Lego-LIGO User Guide

Version 1

Dr. Chris North, Daniel Greenhouse

September 24, 2017

0.1 Introduction

This document is intended to provide some explanation of the workings of Lego-LIGO and its various features. The code referred to is the python file: Lego_LIGO_GUI.

0.2 Glossary

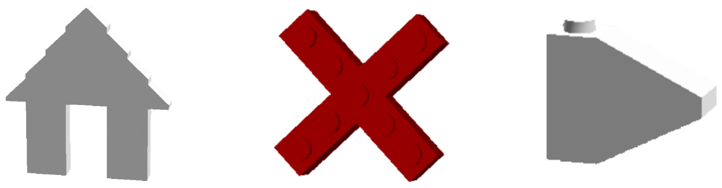
0.2.1 Terms used

Below is a description of some of the terms referred to in both the code and this user guide.

- GUI is the graphical user interface displayed on the touchscreen monitor from which Lego-LIGO can be used
- Wave-carrying beams refer to the long, moving Lego strips which the plastic waves sit on top of
- *zero* position is when the horizontal section of the wave-carrying beam *is thought* to lie between the two rows of red tiles on the floor of Lego-LIGO
- *hard zero* position is when the horizontal section of the wave-carrying beam *actually* lies between the two rows of red tiles on the floor of Lego-LIGO
- Degrees refers to the degrees that the motor has moved through. Due to the design of the motor mechanism this is directly proportional to the displacement of the mirror from the *zero* position
- ‘Manual’ and ‘Self-Movement’ are analogous
- ‘Preset’ and ‘Choose a Movement’ are analogous
- Motor A is mirror X
- Motor B is mirror Y
- A *normal* button in calibration and manual mode is either ‘Forward’ or ‘Backward’
- A *fast* button in calibration and manual mode is either ‘Fast Forward’ or ‘Fast Rewind’

0.2.2 Button functions

The following buttons perform the said functions throughout.
Home, figure 1a, returns to the home screen shown in figure 1.3
Exit, figure 1b, exits the application
Next, figure 1c, loads the next screen of the GUI



(a) Home Button (b) Exit Button (c) Next Button

Figure 1: Images of buttons used throughout the GUI.

0.3 Understanding the motors and the mirror's movements

0.3.1 How the mirrors move

Each motor can be told at what power[1] it should run at. The mirrors move as a consequence of an inputted motor power and inputted time that the motor should run for in sections 13 or 14 of the code; section 13 calculates these whereas section 14 has them inputted from previous functions. The code makes use of python's timer by taking the current time when the motor command is called and then running the motor at the inputted power while the difference between the current time and the time that the motor command is called is less than the inputted time. Then, while the time difference is greater than the inputted time but less than the sum of the inputted time and the so called 'stop_buffer', the motor is run at zero power. This gives the motor time to stop at its current position rather than allowing the motor to *coast* past its desired position.

This method appears to work adequately but is by no means precise and is a source of the constant need for calibration. Should the user encounter issues with the stopping time of the motor they may wish to alter 'stop_buffer' in section 3 *PARAMS* of the code. Also in section 3 *PARAMS* is 'pwrcoef'

[1]The power output is in the python file: BrickPi. The input has no discernable units and so power and speed can be seen as the same in this situation.

which is the coefficient for translating power to the motor degree position. This is measured from the motor itself and may need to be altered should the user see fit.

0.3.2 How the position of the mirrors is tracked

One of the biggest difficulties the user shall encounter when using Lego-LIGO is the discrepancy between the actual motor position and the assumed motor position. As previously mentioned, the degree position of the motor is directly proportional to the displacement of the mirror (providing it stays on the rack) and so guessing the motor's degree position shall suffice.

The motors cannot provide direct feedback to the BrickPi and so it is up to section 11 of the code ('position_tracker') to guess at the position of the motor. Whenever there is a command to move the motor, in sections 13 and 14, the power that each motor runs at and the time that each motor runs for is passed into the 'position_tracker'. This updates the global parameters *MXPos* (motor X position) and *MYPos* (motor Y position) according to the equation

$$\textit{motor power} \times \textit{time the motor runs for} \times \textit{the power coefficient.}$$

0.4 Getting Started

The Raspberry Pi will turn on when power is supplied to it. It requires, at least[2], two power cables; one power cable[3] should go to the Brick Pi and a miniUSB should go to either the touchscreen monitor or the Raspberry Pi. If insufficient power is being supplied to the Pi a yellow lightning bolt shall appear in the top-right corner of the screen.

On start up please double click on the 'runLegoLIGO' application which is available on the desktop. An 'Execute File' window shall load and the user should select 'Execute'. The GUI shall load.

[2]Both the monitor and the Raspberry Pi can take a miniUSB, but the power is shared and so two power cables can power all three.

[3]Type: at least 600mA; 12V; exterior diameter is 5.5mm; interior diameter is 2.5mm

Chapter 1

Using Lego-LIGO

1.1 Welcome Screen

On start up the GUI shall display the ‘Welcome Screen’ as shown in figure 1.1. The cross in the top right of the screen exits the application. The arrow in the bottom right proceeds to the ‘Calibration Screen’ as shown in figure 1.2.

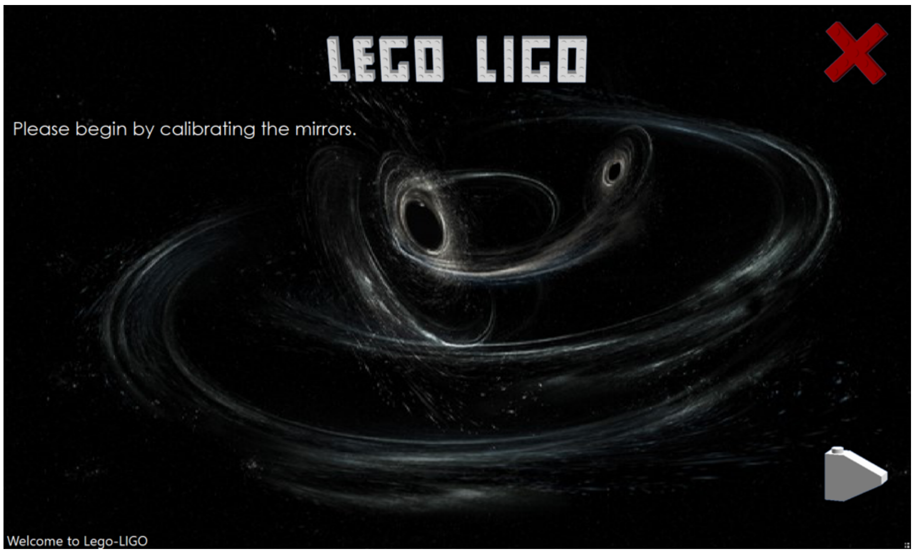


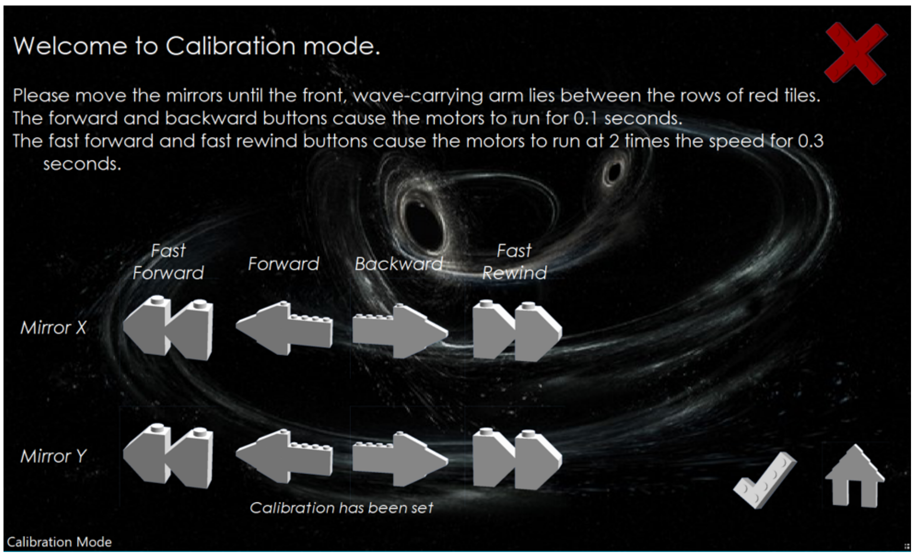
Figure 1.1: Welcome Screen

1.2 Calibration

The calibration mode allows the user to move the mirrors back and forth using the buttons provided. It is shown in figure 1.2. The *fast* buttons move the mirrors quicker and longer by factors specified in section 3 *PARAMS* of the code[1]. The user should use the motor controls to move the front, wave carrying arms until the cross beam lies between the rows of red tiles. This is specified in the on-screen instructions.

The user should then select the 'set calibration' button which is the white tick (in the bottom right-hand corner of figure 1.2, second in from the right). To encourage the user to do so, the home button shall only appear once the 'set calibration' has been pressed.

This sets the global parameters *MXPos* and *MYPos* to be at 0. It is worth noting that this is a very approximate tracking system since it relies on where the motors should be rather than on feedback from the motor as to where they actually are. As a consequence the motors should be regularly calibrated.



Welcome to Calibration mode.

Please move the mirrors until the front, wave-carrying arm lies between the rows of red tiles.
The forward and backward buttons cause the motors to run for 0.1 seconds.
The fast forward and fast rewind buttons cause the motors to run at 2 times the speed for 0.3 seconds.

Calibration has been set

Figure 1.2: Calibration Mode Screen

[1]For more information, see section 2.5.1 of this document

1.3 Home Screen

The 'Home' screen is loaded whenever the home button (shown in figure 1a) is pressed. It is shown in figure 1.3.

The 'Home' screen provides the user with three options: 'Calibration', 'Manual', and 'Preset'. Selecting these shall load the subsequent screen (Calibration is the same as before).

On selection of 'Preset' the mirrors shall return to their *zero* position. This is based on the tracked position and so may be slightly different to the *hard zero* position.

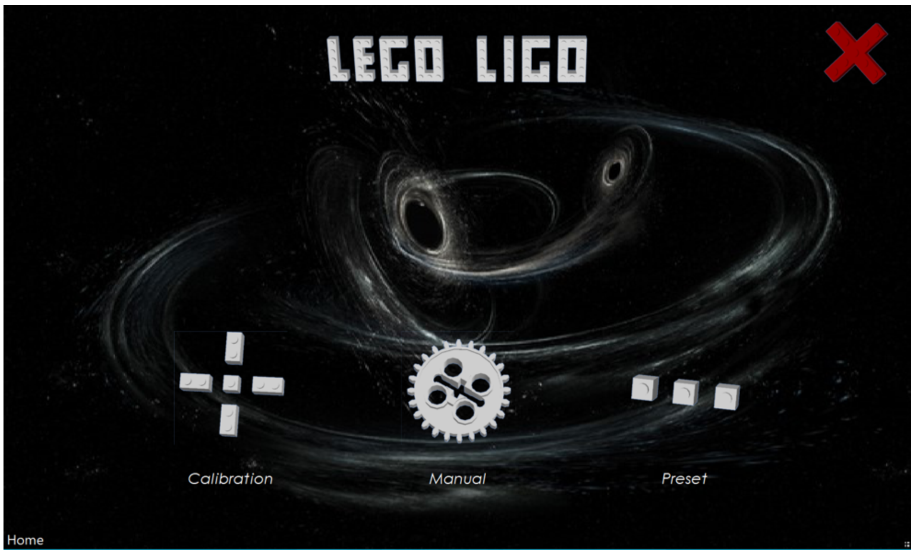


Figure 1.3: Home Screen

1.4 Manual

In 'Manual' mode the user is presented with the same motor movement types as in 'Calibration' mode. It is shown in figure 1.4.

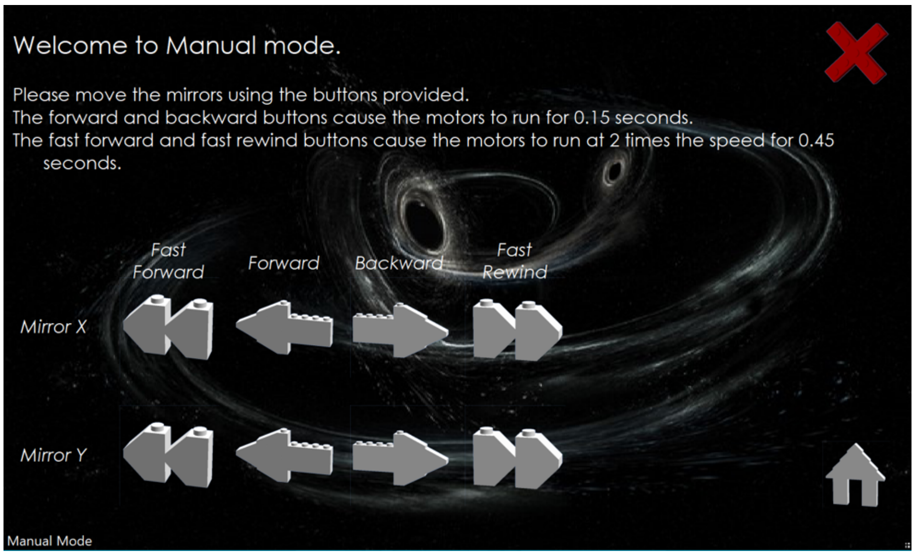
Manual mode is to be used for when the demonstrator wants control of the motors. This can be particularly useful for explaining interference.

The key difference between 'Manual' and 'Calibration' mode is that paramaters in section 3 *PARAMS*[2] scales/alters the time and power run at. The idea behind this is to make 'Manual' mode easier when explaining the interference.

[2] see section 2.5.1 of this document

When the motor has moved too far from the *zero* position (also specified in section 3 *PARAMS*[3]), a message shall display on the screen and the mirrors shall return to their *zero* position. This is to prevent the user from continuously running the motor past its limit which could cause damage to Lego-LIGO.

On exit the mirrors shall return to their *zero* position. This is based on the tracked position and so may be slightly different to the *hard zero* position.


Welcome to Manual mode.

Please move the mirrors using the buttons provided.
The forward and backward buttons cause the motors to run for 0.15 seconds.
The fast forward and fast rewind buttons cause the motors to run at 2 times the speed for 0.45 seconds.

Figure 1.4: Manual Mode Screen

1.5 Preset

The 'Preset' mode offers the user a list of movements. It is shown in figure 1.5. These are the names of the '.csv' files in the 'Test code' folder.

Before selecting a movement the user should take particular care to ensure that the motors are close to their *hard zero* position[4].

The screen has a maximum limit of 10 files and adding more '.csv' files to the 'Test code' folder shall load the 'File Warning' screen shown in figure 1.7.

Choosing a file shall load the corresponding 'Graph' screen.

[3] see section 2.5.1 of this document
[4] There is some margin of error given by 'presetbuffer', see section 2.5.1 of this document

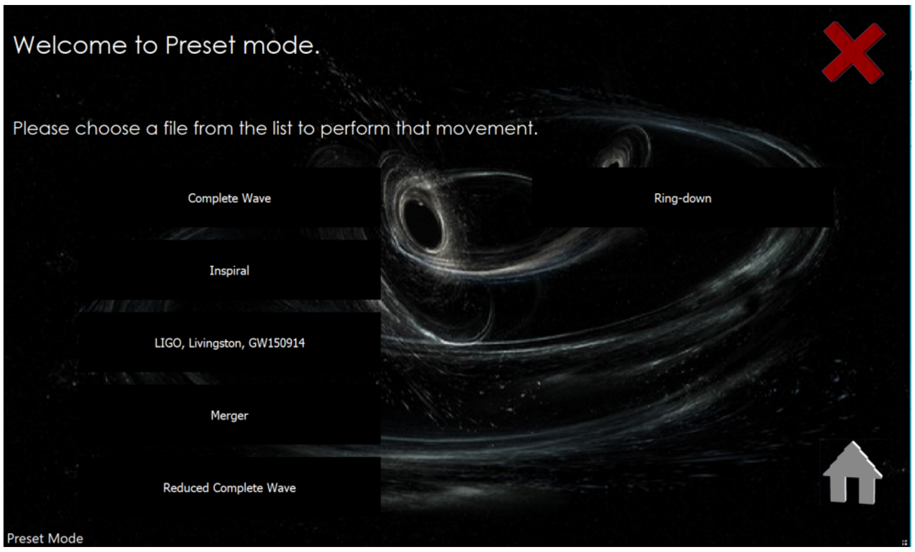


Figure 1.5: Preset Mode Screen

1.6 Graph

Once a movement has been selected from the options in 'Preset' mode, the GUI shall display the graph of the selected movement. An example of this is shown in figure 1.6. The graph has strain on the y-axis. This uses a rough approximation of the change in mirror position and the length of one of Lego-LIGO's arms. Should the user want to find this accurately they should alter 'strain' in section 6 of the code. Time is on the x-axis.

It is here also that the motors receive their commands to move.

The user must wait until the motors have finished performing their selected movement and then returned to their *zero* position. The status bar shall alert the user when the mirrors are returning to their *zero* position.

Once finished, the user should press the 'Home' button located in the bottom-right corner of the screen to return to the 'Home' screen.

1.7 File Warning

When more than 10 files are in the 'Test Code' folder, the screen shown in figure 1.7 shall be displayed when 'Preset' is chosen from the 'Home' screen.

This is due to insufficient space on the screen. See section 2.2 of this document for more information.

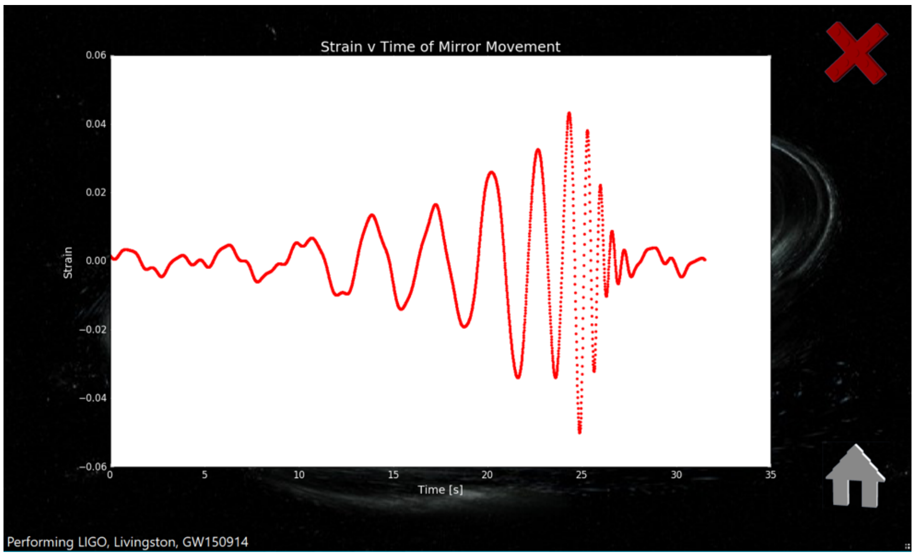

Strain v Time of Mirror Movement
Strain
0.06
0.04
0.02
0.00
−0.02
−0.04
−0.06
0
5
10
15
20
25
30
35
Time [s]
Performing LIGO, Livingston, GW150914

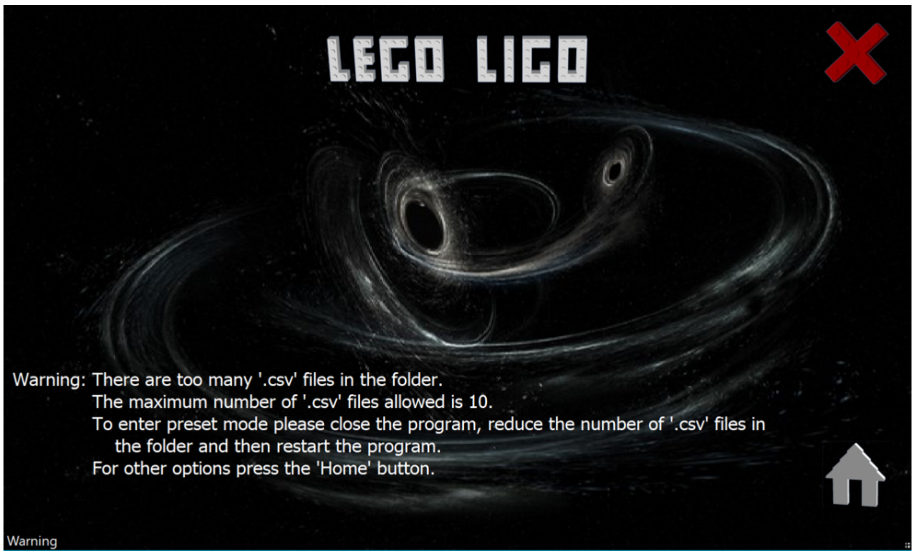
Figure 1.6: Graph Screen

1.8 Speed Warning

The screen shown in figure 1.8 shall be displayed if a file is chosen from the 'Preset' screen that contains a gap between data points which would require the motor to run at an unachievable speed.[5]

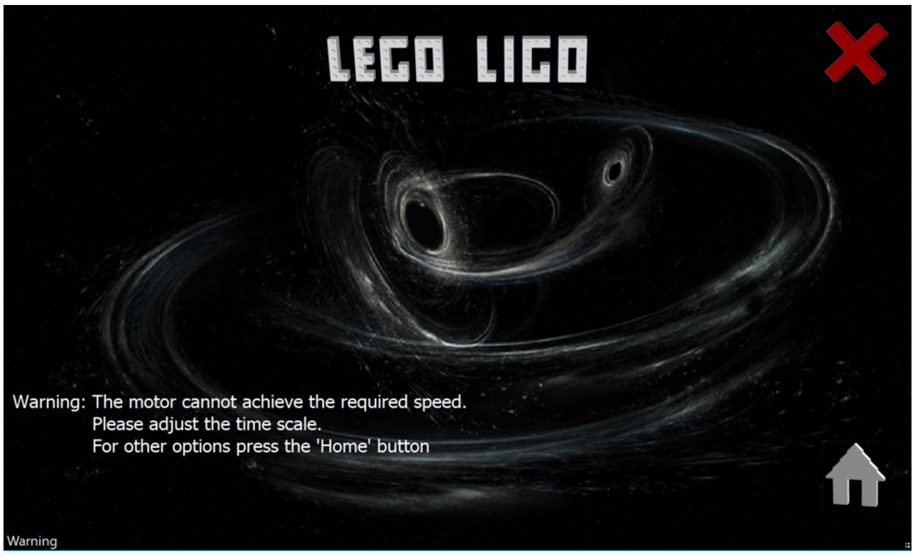
This limiting speed is set manually in section 6 and should be investigated by checking the limit of motors used.

[5]n.b. currently this is not working when in 'Stationary Point' mode. For further information see sections 2.1 and 2.6 of this document.



Warning: There are too many '.csv' files in the folder.
The maximum number of '.csv' files allowed is 10.
To enter preset mode please close the program, reduce the number of '.csv' files in the folder and then restart the program.
For other options press the 'Home' button.

Figure 1.7: File Warning Screen



Warning: The motor cannot achieve the required speed.
Please adjust the time scale.
For other options press the 'Home' button

Figure 1.8: Motor Warning Screen

Chapter 2

Editing Lego-LIGO

When wishing to edit Lego-LIGO on a PC different to the Raspberry Pi, the user should make the following adjustments to the code:

- in section 1 of the code, the 'RP' toggle should be set to 'False'
- in section 3 of the code, a valid file path of the file containing the images should be entered in 'else: IMAGE_LOCATION ='
- in section 3 *PARAMS* of the code, a valid file path of the 'Test Code' file containing all of the '.csv' files should be entered in 'FOLDER'

The user may wish to also alter the window dimensions in section 7 of the code under 'else'.[1]

2.1 Motor Modes

Lego-LIGO can recreate the waveform using the motors in two ways. One finds the stationary points of a wave and runs the motor to these points. The other selects the data point every x[2] seconds. Some waveforms may work better in one mode or the other although most tend to work better when finding the stationary points since this limits the motor's *jerky* movements.

The 'Stationary_points =' toggle allows the user to choose between these modes. 'True' makes the motors run to each stationary point while 'False' makes the motors run to the data point at every x seconds.

[1]RP is True should be left intact unless the touchscreen monitor is to be replaced

[2]x is the data point spacing and can be specified in section 3 *PARAMS* 'data_point_spacing', for more information see section 2.5.1 of this document

2.2 Adding Files

Lego-LIGO can take on additional preset movements. Should the user wish to add additonal preset movements they should do so by taking the file and saving it in the designated ‘Test Code’ folder. The file should be of ‘.csv’ type and have two columns of first time and second strain. The time is scaled by the amount stated in section 3 *PARAMS*, ‘time_scale’. The strain is relative since when the motor performs the file it takes the maximum strain difference and then corresponds this to the maximum number of degrees that the motor can turn through. It is assumed that the first line of the file is the column heading and so this line is skipped when the file is read.

A maximum of ten files are allowed in the ‘Test Code’ folder, exceeding this limit shall bring up the ‘File Warning’ screen, shown in figure 1.7, when the ‘Preset’ mode is selected from the ‘Home’ screen shown in figure 1.3.

2.3 Images

Should the user wish to change an image, the following convention is used.

Use	Name	Type
Background Image	Background	JPG
Title Image	Title	PNG
Home Button Image	Home	PNG
Exit Button Image	Exit	PNG
Calibration Button Image	Calibration	PNG
Manual Button Image	Manual	PNG
Preset Button Image	Preset	PNG
Next Button Image	Next	PNG
Set Calibration Button Image	Set	PNG
Fast Forward Button Image	Fast_Forward	PNG
Forward Button Image	Forward	PNG
Backward Button Image	Backward	PNG
Fast Rewind Button Image	Fast_Rewind	PNG

2.4 GUI

In sections 8 and 9 of the code there is a system for the positioning of buttons and labels. These all scale with the window dimensions and some sections require x and/or y positions when there are several similar buttons to position them accordingly on the screen. If the user wishes to alter these they should read the comments in the various subsections of the code in order to gain understanding of the positioning (it should be fairly self-explanatory from the names of the variables). Sections 8 and 9 do not create the buttons or labels (these are created in user interface setup in section 15) but instead just alter the appearance.

2.5 Troubleshooting

2.5.1 Software

The user may find it beneficial to read through section 0.3 of this document before deciding to make any alterations that they see necessary.

The following are descriptions of some of the quick edits that the user may wish to make by making use of the parameters in section 3 *PARAMS* of the code. Terms from the glossary, section 0.2, are used.

Overall Maximum Degree. To change the maximum degree that the motor can run to from zero, alter 'degree_max'. This should be a direct measurement of the degrees turned through from the *hard zero* position and the limit of Lego-LIGO.

Preset Maximum Degree. To change the maximum degree that the motor can run to while in preset mode, alter 'presetbuffer'. This multiplies the previously mentioned maximum degree by the given amount ('presetbuffer') and so changing the maximum degree shall also affect the distance the mirrors move in preset mode. This shall only affect the preset movement and is especially useful when determining the wavelength etched onto the plastic strips. It helps to ensure that the mirrors don't *crash* and so by using a lower buffer value the user can perform more preset movements between calibrations before the discrepancy between actual motor position and assumed motor position causes a *crash*.

Manual Maximum Degree. To change the maximum degree that the motor can run to while in manual mode, alter 'manualbuffer'. This multiplies the previously mentioned maximum degree by the given amount ('manualbuffer') and so changing the maximum degree shall also affect the maximum distance the mirrors can move in manual mode. This shall only affect the manual movement and is used to help prevent the user from *crashing* the mirrors. Once the limit has been reached the mirrors shall immediately return to their *zero* position.

Mirror Return Speed. To change the speed of the mirrors when returning to their *zero* position, alter 'return_pwr'.

Manual, *Normal* Speed. To change the speed of the mirrors when a *normal* button in manual mode is pressed, alter 'self_movement_pwr'.

Calibration, *Normal* Speed. To change the speed of the mirrors when a *normal* button in calibration mode is pressed, alter 'Calibrate_pwr'.

Calibration and Manual, *Fast* Speed. To change the speed of the *fast* buttons relative to the *normal* buttons in calibration and manual mode, alter 'Fast_power_multiplier'. This simply multiplies the power given to the motor when the *standard* buttons are pressed by the said amount ('Fast_power_multiplier') to give the power given to the motor when the *fast* buttons are pressed.

Time Scale. To change the time multiplier from the '.csv' file to that performed by the motors, alter 'time_scale'.

Manual, *Normal* Time. To change the time that the motors run for with each press of a *normal* button in manual mode, alter 'self_movement_run_time'.

Calibration, *Normal* Time. To change the time that the motors run for with each press of a *normal* button in calibration mode, alter 'Calibrate_run_time'.

Calibration and Manual, *Fast* Time. To change the time that the *fast* buttons cause the motors to run for relative to the *normal* buttons in calibration and manual mode, alter 'Fast_time_multiplier'. This simply multiplies the time that the motors run for when the *standard* buttons are pressed by the said amount ('Fast_time_multiplier') to give the time that the motors run for when the *fast* buttons are pressed.

Power coefficient To change the power coefficient when translating power and time to degree position, alter 'pwrcoef'.

2.5.2 Hardware

Lego-LIGO's main vulnerability is the dislodging of the wave-carrying beam and mirror from the motor. Should the motors run past their limit the mirror's rack and wave-carrying beam's rack could run off the motor's cog.

This should be fixed by entering calibration mode[3]. The user should run the motor forward until both sets of racks come completely off the cogs. There can be difficulty detaching the wave-carrying beam and this should be dealt with by carefully raising it above any hardware limits. Do not worry if the beam breaks since it is easier to repair this once the racks return under the cogs influence. Mirror X causes extra difficulty due to the corner mechanism. Depending on the severity of the separation, the user may find it useful to remove the shortest wave-carrying beam.

The user should begin by rejoining the mirror. The user should ensure the mirror's racks are square to the motor and slowly push it toward the motor until it reaches the cogs. While keeping a slight pressure on the mirror toward the motor, run the motor backward using the 'backward' button. Once the mirror has a good few pins past the cog, the user should attach the wave-carrying beam. This should be done in the same manner, paying particular attention to ensuring the racks are square to the motor. Once the mirror and wave-carrying beam is re-attached, the user should check a successful operation by running the mirror back and forth. The hardware limits should be felt by the wave-carrying beam only. If the mirror interferes with this the user should start again.

Finally, with mirror X only, the user should run the mirror to its forward limit. Once there, the user should replace the shortest wave-carrying arm at its own limit (up against the outer wall) ensuring it fits correctly in the groove. It is sensible to again check the hardware stops are felt by the long, wave-carrying beam only.

For other issues with the breakage of parts of the Lego the computer model[4] should be consulted. This has a complete replication of Lego-LIGO although over time designs may change slightly from it and certain Lego parts may become unavailable. Consequently, the user should use their own discretion in the rebuild.

For the more vulnerable parts, such as the mirror cases and around the motor enclosure, gluing should be considered.

A Raspberry Pi 3 is used with a Dexter Industries modified Raspbian OS stored on a micro SD card. A Dexter Industries BrickPi + hat is used alongside Lego Mindstorms Large motors. The monitor is 'The Official Raspberry Pi 7" Touchscreen' with a 800 x 480 pixel display.

[3]manual mode could be used but typically calibration mode uses smaller mirror movements with each press and shall not cause the mirrors to reset their positions once degree max has been reached

[4]This is the Lego Digital Designer LXF Model File (.lxf), entitled 'Lego-LIGO Model'

2.6 Furture Edits

Future users may wish to provide the following edits:

- fix the bug in the resetting
- Add a stop button to the preset movement
- Plot the graph as the motor is moving.
- Provide a motor speed warning when the motor is in stationary points mode
- Change the plastic wave strips with sinusoidal waves etched on for *opacity gradient* waves and make use of Lego's Light sensor and an LED to provide real-time feedback of the interference.

Chapter 3

Contacts

For any further queries please feel free to contact:

Daniel Greenhouse
University of Birmingham
dmg697@student.bham.ac.uk