WM164

Smart Solutions Assessment 3

U1921983

___

TABLE OF CONTENTS

(1) ANALYSIS OF THE PROBLEM ... 3

(I) PROBLEM DEFINITION ... 3

(II) STAKEHOLDERS ... 3

(III) RESEARCH THE PROBLEM ... 4

(IV) SPECIFIY THE PROPOSED SOLUTION ... 5

(2) DESIGN OF THE SOLUTION ... 7

(I) DECOMPOSE THE PROBLEM ... 7

(II) DESCRIBE THE SOLUTION ... 9

(III) DESCRIBE THE APPROACH TO TESTING ... 10

(4) EVALUATION ... 11

(I) DESCRIBE THE FINAL PRODUCT ... 11

(II) MAINTENANCE AND DEVELOPMENT ... 13

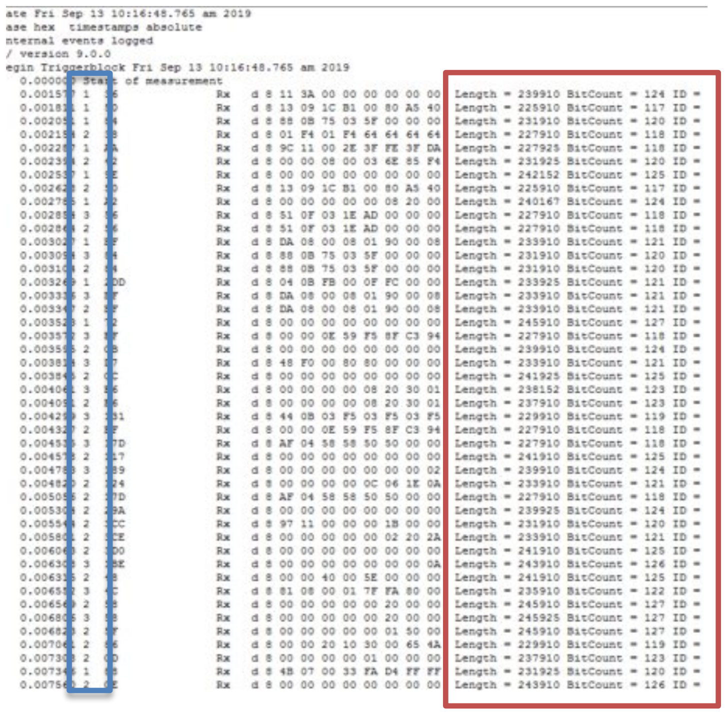
REFERENCES ... 13

(1) ANALYSIS OF THE PROBLEM

(I) PROBLEM DEFINITION

My program is going to involve the analysis and manipulation of CAN logs. A Controller Area Network (CAN bus) is a robust vehicle bus standard designed to allow microcontrollers and devices to communicate with each other's applications without a host computer (Kawahara et al., 2014). CAN logs are an event driven protocol and therefore do not take data at consistent time intervals. CAN logs taken from vehicles are a collection of frames as they appear on the bus, recorded in a time sequential order (Siti-Farhana et al., 2019).

The problem that we currently face is that the buses are recording data 24 hours a day, 7 days a week. This leaves a significant number of hours of data being recorded but, in many cases, not used by the user. CAN logs can only be replayed in real time from start to finish using the current CANaylser application that is available to JLR employees. The problem associated with this is that if the log file is recorded over a long period and there are periods of time with CAN bus inactivity within it, the inactivity periods must be replayed within the tool as there is no option to remove them. My solution will be more user friendly using a GUI. It will allow for the periods of inactivity to be removed which will then save time. My program will analyse the file and return the time stamps/duration of CAN bus inactivity, using 2D arrays not pandas. It will request user input for start and finish times of the window of the log file that is to be extracted allowing analysis in the graphical tool. It will offset the time stamps of the recorded frames such that the chosen window starts from time 0, thus allowing the tool to start replaying the data as soon as requested to start. It will support an option to parse the log file to remove unwanted data as shown in the red outline in Figure 1. It will allow the user to save the processed log file in a format that is compatible with the graphical analysis tool. All features will be implemented into a GUI using tkinter with check boxes and input boxes.



```
ate Fri Sep 13 10:16:48.765 am 2019
ase hex  timestamps absolute
nternal events logged
/ version 9.0.0
egin Triggerblock Fri Sep 13 10:16:48.765 am 2019
   0.00000  Sta  of measurement
   0.00157  1    6          Rx   d 8 11 3A 00 00 00 00 00 00  Length = 239910 BitCount = 124 ID =
   0.00181  1    D          Rx   d 8 13 09 1C B1 00 80 A5 40  Length = 225910 BitCount = 117 ID =
   0.00205  1    4          Rx   d 8 88 0B 75 03 5F 00 00 00  Length = 231910 BitCount = 120 ID =
   0.00215  2    B          Rx   d 8 01 F4 01 F4 64 64 64 64  Length = 227910 BitCount = 118 ID =
   0.00228  1    A          Rx   d 8 9C 11 00 2E 3F FE 3F DA  Length = 227925 BitCount = 118 ID =
   0.00239  2    2          Rx   d 8 00 00 08 00 03 6E 85 F4  Length = 231925 BitCount = 120 ID =
   0.00255  1    E          Rx   d 8 00 00 00 00 00 00 00 00  Length = 242152 BitCount = 125 ID =
   0.00262  2    D          Rx   d 8 13 09 1C B1 00 80 A5 40  Length = 225910 BitCount = 117 ID =
   0.00278  1    2          Rx   d 8 00 00 00 00 00 08 20 00  Length = 240167 BitCount = 124 ID =
   0.00285  3    6          Rx   d 8 51 0F 03 1E AD 00 00 00  Length = 227910 BitCount = 118 ID =
   0.00286  2    6          Rx   d 8 51 0F 03 1E AD 00 00 00  Length = 227910 BitCount = 118 ID =
   0.00302  1    F          Rx   d 8 DA 08 00 08 01 90 00 08  Length = 233910 BitCount = 121 ID =
   0.00309  3    4          Rx   d 8 88 0B 75 03 5F 00 00 00  Length = 231910 BitCount = 120 ID =
   0.00310  2    4          Rx   d 8 88 0B 75 03 5F 00 00 00  Length = 231910 BitCount = 120 ID =
   0.00326  1    DD         Rx   d 8 04 0B FB 00 0F FC 00 00  Length = 233925 BitCount = 121 ID =
   0.00333  3    F          Rx   d 8 DA 08 00 08 01 90 00 08  Length = 233910 BitCount = 121 ID =
   0.00334  2    F          Rx   d 8 DA 08 00 08 01 90 00 08  Length = 233910 BitCount = 121 ID =
   0.00352  1    2          Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID =
   0.00357  3    F          Rx   d 8 00 00 0E 59 F5 8F C3 94  Length = 227910 BitCount = 118 ID =
   0.00359  2    B          Rx   d 8 00 00 00 00 00 00 00 00  Length = 239910 BitCount = 124 ID =
   0.00381  3    7          Rx   d 8 48 F0 00 80 80 00 00 00  Length = 233910 BitCount = 121 ID =
   0.00384  2    C          Rx   d 8 00 00 00 00 00 00 00 00  Length = 241925 BitCount = 125 ID =
   0.00406  3    6          Rx   d 8 00 00 00 00 08 20 30 01  Length = 238152 BitCount = 123 ID =
   0.00409  2    6          Rx   d 8 00 00 00 00 08 20 30 01  Length = 237910 BitCount = 123 ID =
   0.00429  3    31         Rx   d 8 44 0B 03 F5 03 F5 03 F5  Length = 229910 BitCount = 119 ID =
   0.00432  2    F          Rx   d 8 00 00 0E 59 F5 8F C3 94  Length = 227910 BitCount = 118 ID =
   0.00453  3    7D         Rx   d 8 AF 04 58 58 50 50 00 00  Length = 227910 BitCount = 118 ID =
   0.00457  2    17         Rx   d 8 00 00 00 00 00 00 00 00  Length = 241910 BitCount = 125 ID =
   0.00478  3    89         Rx   d 8 00 00 00 00 00 00 00 02  Length = 239910 BitCount = 124 ID =
   0.00482  2    24         Rx   d 8 00 00 00 00 0C 06 1E 0A  Length = 233910 BitCount = 121 ID =
   0.00505  2    7D         Rx   d 8 AF 04 58 58 50 50 00 00  Length = 227910 BitCount = 118 ID =
   0.00530  2    9A         Rx   d 8 00 00 00 00 00 00 00 00  Length = 239925 BitCount = 124 ID =
   0.00554  2    CC         Rx   d 8 97 11 00 00 00 1B 00 00  Length = 231910 BitCount = 120 ID =
   0.00580  2    CE         Rx   d 8 00 00 00 00 00 02 20 2A  Length = 233910 BitCount = 121 ID =
   0.00606  2    D0         Rx   d 8 00 00 00 00 00 00 00 00  Length = 241910 BitCount = 125 ID =
   0.00630  3    BE         Rx   d 8 00 00 00 00 00 00 00 0A  Length = 243910 BitCount = 126 ID =
   0.00631  2    8          Rx   d 8 00 00 40 00 5E 00 00 00  Length = 241910 BitCount = 125 ID =
   0.00655  3    C          Rx   d 8 81 08 00 01 7F FA 80 00  Length = 235910 BitCount = 122 ID =
   0.00656  2    8          Rx   d 8 00 00 00 00 00 20 00 00  Length = 245910 BitCount = 127 ID =
   0.00680  3    8          Rx   d 8 00 00 00 00 00 20 00 00  Length = 245925 BitCount = 127 ID =
   0.00682  2    F          Rx   d 8 00 00 00 00 00 01 50 00  Length = 245910 BitCount = 127 ID =
   0.00706  2    6          Rx   d 8 00 00 20 10 30 00 65 4A  Length = 229910 BitCount = 119 ID =
   0.00730  2    D          Rx   d 8 00 00 00 00 01 00 00 00  Length = 237910 BitCount = 123 ID =
   0.00734  1    8          Rx   d 8 4B 07 00 33 FA D4 FF FF  Length = 231925 BitCount = 120 ID =
   0.00756  2    E          Rx   d 8 00 00 00 00 00 00 00 00  Length = 243910 BitCount = 126 ID =
```

The program will also allow the user to change the CAN channels highlighted in blue, saving time when uploading the program CANalyser.

Using this program will reduce processing time significantly so data analysts and software engineers can access the necessary data and visuals as soon as possible at no extra cost to the business.

(II) STAKEHOLDERS

My program is aimed for use in the workplace or working from home. The users are primarily employees of Jaguar Land Rover; however, it could be used by students in WMG to teach about CAN logs. JLR need to analyse and deal with this data at a high rate but cannot do so if the wait time to process this data is the same time it has taken to collect it.

(III) RESEARCH THE PROBLEM

My program will need the use of loops, selection statements and functions to allow it to work at the highest level of functionality (Browning and Alchin, 2019). Loops will help me iterate through the large amounts of data that I will be processing. Using loops makes my program more efficient and reduces the length of my code while also making it more reader friendly (Milliken, 2020). Within the loops I can then use selection statements to carry out the necessary actions required by the user.

I will use lists and arrays to store the data of the CAN logs, giving my program access to each digit/letter of data in the document. I can then iterate through each line of data using nested loops.

I will be using a Graphical User Interface (GUI) to provide a user-friendly experience and allow the user to carry out their task with ease and know exactly how to do that task. The GUI will have a login page to only allow access to required users and then a main screen. For the login I will use a SHA-1 hash to securely store the passwords. I can then compare hashes when the user inputs a password instead of plain text passwords. Hashing makes the login secure because you cannot decrypt a hash and go back to plain text from it (Ntantogian et al., 2019). I will be using tkinter for my GUI as in many cases it is already installed in python and is a readily available library (Chaudhary, 2013). The GUI allows me to have drop down boxes, buttons and interactive file I/O giving the program a user-friendly experience.

(IV) SPECIFIY THE PROPOSED SOLUTION

I will need a laptop or PC and the software needed is Python 3.8 with the anaconda library. I will also need a keyboard and mouse to navigate the menu. The version of anaconda should also be the most recent version to keep all aspects up to date.

I have decided to make these my requirements as it is easy to acquire these items or install the software and the majority of my stakeholders will be able to get these easily. This means that it saves time trying to convert the program into different formats. The computers will not need a network connection as the program will be standalone and will not save/upload/download anything to/from the internet.

The limitations I have will be the fact that python doesn't cope well with high intensity programs, however my program will have simple textures so it should be suitable. Another limitation would be that my program requires the stakeholders to have software installed and they cannot just run it straight away, but since this is a simple piece of software to install it is not a large issue.

I am not using any copyrighted material and therefore, if I wanted to could sell the software to parties similar to my stakeholders.

In the future I would try to compile the program into a package.

1. Aesthetic objectives
    a. The window size is 800x300 pixels and the window should be non-resizable
    b. The colour scheme is a white background
    c. The JLR logo will be placed at the top of the window
    d. Black text
    e. The font used will be the standard tkinter font and will be consistent throughout the software
    f. Menu Options
        i. These have white/black text
        ii. They will be buttons
        iii. The buttons will play a click animation when clicked
        iv. They should be easy to read.
        v. 2 labels – Username and password
    g. Data screen
        i. The screen should be shown identically to the main login screen, but the labels will be different
2. Inputs
    a. The user will control everything using a keyboard and mouse therefore they will be needed
    b. Screens are navigated and actions are carried out by pressing enter when on-screen buttons are selected
    c. If "exit" is pressed the program quits.
3. Processing

   a. The program will check the username and password input against the saved details in the login file.

      i. If they match then the main menu will be opened, if they don't match it will not allow access and if the boxes are empty it will say "input a password".

   b. It will be checking if any keys have been pressed and if any of the keys are pressed it will respond accordingly

   c. Key presses

      i. Key presses should only be used to input text or times

   d. Calculations

      i. The program will take the input times from the user and find the closest time to those in the input file.

      ii. The program will, if asked, parse the unnecessary data.

   e. CAN channels

      i. Code will allow user to change what CAN channels are being used

4. Output

   a. The main output will be the computer screen that the player is using. The screen will be used at all times and won't be resizable due to the dimensions of my backgrounds.

   b. The program will output a file that has been formatted to the user's requirements.

These requirements will form most of my design and I shall base most of my design on these. However, this may change as the program is designed and developed. My stakeholders thought the success criteria were accurate to what they were looking for and that they would be happy with the outcome if it were along with these requirement

(2) DESIGN OF THE SOLUTION

(I) DECOMPOSE THE PROBLEM

I am going to be coding in Python, as it is an easily accessible language that I can use at work, home and university. It has many libraries that I will be able to use to help me in my development. It also has the functionality to incorporate Tkinter, which I will use, to make a GUI and actual program interface. During my design I will be using pseudocode/flowcharts and top down diagrams (Kapil, 2019).

My program can be broken easily into different functions and procedures. I can use these to decide which bits of the code are executed at certain times. I will be repeating some code multiple times so creating functions will help me reduce overall size of the code

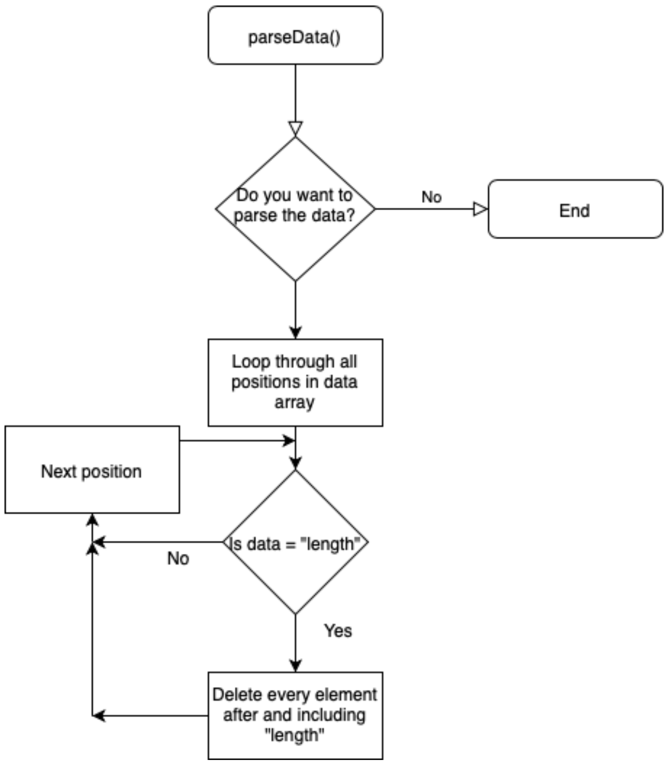
For example, I can break down my program to small manageable sections. Then I can use pseudocode and flowcharts to plan the development.

```
def getData
    file = "raw data"
    for line in file:
    array = []
    array = line split by spaces
    if "Statistic:" in array:
        continue
    elif "CAN" in array:
        continue
    elif "ErrorFrame" in array
        continue
    else:
        append array to data

def parseData():
    x = input("Do you want to parse the unnecessary data? Y/N")
    if x == "Y":
    for i in range(0, len(data))
        for j in range(0, len(data[i]))
            if data[i][j] == "Length":
                del every position after data[i][j]
```

```
                    break
```

The pseudocode is in broken English/ basic code to make it easier to understand and plan. This can then be translated into python code with the correct syntax.


parseData()
Do you want to parse the data?
No
End
Loop through all positions in data array
Next position
Is data = "length"
No
Yes
Delete every element after and including "length"

The same can be done with a flowchart.

(II) DESCRIBE THE SOLUTION

I will consider how the different parts of my code will work so I will need to visualise how myself and the stakeholders want aspects to look like. The program will be split into 2 menus.

Functions I will use:

Func – ‘getData’ – This function is used to open the chosen file and format it so it can be manipulated as requested.

Func – ‘parseData’ – This function is used to remove the unnecessary parts of data but only if the user ticks the box is it called.

Func – ‘dataManip’ - function that returns the closest timestamps to user input numbers for start and end of desired section, this function takes two parameters: the start and end time input by the user.

Func – ‘tkLogin’ – this function sets the login window up using tkinter and calls the ‘mainloop’ to open a window and allow the user to login or register.

Func – 'register' – is called when the user presses register, it requires the master user and password to be entered to register a new user.

Func – 'loginTest' – This function compares the user input username and password with the saved versions to see if it matches and then decides whether access is allowed.

Func – 'mainMenu' – This function opens the mainMenu after the user has input their username and password and allows the selecting of a file to format and output.

All of my functions can operate without the running of another. This allows independence in the program and seamless modification and no function relies on another (Milliken, 2020).

My program will provide a modular solution in the form of a graphical user interface.

(III) DESCRIBE THE APPROACH TO TESTING

(a) Identify the test data to be used during the iterative development and post development phases and justify the choice of this test data.

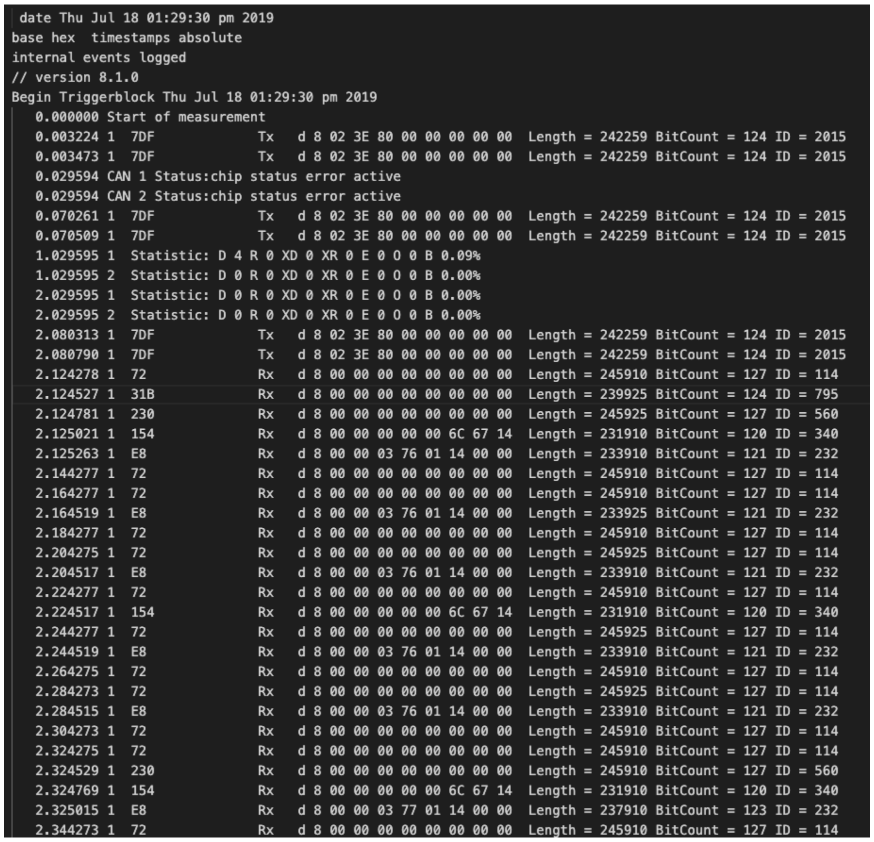
Testing the program could prove difficult as the data being used is always extremely large and using a trace table to follow it would be inefficient and inaccurate. Instead of doing tests for each variable I will try and test the function of the program so that each input produces the expected outcome and therefore shows that the code works in the intended way. I need to consider the fact that the user may press multiple keys at once and must decide which keys are the most important.

Firstly, I can alpha test the program by testing the variables with testing tables as far as I can without wasting too much time and putting too many resources into it. I can then perform black box testing, where the user does not know what is being testing. I can give it to a stakeholder and ask them to perform tasks to see if it works as they want and how I intended. The most important part of the program is how it works with students and the users, therefore black box testing would be an important part of the testing phase and it would show how well the program and user interact with each other (Kapil, 2019).

Using black box testing will be free from any bias that I may have that could mean certain bugs might not be found. However, there will be other times where it is easier for me to test the system in white box testing, as I know more what are the weaker parts of the system that I can test more rigourously and stress test the variables to expose weaknesses (Kelly and Safari, 2019).

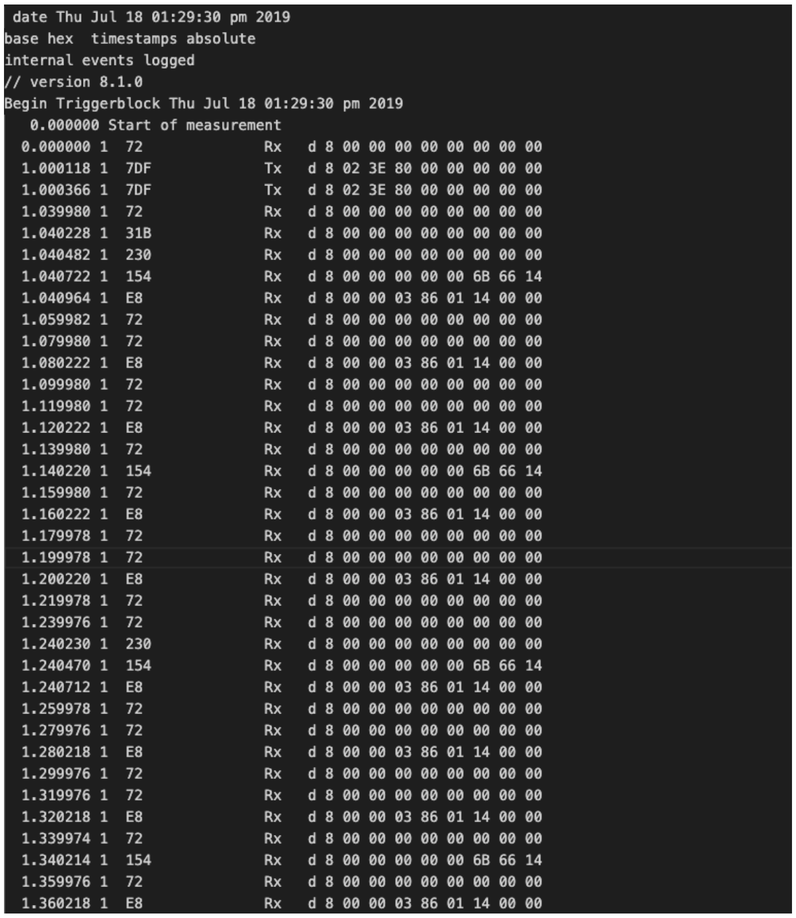
I can get feedback from the black box testing by using a questionnaire to gather information from the users to see what they found useful or possibly elements that were counterproductive.

Candidate Name: Candidate Number:



```
 date Thu Jul 18 01:29:30 pm 2019
base hex  timestamps absolute
internal events logged
// version 8.1.0
Begin Triggerblock Thu Jul 18 01:29:30 pm 2019
   0.000000 Start of measurement
   0.003224 1  7DF             Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   0.003473 1  7DF             Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   0.029594 CAN 1 Status:chip status error active
   0.029594 CAN 2 Status:chip status error active
   0.070261 1  7DF             Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   0.070509 1  7DF             Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   1.029595 1  Statistic: D 4 R 0 XD 0 XR 0 E 0 O 0 B 0.09%
   1.029595 2  Statistic: D 0 R 0 XD 0 XR 0 E 0 O 0 B 0.00%
   2.029595 1  Statistic: D 0 R 0 XD 0 XR 0 E 0 O 0 B 0.00%
   2.029595 2  Statistic: D 0 R 0 XD 0 XR 0 E 0 O 0 B 0.00%
   2.080313 1  7DF             Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   2.080790 1  7DF             Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   2.124278 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.124527 1  31B             Rx   d 8 00 00 00 00 00 00 00 00  Length = 239925 BitCount = 124 ID = 795
   2.124781 1  230             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245925 BitCount = 127 ID = 560
   2.125021 1  154             Rx   d 8 00 00 00 00 00 6C 67 14  Length = 231910 BitCount = 120 ID = 340
   2.125263 1  E8              Rx   d 8 00 00 03 76 01 14 00 00  Length = 233910 BitCount = 121 ID = 232
   2.144277 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.164277 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.164519 1  E8              Rx   d 8 00 00 03 76 01 14 00 00  Length = 233925 BitCount = 121 ID = 232
   2.184277 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.204275 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245925 BitCount = 127 ID = 114
   2.204517 1  E8              Rx   d 8 00 00 03 76 01 14 00 00  Length = 233910 BitCount = 121 ID = 232
   2.224277 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.224517 1  154             Rx   d 8 00 00 00 00 00 6C 67 14  Length = 231910 BitCount = 120 ID = 340
   2.244277 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245925 BitCount = 127 ID = 114
   2.244519 1  E8              Rx   d 8 00 00 03 76 01 14 00 00  Length = 233910 BitCount = 121 ID = 232
   2.264275 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.284273 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245925 BitCount = 127 ID = 114
   2.284515 1  E8              Rx   d 8 00 00 03 76 01 14 00 00  Length = 233910 BitCount = 121 ID = 232
   2.304273 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.324275 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.324529 1  230             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 560
   2.324769 1  154             Rx   d 8 00 00 00 00 00 6C 67 14  Length = 231910 BitCount = 120 ID = 340
   2.325015 1  E8              Rx   d 8 00 00 03 77 01 14 00 00  Length = 237910 BitCount = 123 ID = 232
   2.344273 1  72              Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
```

To test my software, I will run the code and carry out the functions as you normally would. After inputting my desired timestamps and whether or not I would like to parse the unnecessary data then I would compare my original CAN log to the new converted one.

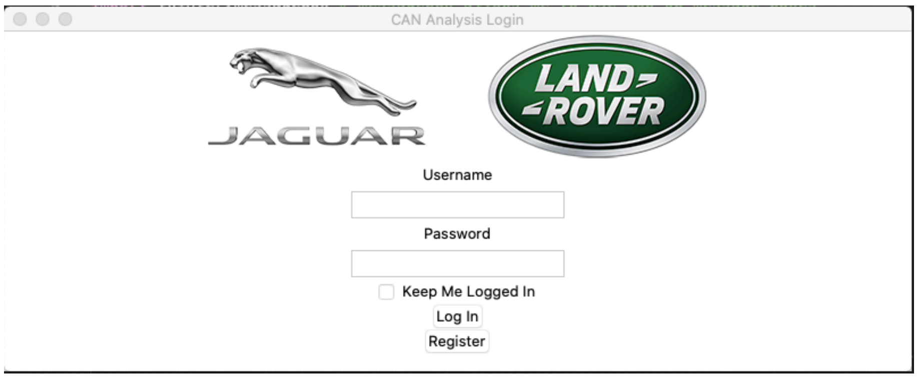


```
 date Thu Jul 18 01:29:30 pm 2019
base hex  timestamps absolute
internal events logged
// version 8.1.0
Begin Triggerblock Thu Jul 18 01:29:30 pm 2019
   0.000000 Start of measurement
  0.000000 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.000118 1  7DF             Tx   d 8 02 3E 80 00 00 00 00 00
  1.000366 1  7DF             Tx   d 8 02 3E 80 00 00 00 00 00
  1.039980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.040228 1  31B             Rx   d 8 00 00 00 00 00 00 00 00
  1.040482 1  230             Rx   d 8 00 00 00 00 00 00 00 00
  1.040722 1  154             Rx   d 8 00 00 00 00 00 6B 66 14
  1.040964 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.059982 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.079980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.080222 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.099980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.119980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.120222 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.139980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.140220 1  154             Rx   d 8 00 00 00 00 00 6B 66 14
  1.159980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.160222 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.179978 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.199978 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.200220 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.219978 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.239976 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.240230 1  230             Rx   d 8 00 00 00 00 00 00 00 00
  1.240470 1  154             Rx   d 8 00 00 00 00 00 6B 66 14
  1.240712 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.259978 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.279976 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.280218 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.299976 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.319976 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.320218 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.339974 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.340214 1  154             Rx   d 8 00 00 00 00 00 6B 66 14
  1.359976 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.360218 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
```

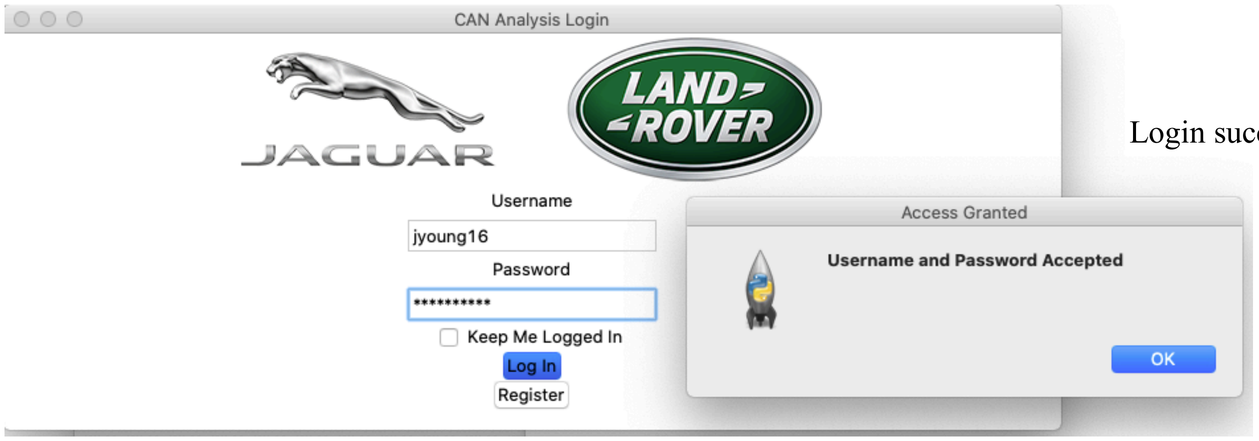
In this case I chose timestamps 22 and 23.4. The program has offset the timestamps correctly to 0 and 1.36 (not 1.4 as 1.36 is the closest timestamp available). It has also parsed the data so everything right of "Length" is removed.

(3) EVALUATION

(I) DESCRIBE THE FINAL PRODUCT



tkLogin

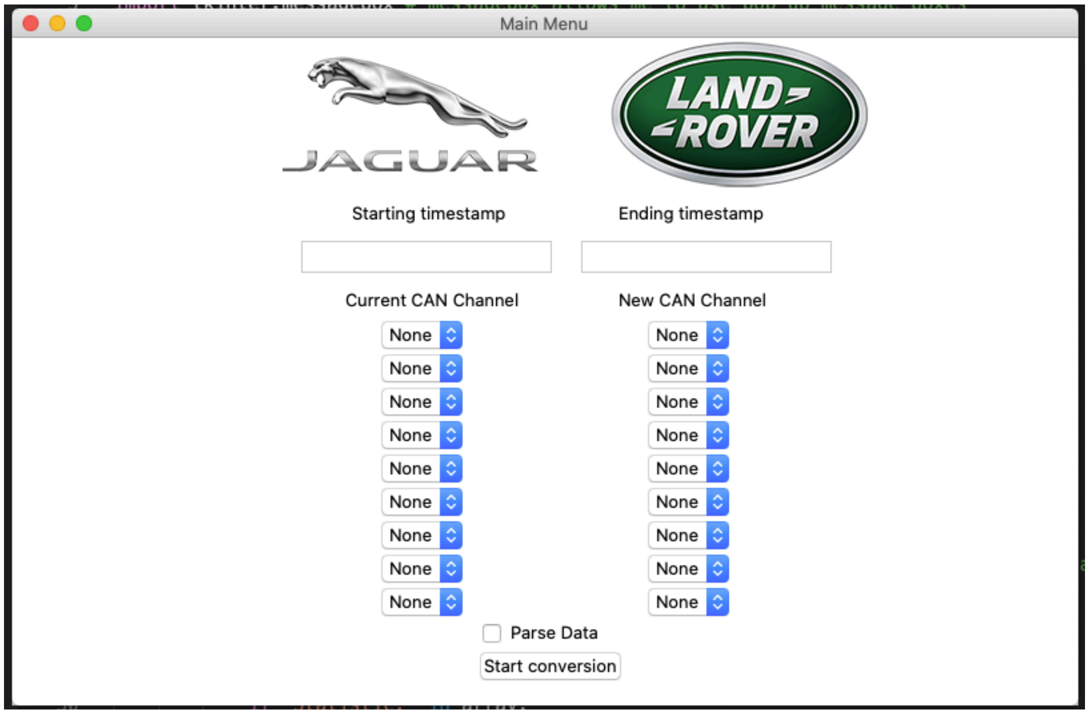


Username: jyoung16

Password: **********

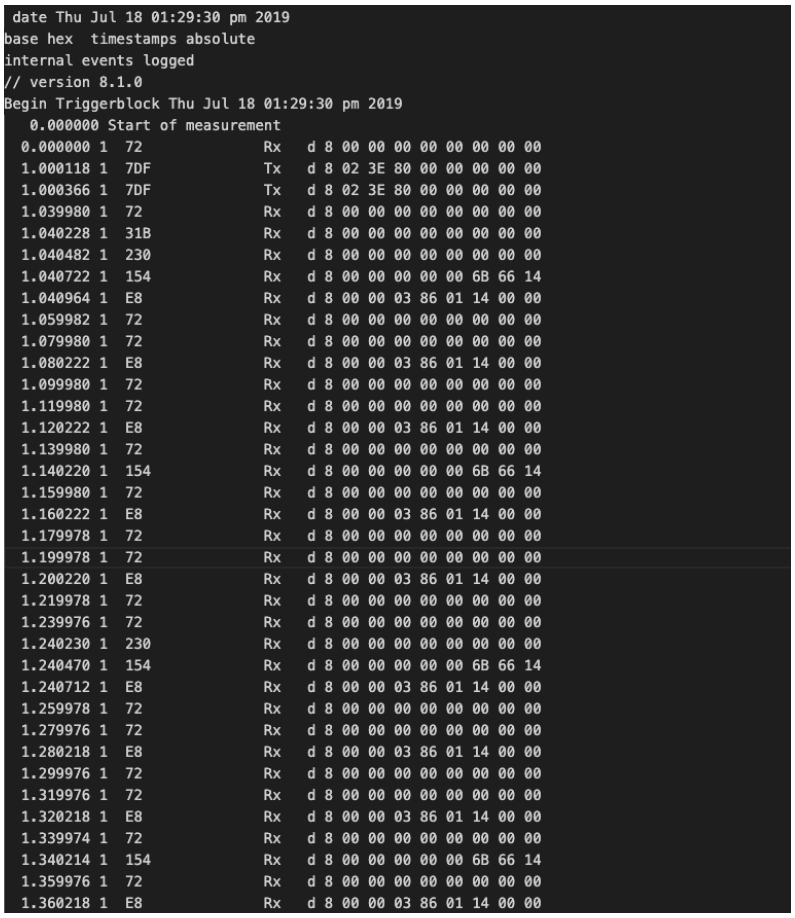
Username and Password Accepted

Login successful



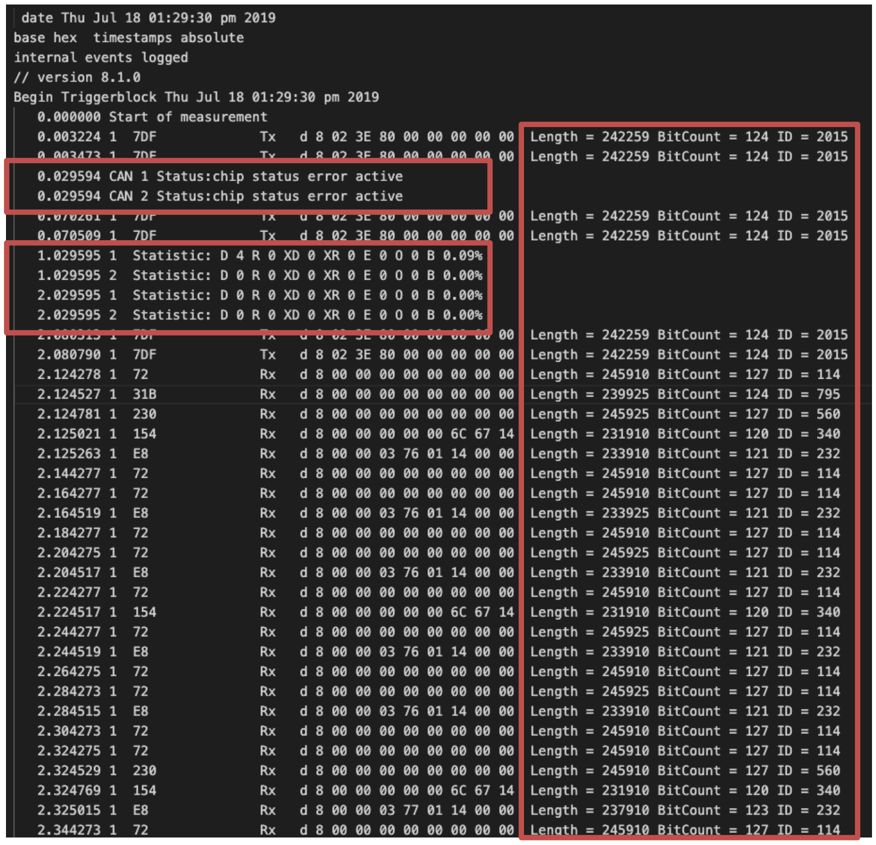
Main Menu

Candidate Name: Candidate Number:



```
 date Thu Jul 18 01:29:30 pm 2019
base hex  timestamps absolute
internal events logged
// version 8.1.0
Begin Triggerblock Thu Jul 18 01:29:30 pm 2019
   0.000000 Start of measurement
  0.000000 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.000118 1  7DF             Tx   d 8 02 3E 80 00 00 00 00 00
  1.000366 1  7DF             Tx   d 8 02 3E 80 00 00 00 00 00
  1.039980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.040228 1  31B             Rx   d 8 00 00 00 00 00 00 00 00
  1.040482 1  230             Rx   d 8 00 00 00 00 00 00 00 00
  1.040722 1  154             Rx   d 8 00 00 00 00 00 6B 66 14
  1.040964 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.059982 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.079980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.080222 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.099980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.119980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.120222 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.139980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.140220 1  154             Rx   d 8 00 00 00 00 00 6B 66 14
  1.159980 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.160222 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.179978 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.199978 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.200220 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.219978 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.239976 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.240230 1  230             Rx   d 8 00 00 00 00 00 00 00 00
  1.240470 1  154             Rx   d 8 00 00 00 00 00 6B 66 14
  1.240712 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.259978 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.279976 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.280218 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.299976 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.319976 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.320218 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
  1.339974 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.340214 1  154             Rx   d 8 00 00 00 00 00 6B 66 14
  1.359976 1  72              Rx   d 8 00 00 00 00 00 00 00 00
  1.360218 1  E8              Rx   d 8 00 00 03 86 01 14 00 00
```

Data after conversion



```
 date Thu Jul 18 01:29:30 pm 2019
base hex  timestamps absolute
internal events logged
// version 8.1.0
Begin Triggerblock Thu Jul 18 01:29:30 pm 2019
   0.000000 Start of measurement
   0.003224 1  7DF            Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   0.003473 1  7DF            Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   0.029594 CAN 1 Status:chip status error active
   0.029594 CAN 2 Status:chip status error active
   0.070261 1  7DF            Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   0.070509 1  7DF            Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   1.029595 1  Statistic: D 4 R 0 XD 0 XR 0 E 0 O 0 B 0.09%
   1.029595 2  Statistic: D 0 R 0 XD 0 XR 0 E 0 O 0 B 0.00%
   2.029595 1  Statistic: D 0 R 0 XD 0 XR 0 E 0 O 0 B 0.00%
   2.029595 2  Statistic: D 0 R 0 XD 0 XR 0 E 0 O 0 B 0.00%
   2.080313 1  7DF            Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   2.080790 1  7DF            Tx   d 8 02 3E 80 00 00 00 00 00  Length = 242259 BitCount = 124 ID = 2015
   2.124278 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.124527 1  31B            Rx   d 8 00 00 00 00 00 00 00 00  Length = 239925 BitCount = 124 ID = 795
   2.124781 1  230            Rx   d 8 00 00 00 00 00 00 00 00  Length = 245925 BitCount = 127 ID = 560
   2.125021 1  154            Rx   d 8 00 00 00 00 00 6C 67 14  Length = 231910 BitCount = 120 ID = 340
   2.125263 1  E8             Rx   d 8 00 00 03 76 01 14 00 00  Length = 233910 BitCount = 121 ID = 232
   2.144277 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.164277 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.164519 1  E8             Rx   d 8 00 00 03 76 01 14 00 00  Length = 233925 BitCount = 121 ID = 232
   2.184277 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.204275 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245925 BitCount = 127 ID = 114
   2.204517 1  E8             Rx   d 8 00 00 03 76 01 14 00 00  Length = 233910 BitCount = 121 ID = 232
   2.224277 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.224517 1  154            Rx   d 8 00 00 00 00 00 6C 67 14  Length = 231910 BitCount = 120 ID = 340
   2.244277 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245925 BitCount = 127 ID = 114
   2.244519 1  E8             Rx   d 8 00 00 03 76 01 14 00 00  Length = 233910 BitCount = 121 ID = 232
   2.264275 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.284273 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245925 BitCount = 127 ID = 114
   2.284515 1  E8             Rx   d 8 00 00 03 76 01 14 00 00  Length = 233910 BitCount = 121 ID = 232
   2.304273 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.324275 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
   2.324529 1  230            Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 560
   2.324769 1  154            Rx   d 8 00 00 00 00 00 6C 67 14  Length = 231910 BitCount = 120 ID = 340
   2.325015 1  E8             Rx   d 8 00 00 03 77 01 14 00 00  Length = 237910 BitCount = 123 ID = 232
   2.344273 1  72             Rx   d 8 00 00 00 00 00 00 00 00  Length = 245910 BitCount = 127 ID = 114
```

Data before conversion – sections marked red have been removed.

(II) MAINTENANCE AND DEVELOPMENT

As a first version of my program I am happy with the design and functionality of it. In future I would like to add more versatility and have more files available for conversion. The main reason for not doing this in the first development is that I did not have the time to I made the decision not to include more due to that.

I would also adjust the 'login' screen, so it looked slightly different to the main menu and more professional. I would also like to add more security or implement connection to the JLR network The program's structure is quite modular due to the fact that it uses many functions. This means that further development on this program is relatively easy since code in functions and modules can be edited easily without having to adjust the whole program and interfering with working code.

This means that the program is quite maintainable in the future, since fixing errors or changes to each class can be done independently of the main loop. Certain functions can be changed, and this would not affect any important parts of the program. I have also added comments to my code which can title each function and definition to make it easier to understand in the future. I can adjust names or variables, add new ones, remove variables and do the same with functions and it would not drastically affect the running of the program (Milliken, 2020).

Since most changes to the program that I would add or stakeholders would add do not affect the main loop, I am satisfied with the standard of maintainability of the program.

I would like to package the program making it more distributable in the future however this is a hard task at this point in time.

REFERENCES

BROWNING, J. B. & ALCHIN, M. 2019. Pro Python 3 : Features and Tools for Professional Development.

CHAUDHARY, B. 2013. Tkinter GUI application development HOTSHOT.

KAPIL, S. 2019. Clean Python : Elegant Coding in Python.

KAWAHARA, K., MATSUBARA, Y. & TAKADA, H. 2014. A Simulation Environment and preliminary evaluation for Automotive CAN-Ethernet AVB Networks.

KELLY, S. & SAFARI, A. O. R. M. C. 2019. Python, PyGame, and Raspberry Pi Game Development.

MILLIKEN, C. P. 2020. *Python projects for beginners : a ten-week bootcamp approach to Python programming*.

NTANTOGIAN, C., MALLIAROS, S. & XENAKIS, C. 2019. Evaluation of password hashing schemes in open source web platforms. *Computers & Security,* 84**,** 206-224.

SITI-FARHANA, L., ABU TALIB, O. & MUHAMMAD-HUSAINI, A.-B. 2019. Intrusion detection system for automotive Controller Area Network (CAN) bus system: a review. *EURASIP Journal on Wireless Communications and Networking***,** 1.

Gitlab URL: https://mygit.wmg.warwick.ac.uk/u1921983/SmartSolutionsAssessment3