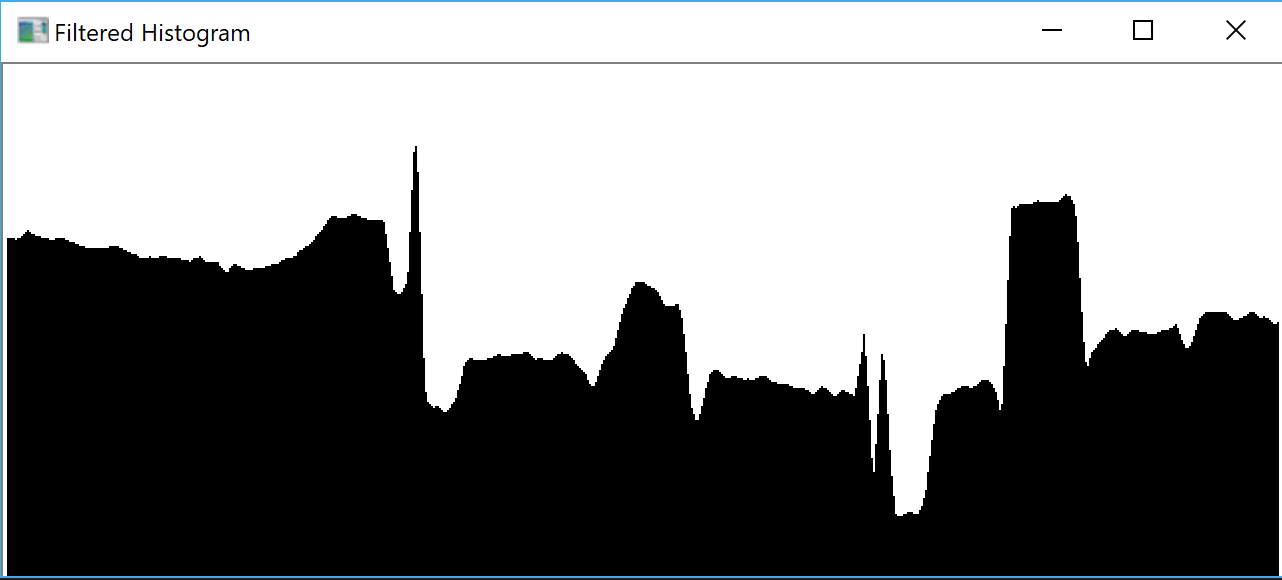


Figure 10:Filtered Histogram

Step 7: The main part of the code was finished, to reduce the background noise like shown in the images above cardboard with a white background was added which reduced the background noise. The above image shows multiple peaks due to the histogram picking up the background noise, when the card was placed it made the histogram more accurate when determining the buckle and ovality of the wheel.

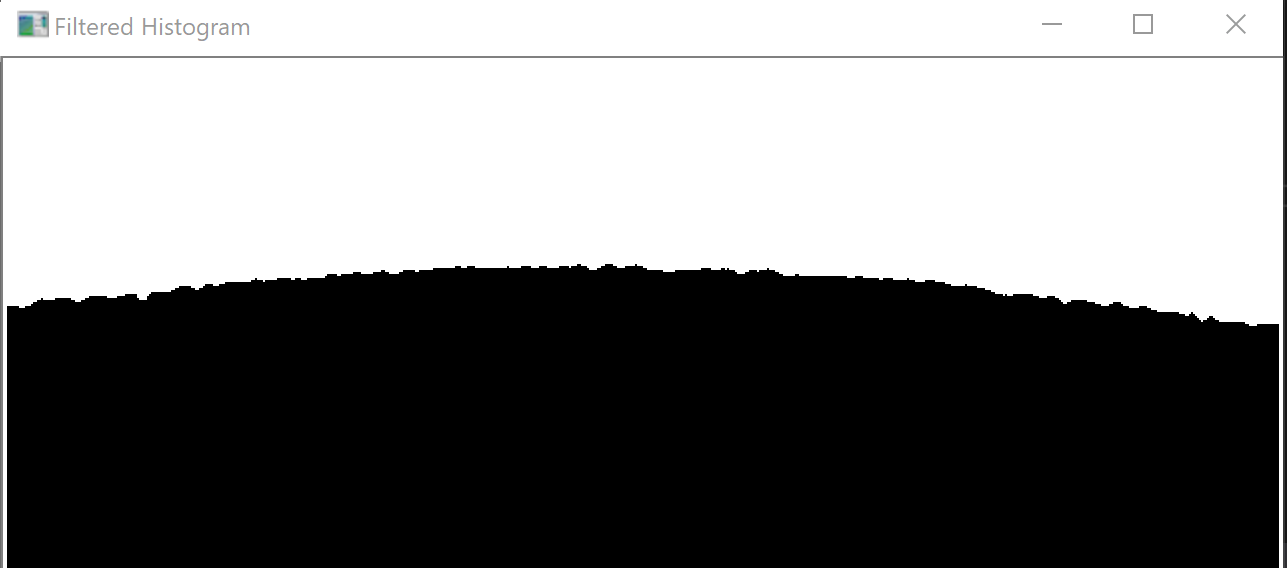


Figure 11: Image of white background histogram

Step 8: The histogram will spike when the wheel moves from left to right; the location of the displacement from the movement can then be determined. This is based on the intensity values plotted by the histogram feed from the input of the cameras visual display. The code initialises the restraints of the width of the values to range from 0 to 640 from left to right, based on the dimension of the window for the histogram being 640x480p. To reduce any noise or background interference from the light, an iteration is taken from the left side starting at 230 pixels, and from the right side at 410. These values are an equal distance from the midpoint at 320, and so the average taken will still be calibrated to determine the displacement.

Step 9: Once the above steps are complete this is then repeated for the second camera as the algorithm for the program does the same operation. The difference between the two cameras are the positions, the buckle camera is facing the wheel while the ovality camera is facing the side of the wheel.



Figure 12:Buckle and ovality camera setup

Step 10: Calculate the final value to send to the data processing team, this is calculated byusing the formula:

Displacement = (((left\_location + right\_location)/2)-320) / 10

Implementing this equation will print final value to the user for the displacement of both the buckle and the ovality of the wheel for each spoke. The algorithm is based on the histogram plot taking the highest intensity peak from the left and the right side, so the camera detects the edges of each side of the wheel. The average of these two values is taken, and then subtracted from the origin point of the midpoint at 320 in the x-plane. The value is then divided by 10 for the conversion ratio from 10 pixels to 1mm, giving a displacement value to a degree of uncertainty of ±0.5mm.

Step 11: To send the data to the team a protocol will be used to send the data to the processing team, this protocol will involve using serial communication. The RS232 cable would be used to transfer the data in ASCII one bit at a time. This would be done by some programming code which would convert the stored data location from the code into ASCII then this data would be sent over to the other team, once this is sent then the other team can use that data to correct the buckle and ovality. To send data from one device to another , example of using rs232 code is online which has been copied into this report to give a understanding on how the two devices would send and receive the data, all credit goes to the Stackechange forum [9]. The code picks up the serial port device and using the location of the device found from the RS232 setup, then the data can be coded and sent to the receiving device.

The code from Appendix D writes a string called “TEST” from the serial port location (e.g. /dev/ttyA) then storing that data. After, this is sent to the computer using a simple “printf” function directed at the stored data from the serial device. Even though this program doesn’t do what is needed for this project, it still gives an understanding on how the data would be sent between the two devices using serial communication ports.

A simple example of converting binary can be seen in Appendix C this shows program opens and asks the user to input the character and then this program changes the character to ASCII. This would then be sent to the data processing team using serial communication, the code was found online, and it has been slightly modified [10].

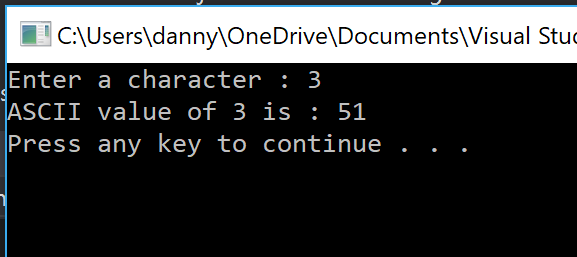


Figure 13:ASCII conversion

Appendix C and D shows two types of code which would be combined and modified to send the data from the raspberry pi cameras to the second raspberry pi which is used by the data processing team. It was unfortunate that the code for this wasn’t fully complete for this project as it’s hard to understand how this would properly work without testing the code in practice. The two pieces of code show an idea on how the team would approach the protocol to send data between devices.

# 4.0 Results

Results were successful in buckle from implementing the formula to determine how the bike wheel being tested has buckled either side of the neutral x axis. From initialising the origin point of the stands for the camera and for the bike wheel, the data was obtained not only from early testing, but also from results gathered during the bench inspection by manually rotating the wheel at a reasonable speed to allow the wheels frame edges to remain in focus for the camera.

As the width of the bike wheel was measured to be approximately 20mm wide, and the conversion between the unit of 1mm to pixels was 10 pixels per 1mm, the window created displaying the histogram would be scaled to this ratio. In that, because the window was 640 pixels wide, then approximating the wheel being in the centre of the frame taking up a third of the size of the window gave a degree of tolerance. Consequently, any displacement that deviated either side of the centre would reflect in there being buckle detected to within above or below 0.5mm of tolerance in the date gathered.

The results obtained for the ovality were also based on the same principle of the methods used for evaluating the buckle of the wheel. Using the other Raspberry Pi camera, the same formula was executed to determine the ovality. Ovality is based off the variation in buckle, so if the tension in the spoke is too high than in other identical spokes, the ovality of the wheel will change based on this adjustment. Therefore, it was fundamental that the camera’s used were synchronised to be level at the same spoke, in order to have this information communicated to group 3 for the group to make the necessary adjustments for the tension in the right spoke based on the position of its nipple.

The results obtained only show an ideal example of how the program works; the buckle and ovality produce values which are shown in the figures below. Unfortunately results for serial communication were not reached in time, due to the time constraints and from problems the other groups faced. As a result, the values are based on how the system would perform in an ideal sense, as the motor was never attached to the system during testing for the bench inspection that was carried out.

## 4.1 Error Analysis

Throughout the course of the project, with the field of the subject being so technical and demanding in its understanding, errors arose for which were both avoidable and unavoidable. In short summary, there are three significant ways of defining errors that exist; Gross errors, blunders and measurement errors. [11]

Gross errors are the cause of human error, that could be due to the experimenter recording down the wrong value from misinterpreting the data obtained from equipment. Because there was no use of measuring equipment used in the image processing other than the recording of the width of the bike wheel, it can be said that there were no gross errors in this project. As more than one person measured the width of the bike wheel’s frame, with both obtaining the same value, it’s safe to assume that this measurement was correct.

Blunders are similar to that of gross errors, where the user has made a mistake in noting down the wrong value, missing values, or any form of faulty recording. The only recording carried out was already outlined as a possible gross error, due to the circumstances of how the measurement happened, so blunders can be ignored.

Measurement errors have two leading factors that result in the existence of error, which are systematic and random errors. A measurement error typically occurs from the noting of data that deviates from the true value that it should be represented as.

Random errors occur due to external factors that are mostly out of a person’s control, such as temperature, humidity and fluctuation in a voltage supplied to name just some experimental factors. These occur irregularly, with the factor that had a direct impact on being able to accurately record data for the buckle and the ovality of the wheel was due to the lighting in the background. Because the live video from the Raspberry Pi cameras was transformed into greyscale, only white and black effected the intensity values for the output of the histogram.

Systematic errors can be further defined and divided into four specific types of error, because of the variables involved with this class of error.

An instrumental error is that due to the measuring instrument being used that has faulted. Some examples include the hysteresis of magnetic induction, or friction introduced in an experiment. Neither of these cases could have existed within the sub groups part of the project, because only cameras and high level language written on computer software were used, and this type of error could not have existed due to these factors.

Environmental errors link closely to random errors, to do with the device that is measuring for factors such as temperature, humidity or pressure. Within a controlled environment however, these factors can be controlled without being easily influenced by external conditions. There was an environmental error that existed within the project, that being due to the artificial lighting conditions due to the room’s ceiling lights influencing the results.

Observational errors are that due to human error in observing readings from a device or instrument such as evaluating the length of an object using a ruler. Parallax error would be the leading cause of this, due to misalignment of visually being able to read off the correct measurement from the instrument, as well because of the uncertainty that the instrument has too. This could be the case when the stands for the cameras and the bike wheel stand are being setup in their correct positions, giving some slight offset from the cameras not being truly perpendicular to the front and side view of the wheel for measuring the buckle and ovality. This may also be true for having the cameras at the exact midpoint of the window produced at pixel value 320, with the possibility that it is slightly off to 319 for example.

Theoretical errors are created when the theory behind a modelled system goes against what is stated, that leads to consequential errors because of them coming into effect when it is assumed that there should be no faults within the principle of the theory used. The assumptions made that are affected by this are:

* For a standard resolution of 640x480p, 1mm = 10 pixels.
* All wheels used have the same number of spokes, and they are all of uniform size for a given radius.
* The initial setup conditions are to be modelled as ideal, in that the position of the bike wheel is always in the same position.

Based on these assumptions, there exists some theoretical errors within the project, as there are conditions modelled as ideal, which in practice will never be exact, so each time of testing will differ from all previous and future tests.

## 4.2 Multimedia

Throughout the course of the project’s completion, a blog was created and updated over time, discussing how each week the group was able to develop and complete more work after every week had passed. This was an innovative idea, that could be used for possible clients to take an interest in whom might wish look further into the group’s wheel truing system, as well as other project groups of past, present and future. The blog is presented online [12], where it is easily accessible for anyone who might wish to take inspiration from the work set, or alternatively the layout and style of approach in the presentation of the project being completed; as a diary for recording the weekly updates for users to add comments to or query anything they may be unsure of regarding the project.

# 5.0 Discussion

## 5.1 Analysis of Results

Despite the results obtained not being accurate for the true representation of how the system would perform under proper use as a product for users, there was simulated testing that was able to be carried out to give an indication of the technical functionality for how the system should operate.

The algorithm used to measure the buckle and the ovality computed an input that was the position of the wheel’s frame edge, using a histogram function to determine the intensity levels of the pixel value’s location at the boundary edges of the wheel. From then, the average taken from the left and right location of the frame’s edges is subtracted from the origin position of the wheel, to determine the shift in displacement to the left or to the right; the sign of the displacement will determine whether the buckle and/or ovality is to the left (negative) or to the right of the neutral axis (positive). This value is divided by ten to convert the ratio of 10 pixels per 1mm, showing that the uncertainty obtained from the output value to be ± 0.5mm tolerance for the system. When the wheel turns, any anomaly results where the location of the wheel differed dramatically at one value would be ignored, as the program that has been created constantly scans for the location values, and so this is put down to an environmental error in the lighting misconstruing the data that is achieved for the values before and after it, to assertively disregard this minor error.

**2.**The errors that amassed for the duration of the project, on reflection, could be significantly reduced if not eradicated for future use of testing and post development. The sources of the errors that effected the results, which were previously discussed in 4.1 Error Analysis, could be reduced for the following errors.

The lighting conditions could be better controlled, to reduce the environmental error obtained from the background interference of the lighting from outside and the ceiling. If the wheel truing system could be controlled and performed indoors with no sunlight from outside shining through windows, with a box or cover surrounding the wheel’s stand as well as the cameras, then there would be no cause of error due to this, thus completely removing this systematic error. As well as this, white paper or cardboard will oppose the dark created surroundings for the camera to detect, revealing a stark contrast between light and dark when detecting the edges of the wheel. This will reduce the chance of random error, as this will likely reduce the occurrence of anomalies in random high spike values for displacement.

To further reduce systematic errors, the use of lasers must be permitted and passed by health and safety through the department. For the wheel to be perfectly concentric and perpendicular to both sets of cameras, a class II laser system that is used under laboratory conditions in a safe and controlled manner would drastically reduce the occurrence of parallax error and thus observational errors. As the wheel truing system is supposed to be fully self-automated, there should be no human interaction to influence the outcome of the data, so introducing this with a balance to ensure the stands for the wheel and the cameras are both level will help to reduce any systematic errors this way.

Instrumental error from measuring the width of the wheel’s width is a simple fix, by simply using digital Vernier callipers using the internal jaws for the distance between the two edges. Not only will this reduce the uncertainty of using a millimetre rule instead, but also observational error from parallax error by recording an offset measurement.

One final error that should be considered is the focus of the camera’s quality; the current cameras being used are 480p quality. With some small investment, a 1080p camera could be used that would have a better capability of recording fps for the live feed from the cameras when the wheel rotates. Enabling better quality would provide more accurate results for detecting the displacements, and so the instrumental and chance of random error from anomalous results would reduce.

Upon completion of the project set out, looking back at what was first set out for the objectives of the task in hand at the beginning of the dissertation, there was some success from the sub group for the completion of the wheel truing system. The primary objectives stated were to create a program to determine the location of buckle and ovality. Moreover, the system could then send data to the data processing team using serial communication methods(e.g. RS232 or Bluetooth).

Comparing the objectives achieved to these set out prior, it is evident that the group was successful in implementing a way of having a program determine a wheel’s buckle and ovality, but not successful in being able to then use this data to transmit to the sub group 3. It was shown during the bench inspection that through simulation of results, that this was achievable in obtaining a value for both a displacement of the buckle and the ovality at each spoke that passes. It was proven by revealing that at the neutral axis where there was no displacement of the bike wheel’s buckle for camera 1, that the output value achieved was 0. And when the bike wheel was moved manually to the left, the output obtained resulted in negative displacement. This same output was also obtained for the ovality at the same point, providing further evidence that this objective had truly been met successfully.

As there was a five-week time constraint on the project, the protocol for being able to transfer and communicate data between groups one and three was not included. During testing, none of the three sub groups managed to collaborate in synchronising each sub section to work collectively to give a fully working operational system. As a result of this, there was no method of storing data and sending it to the data processing team using serial communication. However, research was carried out in how this would be achieved, in order for the data obtained for the buckle and ovality to be used effectively without results having to be simulated to achieve this.

To send the data to group 3, a protocol would be used to send the data to the processing team. This protocol uses serial communication, for which a RS-232 cable would be used to transfer the data in the form of ASCII, one bit per second. This would be achieved using programming code that converts the stored data location from the code into ASCII. This data would then be sent over to the other team, once this is sent then the other team can use that data to correct the buckle and ovality. The code implemented picks up the serial port device, and using the location of the device found from the RS232 setup, the data can then be coded and sent to the receiving device for them to interpret and make the required changes to the bike wheel.

## 5.2 Limitations

With such a project, the design of the system has little room for design or alterations, especially with such a short time constraint put on each sub group. As such, research was primarily focused on pre-existing products, as these had clearly been proven to work and have a practical functionality sought after by users who had a need for the product. In addition, as this was a new subject area not previously explored or understood by any members of the group, and so this led to there being some delay in communication between the groups, from individuals trying to comprehend their roles and responsibilities to better help the group. There was a lot of waiting time for testing within the groups, as group 3 had claimed to have waited 3 weeks for 3D printing to be completed; an area new to them. Adapting to new software such as visual studio, used for writing code in C++, took time and understanding in order to follow guides on installing the external libraries of OpenCV, in order to be able to use the functions used for it.

Regarding post production limitations, there are areas that restrict the operability of the system, based on design limitations that haven’t been implemented. There unfortunately lacks a method of being able to communicate with group 2, to be able to send across data to be used to adjust the tension in the nipples. There was no method of communication to allow for the data to be transferred between the two group’s Raspberry Pi’s. The ways of resolving this would be to include Bluetooth paring between them, or to introduce serial communication on the University’s Wi-Fi for sending a byte of data per second; this would have been optimal as there was not a frequent amount of numbers sent as data to be transferred consistently.

## 5.3 Applications

A wheel truing stand is a unique engineering design, which is specifically made to cater for bicycle wheels. So because of this, there are limited practical applications for where this concept could be used elsewhere by other companies or industries. With some slight modification of adjustments, the technical setup of the project could later be applied for other wheels such as motorbikes or car wheels. However in practice, the method of being able to change the truancy alignment of a motorbike and car wheel would require a different method of being able to modify the wheel’s alignment, as the wheels are not made using spokes; the wheels used are steel frame, resulting in the whole wheel having to be scrapped and replaced in full.

The benefits of using an automated system for wheel truing is to save the user from the time it would normally take without using the system to have to tighten the nipples of the spokes manually. Of course, the disadvantage against this is the human error that is introduced from having the owner attempt to correct the problem for themselves, resulting in an uneven wheel that makes an unbalanced system of the whole bicycle from not being able to truly know the tension in each spoke without estimating.

## 5.4 Ethical Considerations

Regarding the ethical considerations for this project, there are three key areas that should be highlighted. There are the regulatory considerations, the ethical considerations of the project as large-scale manufacture and sales. Finally, there are the ethical considerations for follow on products for future use.

The group ensured that health and safety protocols of the University in the laboratory environment were followed. Regulations had to be worked with and to, set by law as practice to strictly follow. Standard procedure as well as common sense ensured that all work carried out adhered to rules and regulations in compliance with the University. The regulations that concern this group specifically are The Electrical Equipment (Safety) Regulations 1994 [13], the PUWER (Provision and Use of Work Equipment Regulations 1998) . [14], and the WEEE Regulation (Waste electrical and electronic equipment regulation 2013) acts [15].

The first two regulations in summary, outline and state that all electrical components must be previously tested prior to their usage, as well as complying with the guidelines and specifications set out by the manufacturer, to ensure that it is not only safe but legal to operate with. In addition, every new device or component that is used must have a risk assessment carried out, to test and have anything new introduced approved to be fit for use where it is applicable. Moreover, it must be reassured that the components and devices used are not exposed to water. When considering the disposal of any electrical components used, the proper procedure of removing them is to follow the manufacturer’s required disassembly operations. It is the manufacturer’s responsibility to take back and recycle or dispose of electrical equipment, so if any parts need to be replaced or disposed of potentially due to faults with devices used.

Due to the purpose of this sub group, there is little consideration to be had for any ethical considerations, as well as any factors of sustainable development that could affect the production of the end product. Because of the design limitations of this project, the core of the products sourced is purely computer language code written on a computing interface. Only the initial setup of being able to interact and input the code onto a computer software program was primarily needed in order to program the code applicable to the Raspberry Pi’s used.

Any other considerations such as the software and clamp stand used to support the cameras are all pre-manufactured devices and components already built for immediate usage. All in all, with what little there is to consider, the sustainability needed to be considered for this group is factoring the production methods of the parts used, and the reusability/recycling of the materials and products post production. For the project completed initially, there is little influence to be had over the decision behind the choice of material used in manufacture. For the primary choice of cost efficiency, the casing for the Raspberry Pi and monitor, the mouse and keyboard itself are all plastic that is injection moulded from polypropylene, which is a recyclable thermoplastic.

## 5.5 Commercialisation and Intellectual Property

This product already exists, with pre-existing models being used in the consideration of the final design. For other competitor companies to try and copy pre-emptively off the functionality of this group, would only be taken advantage of in terms of being able to compare the functioning process of the overall system, because there is limited practicality in what can be achieved from the wheel truing system. Only if the group were to develop an automated system of being able to true both wheels on a bike without having to take the tyres off manually, would there be a window of opportunity for rival companies to investigate branching off and expanding the product to the targeted customers. However, this could potentially lead to the company’s demise in taking a risk of introducing new technology, which introduces new unexplored areas that could be exploited for not having the competency or workforce capable of achieving such changes.

# 6.0 Conclusion

## 6.1 Future Work

To be able to carry on in a similar field of work, further modifications and justifications would have to be made to further progress with the development of the wheel truing system. As of now, the system is not fully automatic in that the wheel being trued must be manually placed onto the stand, for one of the other two groups to be able to be able to then manually adjust the tension in the spokes individually. This could be improved upon by introducing another two mechatronic systems for a self-loading robotic arm for changing the wheel on and off the stand, as well as a robot to automatically adjust the tension in the spokes with a torque screwdriver. However, this aspect does not affect the sub group being focused on and gives no benefit towards improving the practical applications of the out performance of this highlighted group.

For looking to give guidance and advice to future groups that wish to repeat this project, the use of lasers to be used for a reference point would be insisted. This would certainly make the initialising of the centre of the origin for the bike wheel better aligned, to reduce the chance of influencing. Prior background knowledge into the purpose of the project

## 6.2 Conclusions

This project taught multiple lessons especially on the importance of planning and time management, the main problem with this project was the time length if the team had longer to understand and learn more about each process then the project would have been a success. Another issue was that due to inexperience with the applications it took longer than expected to learn, especially the image processing and serial communication. The image processing and data communication team researched different ways to send the data, the research was between using the RS232 or using a Bluetooth connection to connect the two raspberry pi’s together. The data processing team decided on Bluetooth as they thought it would be easier to achieve whereas the image processing team focused on the RS232 connection as many examples of code to send and receive data between devices where already given online [16], however both methods could be used and since these where untested, the teams are unsure which method would be the best, due to being unable to test in time.

Communication in the project lacked between some areas in the project, especially when deciding on the protocol, since teams where split many people focused on their part of the project forgetting to inform the rest of the teams, resulting in delays as other teams worked on different ideas. Next time it would be ideal to have meetings between the three project managers assigned to each team, this could help with the communication if the project was to be done a second time.

Overall even though the main objective failed which was to produce a working system which would automatically true a bike wheel, the main parts of the objectives where completed which included a working program for both cameras, the first camera being able to determine the buckle and the second camera detecting the ovality of the bike wheel. The failed objected was serial communication which has been discussed above. The project overall has been a great learning experience for the students, as many skills like time management, teamwork and general engineering skills have been improved and developed. It has been a great experience for the team to work on a challenging project.

# References

|  |  |
| --- | --- |
| [1] | secondnaturecycling, “secondnaturecycling,” wordpress, [Online]. Available: https://secondnaturecycling.wordpress.com/2014/05/16/basic-wheel-truing-and-tensioning. [Accessed 15 03 2019]. |
| [2] | Rethunk, “stackoverflow,” 24 10 2014. [Online]. Available: https://stackoverflow.com/questions/12752168/why-we-should-use-gray-scale-for-image-processing. [Accessed 17 03 2019]. |
| [3] | Sessionscollege, “Sessionscollege,” [Online]. Available: https://documents.sessions.edu/eforms/courseware/coursedocuments/colorcorrection\_cs6/lesson2.html. [Accessed 23 03 2019]. |
| [4] | stackexchange, “stackexchange,” [Online]. Available: https://cs.stackexchange.com/questions/14239/deriving-the-sobel-equations-from-derivatives. |
| [5] | projectrhea, “projectrhea,” projectrhea, [Online]. Available: https://www.projectrhea.org/rhea/index.php/An\_Implementation\_of\_Sobel\_Edge\_Detection. [Accessed 17 03 2019]. |
| [6] | “arduino,” arduino, [Online]. Available: https://forum.arduino.cc/index.php?topic=374116.0. [Accessed 24 03 2019]. |
| [7] | engineersgarage, “engineersgarage,” [Online]. Available: https://www.engineersgarage.com/articles/what-is-rs232. [Accessed 18 03 2019]. |
| [8] | “commfront,” [Online]. Available: https://www.commfront.com/pages/3-easy-steps-to-understand-and-control-your-rs232-devices. [Accessed 24 03 2019]. |
| [9] | “stackexchange,” [Online]. Available: https://raspberrypi.stackexchange.com/questions/58048/send-and-receive-data-in-uart-with-raspberry-pi-in-c-and-wiringpi. [Accessed 25 03 2019]. |
| [10] | “programiz,” [Online]. Available: https://www.programiz.com/cpp-programming/examples/ASCII-value-character. [Accessed 22 03 2019]. |
| [11] | T. Agarwal, “Edgefx,” May 2017. [Online]. Available: https://www.edgefx.in/different-types-of-measurement-errors-and-their-error-calculations/. [Accessed 23 March 2019]. |
| [12] | “Wheel Truing System,” February 2019. [Online]. Available: https://wheeltruingsystem.wordpress.com/. |
| [13] | E. &. I. S. Department for Business, “gov.uk,” 30 08 2012. [Online]. Available: https://www.gov.uk/guidance/electrical-equipment-manufacturers-and-their-responsibilities. [Accessed 24 02 2019]. |
| [14] | Raspberrypi, “Raspberry pi,” RASPBERRY PI FOUNDATION, 2019. [Online]. Available: https://www.raspberrypi.org/documentation/hardware/raspberrypi/conformity.md. [Accessed 20 02 2019]. |
| [15] | HSE, “HSE - waste electrics,” HSE, 2019. [Online]. Available: http://www.hse.gov.uk/waste/waste-electrical.htm. [Accessed 2018 02 22]. |
| [16] | “daniweb,” [Online]. Available: https://www.daniweb.com/programming/software-development/threads/56329/serial-port-communication-using-c#. [Accessed 22 03 2019]. |

# Appendices

## Appendix A

### Camera 1: Detecting the Buckle

1. #include "opencv2/opencv.hpp"//opencv librarys , these are needed to run the code , you must manually add them from https://opencv.org/releases.html
2. #include "opencv2/highgui/highgui.hpp"//opencv lib
3. #include "opencv2/imgproc/imgproc.hpp"//opencv lib
4. #include <iostream>//c++ library
6. **using** **namespace** cv;//opencv namespace
7. **using** **namespace** std;//standard namespace, dont have to use std::cout ect..
9. **void** linehistDisplay(**int** histogram[], **const** **int** width, **const** **char**\* name) //creates and displayes the line to track on histogram
10. {
11. **int** hist[width];
12. **for** (**int** i = 0; i < width; i++)//sets i at 0 , and when i is less than the width (640) then increment the value of i by 1 until it matches the width
13. {
14. hist[i] = histogram[i];//initializes hist to ake it equal the histogram value [1]
15. }
16. // draw the histogram
17. **int** hist\_w = width; **int** hist\_h = 256;//sets the value of the histogram width to "width" and makes the height 256 and this will display and draw the histogram
18. **int** bin\_w = cvRound((**double**)hist\_w / width);//declares bin\_w using cvRound(rounds the foating point number (from width and his\_w) to the nearest integer)
20. // create image for the histogram
21. Mat histImage(hist\_h, hist\_w, CV\_8UC1, Scalar(255, 255, 255));//calls array height,array width ,CV\_8UC1 - 1 channel (greyscale), 255 x3 makes the histogram white
23. //draws the intensity line for histogram
24. **for** (**int** i = 0; i < width; i++)//sames loop as above , sets the pixel line to track the location
25. {
26. line(histImage, Point(bin\_w\*(i), hist\_h), Point(bin\_w\*(i), hist\_h - hist[i]), Scalar(0, 0, 0), 1, 8, 0);//draws the intensity line to track location
27. /\*points to the width and height values which creates the line ,bin\_w(final value from the cvround operation with value of i),
28. , hist\_h - hist[i] takes away original height with width, allows to make horizontal line to track the middle of the screen
29. 1,8,0 mean thickness , linetype\*/
30. }

33. // displays histogram
34. namedWindow(name, CV\_WINDOW\_AUTOSIZE);//creates window Autosize means the image is created in its original size, the original size is 640x480p
35. imshow(name, histImage);//displays histogram
36. }

39. **int** main(**int**, **char**\*\*)
40. {
41. **int** intensity[640];// sets intensity values
42. **int** filtintensity[640];// sets intensity values the filter just makes the lines more clear
43. **int** left\_location;// location values left movement , used to work out the final position
44. **int** right\_location;// location values right movement  , used to work out the final position
45. **int** x\_co;//stores the calculated value
47. printf("Started\n");//stating that the program has started
49. VideoCapture cap(0); // open the default camera
50. **if** (!cap.isOpened())  // check if we succeeded
51. **return** -1;// if not end
53. Mat frame;//Matrix to store frame
54. Mat grey;//Matrix to store the frame as greyscale
56. namedWindow("Grey", 1);//creates a window named grey
57. **for** (;;)// for loop same as using while
58. {
59. cap >> frame; // get a new frame from camera
60. cvtColor(frame, grey, CV\_BGR2GRAY);//converts the frame to greyscale , so only 1 channel instead of 3 channels
61. // Draw the line intensity
62. **int** y = 240; //midpoint vertical point (640x480p) - half of 480
63. **for** (**int** x = 0; x < grey.cols; x++)  //initializes x to equal 0 , then the loop tracks the video columns
64. {
65. intensity[x] = (**int**)grey.at<uchar>(y, x); //original intensity value , (int) allows the value from the uchar(unsigned character) to be turned into a int.
66. }
68. linehistDisplay(intensity, 640, "Original Histogram");  //display window , tracking  intensity values , 640 is related to 640x480p
70. **for** (**int** x = 2; x < (grey.cols - 2); x++)  //this is the exact same as above however the filter is making the histogram lines smooth, better for tracking
71. {
72. filtintensity[x] = ((**int**)grey.at<uchar>(y, x - 2) + (**int**)grey.at<uchar>(y, x - 1) + (**int**)grey.at<uchar>(y, x) + (**int**)grey.at<uchar>(y, x + 1) + (**int**)grey.at<uchar>(y, x + 2)) / 5;  //new intensity calculation
73. //this is the calculation used to smoothen out the lines , the 2 is just used to smooth out the lines this could be changed to lower or higher if needed to make it smoother or more rugged
74. }
76. linehistDisplay(filtintensity, 640, "Filtered Histogram");  //NEW display window , tracking x intensity values , 640 is related to 640x480p
78. left\_location = 0;   //make sure location from left is set at 0
79. right\_location = 640;   //make sure location from right is set at 640
81. **for** (**int** i = 120; i < (grey.cols - 10); i++)  //iteration for the start point of where it is determined across the column lines (allow 10mm from the left boundary of the wheel edge)
82. //120 can be changed depending if we want the histogram to start tracking at a certain height , 120 is a good value for the bike wheel
83. {
84. **if** (filtintensity[i + 10] >(filtintensity[i] + 10))//condition for if the pixel is 10 pixels above the iterative line being scanned
85. //if the filterintensity is 10pixels above the normal threshold then it will start track the location
86. {
87. left\_location = i;   //initialise location\_x to the new value which is the location
88. **break**;  //terminates loop
89. }
90. }
92. **for** (**int** j = 520; j < (grey.cols - 10); j--)  //iteration for the start point of where it is determined across the column lines (allow 10mm from the left boundary of the wheel edge)
93. //120 can be changed depending if we want the histogram to start tracking at a certain height , 120 is a good value for the bike wheel
94. {
95. **if** (filtintensity[j + 10] >(filtintensity[j] + 10))//condition for if the pixel is 10 pixels above the iterative line being scanned
96. //if the filterintensity is 10pixels above the normal threshold then it will start track the location
97. {
98. right\_location = j;   //initialise location to the right to the new value which is the location
99. **break**;  //terminates loop
100. }
101. }
103. printf("location left = %d, location right = %d \n", left\_location, right\_location);   //print to the user the column being detected for highest intensity value's position
104. imshow("Grey", grey);//Displays the video capture in greyscale
105. x\_co = (((left\_location + right\_location)/2)-320) / 10;//calculated the value by taking the locations dividing by 2 and then minusing 320 assume the pixel value is 10 pixel = 1mm
107. printf("X:%d\n", x\_co); //prints out the final value to show how much the wheel has buckled
109. (waitKey(33) >= 0);//waits for a key to be pressed , videos won’t run without this line of code in , as this function delays the capture by so much milliseconds
110. }
111. **return** 0;
112. }
113. //10 pixels is 1mm

## Appendix B

### Camera 2: Detecting Ovality

1. #include "opencv2/opencv.hpp"//opencv librarys , these are needed to run the code , you must manually add them from https://opencv.org/releases.html
2. #include "opencv2/highgui/highgui.hpp"//opencv lib
3. #include "opencv2/imgproc/imgproc.hpp"//opencv lib
4. #include <iostream>//c++ library
6. **using** **namespace** cv;//opencv namespace
7. **using** **namespace** std;//standard namespace, dont have to use std::cout ect..
9. **void** linehistDisplay(**int** histogram[], **const** **int** width, **const** **char**\* name) //creates and displayes the line to track on histogram
10. {
11. **int** hist[width];
12. **for** (**int** i = 0; i < width; i++)//sets i at 0 , and when i is less than the width (640) then increment the value of i by 1 until it matches the width
13. {
14. hist[i] = histogram[i];//initializes hist to ake it equal the histogram value [1]
15. }
16. // draw the histogram
17. **int** hist\_w = width; **int** hist\_h = 256;//sets the value of the histogram width to "width" and makes the height 256 and this will display and draw the histogram
18. **int** bin\_w = cvRound((**double**)hist\_w / width);//declares bin\_w using cvRound(rounds the foating point number (from width and his\_w) to the nearest integer)
20. // create image for the histogram
21. Mat histImage(hist\_h, hist\_w, CV\_8UC1, Scalar(255, 255, 255));//calls array height,array width ,CV\_8UC1 - 1 channel (greyscale), 255 x3 makes the histogram white
23. //draws the intensity line for histogram
24. **for** (**int** i = 0; i < width; i++)//sames loop as above , sets the pixel line to track the location
25. {
26. line(histImage, Point(bin\_w\*(i), hist\_h), Point(bin\_w\*(i), hist\_h - hist[i]), Scalar(0, 0, 0), 1, 8, 0);//draws the intensity line to track location
27. /\*points to the width and height values which creates the line ,bin\_w(final value from the cvround operation with value of i),
28. , hist\_h - hist[i] takes away original height with width, allows to make vertical line to track the middle of the screen
29. 1,8,0 mean thickness , linetype\*/
30. }

33. // displays histogram
34. namedWindow(name, CV\_WINDOW\_AUTOSIZE);//creates window Autosize means the image is created in its original size, the original size is 640x480p
35. imshow(name, histImage);//displays histogram
36. }

39. **int** main(**int**, **char**\*\*)
40. {
41. **int** intensity[640];// sets intensity values
42. **int** filtintensity[640];// sets intensity values and filters makes the lines more clear
43. **int** left\_location;// location values if the wheel moves left  , used to work out the final position
44. **int** right\_location;// location values if the wheel moves left , used to work out the final position
45. **int** x\_co;//stores x and y values
47. printf("Started\n");//stating that the program has started
49. VideoCapture cap(0); // open the default camera
50. **if** (!cap.isOpened())  // check if we succeeded
51. **return** -1;// if not end
53. Mat frame;//Matrix to store frame
54. Mat grey;//Matrix to store the frame as greyscale
56. namedWindow("Grey", 1);//creates a window named grey
57. **for** (;;)// for loop same as using while
58. {
59. cap >> frame; // get a new frame from camera
60. cvtColor(frame, grey, CV\_BGR2GRAY);//converts the frame to greyscale , so only 1 channel instead of 3 channels
61. // Draw the line intensity
62. **int** y = 240; //midpoint vertical point (640x480p) - half of 480
63. **for** (**int** x = 0; x < grey.cols; x++)  //initializes x to equal 0 , then the loop tracks the video columns
64. {
65. intensity[x] = (**int**)grey.at<uchar>(y, x); //original intensity, (int) allows the value from the uchar(unsigned character) to be turned into a int.
66. }
68. linehistDisplay(intensity, 640, "Original Histogram");  //display window , tracking intensity values , 640 is related to 640x480p
70. **for** (**int** x = 2; x < (grey.cols - 2); x++)  //this is the exact same as above however the filter is making the histogram lines smooth, better for tracking
71. {
72. filtintensity[x] = ((**int**)grey.at<uchar>(y, x - 2) + (**int**)grey.at<uchar>(y, x - 1) + (**int**)grey.at<uchar>(y, x) + (**int**)grey.at<uchar>(y, x + 1) + (**int**)grey.at<uchar>(y, x + 2)) / 5;  //new intensity calculation
73. //this is the calculation used to smoothen out the lines , the 2 is just used to smooth out the lines this could be changed to lower or higher if needed to make it smoother or more rugged
74. }
76. linehistDisplay(filtintensity, 640, "Filtered Histogram");  //NEW display window , tracking x intensity values , 640 is related to 640x480p
78. left\_location = 0;   //make sure location from left is set at 0
79. right\_location = 640;   //make sure location from right is set at 640
81. **for** (**int** i = 230; i < (grey.cols - 10); i++)  //iteration for the start point of where it is determined across the column lines (allow 10mm from the left boundary of the wheel edge)
82. //230 can be changed depending if we want the histogram to start tracking at a certain height , 120 is a good value for the bike wheel
83. {
84. **if** (filtintensity[i + 10] >(filtintensity[i] + 10))//condition for if the pixel is 10 pixels above the iterative line being scanned
85. //if the filterintensity is 10pixels above the normal threshold then it will start track the location
86. {
87. left\_location = i;   //initialise left location to the new value of i
88. **break**;  //terminates loop
89. }
90. }
92. **for** (**int** j = 410; j < (grey.cols - 10); j--)  //iteration for the start point of where it is determined across the column lines (allow 10mm from the left boundary of the wheel edge)
93. //number (410) can be changed depending if we want the histogram to start tracking at a certain height
94. {
95. **if** (filtintensity[j + 10] >(filtintensity[j] + 10))//condition for if the pixel is 10 pixels above the iterative line being scanned
96. //if the filterintensity is 10pixels above the normal threshold then it will start track the location
97. {
98. right\_location = j;   //initialise right location to the new value of i
99. **break**;  //terminates loop
100. }
101. }
103. printf("location left = %d, location right = %d \n", left\_location, right\_location);   //print to the user the column being detected for highest intensity value's position
104. imshow("Grey", grey);//Displays the video capture in greyscale
105. x\_co = (((left\_location + right\_location)/2)-320) / 10;//calculated the x value by taking the locations dividing by 2 and then minusing 320 assume the pixel value is 10 pixel = 1mm
107. printf("X:%d\n", x\_co); //prints out the final value
109. (waitKey(33) >= 0);//waits for a key to be pressed , videos wont run without this line of code in , as this function delays the capture by so much milliseconds
110. }
111. **return** 0;
112. }
113. //10 pixels is 1mm

## Appendix C

### ASCII conversion

1. #include<iostream>//c++ libraries
2. **using** **namespace** std;//use functions from the std namespace so you dont have to keep calling namespace
3. **int** main()
4. {
5. **char** c;//stores character
6. cout << "Enter a character : ";//Prompts user to input character
7. cin >> c;//receives the character entered
8. cout << "ASCII value of " << c << " is : " << (**int**)c;//ASCII conversion
9. printf("\n");//new line
10. system("pause");//pauses the command window to view value for testing purposes
11. **return** 0;
12. }

## Appendix D

### Example of Sending messages to raspberry pi

1. #include <stdio.h>//c program library
2. #include <string.h>//string library
3. #include <errno.h>//error number storred in errno
4. #include <wiringPi.h>//to work with raspberry pi
5. #include <wiringSerial.h>//work with raspberry pi
6. **int** main()
7. {
9. **int** fd;//stores fd integer
10. **int** count;//stores count integer
11. **char** ch[100];//arrray of 100 characters stored
13. fd = serialOpen("/dev/ttyA", 9600);//writes string to /dev/ttyA device location , 9600 BAUD rate
15. wiringPiSetup();
16. printf(" -------after wiringPiSetup----\n");// prints to receiving device
17. serialPuts(fd, "TEST"); //Writes string to device /dev/ttyA
18. delay(2000);
19. serialPrintf(fd, ch); //// write uninitialized string to /dev/ttyA reciver device
20. printf("%s", ch);//writes string to std out
21. printf(" -------end main----\n");// prints end main
22. **return** 0;
23. }

## Appendix E

### Project contributions

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Member name** | **Contribution(s)** | **Comments** |
| 1 | Daniel Fox | Abstract, Acknowledgements, 1.0 Introduction, 1.1 Aims and Objectives, 2.0 Research, 3.2 Methods, 6.2 Conclusions, References, Appendix | Spoke in bench inspection and helped deliver presentation.  Completed the buckle and ovality code.  Attempted code for serial communication.  Created the blog and updated it every week  Report – abstract, introduction, research, methods, conclusion, appendix. |
| 2 | Connor Brizell | 4.0 Results, 4.1 Error Analysis, 4.2 Multimedia, 5.0 Discussion, 5.1 Analysis of Results, 5.2 Limitations, 5.3 Applications, 5.4 Ethical Considerations, 5.5 Commercialisation and Intellectual Property, 6.1 Future Work | Spoke in bench inspection and helped deliver presentation.  Completed the buckle and ovality code  Attempted code for serial communication.  Created the poster for the bench inspection.  Report – results, discussion, conclusion |