University of Oklahoma: AME 4442.011 – Group 1

April 24th, 2023

By

John DeJarnette, Bryant Boatright, Colton Wirth, Darius Baker, Devin Dunford, Jared Belser, Jesse Clemence, and Kyler Clark

# Signature Page

**Statement of integrity**

“As a student at the University of Oklahoma, I affirm that I will neither give nor receive inappropriate aid in the completion of any academic exercise. I understand that it is my responsibility to comply with the Academic Misconduct Code.”

Shape

Description automatically generated with medium confidence**Signed**



Text, whiteboard

Description automatically generated



Table of Contents

[Signature Page 1](#_Toc132709082)

[Nomenclature 3](#_Toc132709083)

[List of Figures 4](#_Toc132709084)

[List of Tables 5](#_Toc132709085)

[Project Goal 6](#_Toc132709086)

[Background Theory 7](#_Toc132709087)

[Project Description 9](#_Toc132709088)

[Discussion 10](#_Toc132709089)

[Conclusion 11](#_Toc132709090)

[Works Cited 12](#_Toc132709091)

[Appendix A: Usage Manual 13](#_Toc132709092)

# Nomenclature

Brake Work

Torque

N Engine Speed  
 Displacement Volume

n Number of Revolutions per cycle

Fuel Mass Flow Rate

*mconsumed* Mass of Consumed Fuel

*t* Time

Thermal Efficiency

Heat Transfer Rate In

Lower Heating Value of Fuel

*Tambient*  Ambient Temperature

*Pvapor* Vapor Pressure

*Patm* Atmospheric Pressure

C Correction Factor

Air Mass Flow Rate

AFstoich Stoichiometric Air Fuel Ratio

(Fuel) Equivalence Ratio Variable  
 Volumetric Efficiency

Ambient Density of Air

# List of Figures

[Figure 1: Dashboard UI Design 11](#_Toc133078640)

[Figure 2: Data I/O Stream Sub-Systems 12](#_Toc133078641)

# List of Tables

[Table 1: Summary of Received and Calculated Parameter Types 13](#_Toc133078603)

# Project Goal

The purpose of this project, in conjunction with other lab teams, was to develop and implement a measurement and visualization system to update the lab’s analog measurement systems to digital measurement systems. The desired project result would provide users and interface that displays, in real time, key parameters, both measured and computed, about the engine undergoing testing. This would provide the benefit of a single interface to observe parameters mid-test as well as simplifying recording the test data when compared to recording analog data manually.

Group 1’s portion of the overall lab project was to develop the Simulink dashboard that receives the measurements from different engines, computes any desired parameters, generates output logs, and displays the real time data to the user. The specific requirements listed for Group 1’s project were:

Real-time data visualization and data acquisition system in MATLAB/ Simulink

1. Use the last MATLAB version from OU IT (MATLAB 2022b).
2. Key data direct measurement: Engine speed (in RPM), engine torque (in Nm), fuel mass  
   flow rate (in kg/s), exhaust gas temperature (in  
   ° C), intake air flow (in kg/s), pressure  
   inside barrel (in kPa), ambient pressure (in kPa).
3. Key data calculated: Brake power (in kW), BMEP (in kPa), fuel mass flow rate (in kg/s)  
   BSFC (in kg/kJ), Thermal Efficiency (in %), Volumetric Efficiency (in %).
4. Desktop is provided in the lab, which will be the machine for your final demo.

These requirements were to be implemented on both desktop computers in the lab with the ability to display measurement data from all 6 engines.

# Background Theory

The equations used within the MATLAB function blocks of the logic subsystem in our model are shown as follows:

**Brake Mean Effective Pressure**

**(Eq. 1)**

Brake Mean Effective Pressure (BMEP) is a measure of the average pressure exerted on the piston of an internal combustion engine during one complete engine cycle [1]

BMEP Trends:

Naturally aspirated SI engines: 120 – 150 psi  
 Naturally aspirated CI engines: 100 – 130 psi  
 Supercharged/turbocharged engines: 145 – 75 psi

**Brake Work**

**(Eq.2)**

Brake Work is a measure of the power output of an internal combustion engine and represents the amount of work done by the engine on the output shaft. Power should increase as the load increases.

**Brake Specific Fuel Consumption**

**(Eq.3)**

Brake Specific Fuel Consumption (BSFC) is a measure of the fuel efficiency of an internal combustion engine and allows direct comparison of different engines.

**Fuel Mass Flow Rate**

**(Eq.4)**

Fuel mass flow rate is a measure of the amount of fuel that is being consumed by an internal combustion engine per unit of time.

**Torque**

**(Eq.5)**

Torque is a measure of the rotational force produced by an internal combustion engine and is found by rearraigning the BMEP formula to solve for torque.

**Equivalence Ratio**

**(Eq.6)**

Equivalence ratio is a measure of the fuel-to-air ratio in the combustion process of an internal combustion engine relative to the stoichiometric ratio.

= 1: Stoichiometric reaction (fuel/oxygen completely consumed)

> 1: fuel “rich” reaction (fuel left over after reaction)  
 < 1: fuel “lean” reaction (oxygen left over after reaction)

**Volumetric Efficiency**

**(Eq.7)**

Volumetric efficiency is a measure of how efficiently an engine can move the air/fuel mixture in or out of the combustion chamber.

= 100% : The air pressure in the combustion chamber is close to ambient conditions before combustion.

< 100%: The air pressure in the combustion chamber is lower than ambient.

> 100%: The air pressure in the combustion chamber is higher than ambient.

**Thermal Efficiency**

**(Eq.8)**

Measurement of how well thermal energy from the fuel is being converted into mechanical work. Generally, diesel engines have better thermal efficiency than gasoline engines.

# Project Description

The task of developing a dashboard was split into three main sub-tasks: GUI development, I/O (File and interfacing with group 2) and implementing calculations for desired parameters. Each of these three sub-tasks were placed into their own subsystem in Simulink.

## GUI Development

The user interface was developed using dashboard components within Simulink. These components are able to be linked to signals within Simulink and thus can be connected to the desired signal in the calculations subsystem. The majority of work for this subsystem was put into visually arranging the dashboard to ensure that all desired parameters are displayed in such a way that the component used to display the data is complementary to how the user will most likely be interpreting the data. Care was taken to maximize the usage of space on the lab monitors by maximizing the size of plots and text where possible.

**Dashboard Contents**

1. Plots
   1. Thermal Efficiency
   2. Brake Power (kW)
   3. Exhaust Temperature ()
   4. BSFC (kg/kj)
   5. Volumetric Efficiency
2. Gauges
   1. Torque (Nm)
   2. Engine Speed (RPM)
3. Displays
   1. Fuel Mass (kg)
   2. Fuel Mass Flow Rate (kg/s)
   3. Intake Air Flow Rate (kg/s)
   4. BMEP (kPa)
   5. Oxygen %
   6. Carbon Dioxide %
   7. Carbon Monoxide %
   8. Nitric Oxide PPM
   9. Hydrocarbon PPM
4. Selection Tools
   1. Engine Selection
   2. Motor Start/Stop
5. Testing Controls
   1. Load

The design was selected to plot the values that may fluctuate greatly over the course of a singular/constant loading, whereas any displays were selected for their more constant nature over the course of the same loading. This is such that any varying values can have their trends displayed over the course of this loading for educational/research purposes. Additionally, the dashboard utilizes user controls to allow for variable engine types to be tested currently the dashboard and accompanying code is configured for the following five engines.

1. Yanmar
2. Craftsman Generator
3. John Diesel
4. Chevy V6
5. Kohler

Each engine has slight variances in its displacement volume, and one engine has diesel as the combustible rather than gasoline, so the developed code was made to dynamically adjust this based on the user selection. Another useful control implemented by the team includes the ability for the user to select when to begin data plotting/recording manually rather than immediate upon runtime. This control is the engine start/stop button created to simulate the activation of the engine and allow for this improved user experience.

Finally, the design utilizes gauges to display values that will remain nearly constant over the course of a particular loading but remain integral to the understanding of the engine’s performance. These values were selected to be the engines output torque and speed as these values are the some of the most commonly reported values when referencing an engines performance.

The completed UI/UX design as described is displayed below in Figure 1.

Calendar

Description automatically generated

Figure 1: Dashboard UI Design

## I/O

Input and output consisted of two focuses - taking input from Group 2 and writing output files. Due to the ability to manipulate, organize, and interpret data in other programming tools, a decision was made to use csv or comma separated values, “.csv” extension, as the output file type. File writing was conducted in MATLAB Function blocks using MATLAB file operation functions “fopen” and “fprintf”. One difficulty found in writing to an output file was the dynamic generation of a file name. An initial goal was to include a UI element for the user to provide a string that would be the name of the output file. However, difficulties were found when dynamically defining a string and passing that as a signal to the file output function. Thus, a predetermined output file name was determined using the date and time in the format of:

“yyyy-MM-DD--HH-mm-ss-output.csv” and for example, “2023-04-18--11-34-35-output.csv”.

To generate file names for each run, a triggered subsystem that interprets the on/off state using a rising trigger was used so that a new file name was generated once at the start of each run and otherwise an empty string was passed to the file output function.

A goal for receiving the input data from Group 2 was to create a plug and play system that easily interacts with their model. This was the driving decision for abstracting the data input subsystem away from the logic subsystem. One design decision made that assisted with this abstraction was the use of a filtering function in our logic subsystem that essentially operated as a boolean AND that investigated whether the on/off button was pressed. If the on/off button was selected, then the measured data would be passed into the logic and thus reported to the file output steam. This collection of linked Group 2 input data to finalize calculated outputs and file display is housed within the two sub-systems displayed below in Figure 2.

Table

Description automatically generated

Figure 2: Data I/O Stream Sub-Systems

## Logic and Calculations

For parameters that required calculations that used measured data, MATLAB Function blocks were utilized to programmatically calculate the parameters.

When designing this project two main considerations were made, the first of which being logic as we needed to define how the program would run based on different run conditions (such as varying engine types and control signals), and the second being the background theory calculations that needed to be performed and displayed.

When designing logic, the team sacrificed simplicity for functionality and usability. To begin the team created 3 separate sub-system each of which houses a different functional component of the code. The first of which is the dashboard sub-system for the continuous monitoring and display of the data received from the engines, next the team created a single separate sub-system for group 2 to interface with thereby eliminating any type of interpersonal confusion or potential for code breakage, finally the third sub-system housed all of our own internal logic configuration.

While the dashboard (GUI) contents and breakdown is described above, the following considerations were made when designing the remainder of the logic.

1. Seamless integration of user-controlled start/stop functionality (to display and file I/O)
2. Implementing multiple engine functionality (create an index unique to each engine)
3. Allow for dynamic file name creation to constantly produce new output files.
   1. Rather than re-write old files.

Each of these considerations were built into the design of the logic such that the code would not only function in creating and displaying the necessary values but would provide the operator with a more manageable user – experience.

The values received by team 2 in the second sub – system are displayed in Table 1 below in the directly received column.

Table 1: Summary of Received and Calculated Parameter Types

|  |  |
| --- | --- |
| **Directly Received** | **Calculated** |
| RPM | Brake Power (kW) |
| Exhaust Temperature (℃) | BMEP (kPa) |
| Torque (Nm) | Fuel Mass Flow (kg/s) |
| Fuel Mass (g) | BSFC (kg/kj) |
| Air Flow (kg/s) | Thermal Eff. % |
| Barrel Pressure (kPa) | Volumetric Eff. % |
| Ambient Pressure (kPa) | N/A |
| Oxygen % | N/A |
| Carbon Dioxide % | N/A |
| Carbon Monoxide % | N/A |
| Nitric Oxide PPM | N/A |
| Hydrocarbon PPM | N/A |

With these values and the engine specific displacement volume, each value described in background theory, and displayed in the calculated column of Table 1, was evaluated and reported to the GUI.

Calculations were uniform across all engines with the exception of the number of cylinders available in the engine for engine performance calculations. This factor was implemented in a separate MATLAB function that stored the number of cylinders for each engine and reported it to the dependent functions such as brake work and volumetric efficiency. The equations mentioned in background theory were implemented using MATLAB functions.

# Discussion

The team created this project with the intention of testing it with real-time data from multiple engines. However, during the development phase the team only had the ability to test simulated data both computer generated and created from mock Arduino signals.

For computer generated data the team was able to test constants being sent to each of the requested inputs from team 2. Upon final design implementation the team presented the program with values recorded from one of the previous labs with known values and calculated results. The team noted that the reported values from the MATLAB script matched that as was expected from the known calculations, signifying that the basic set-up of the logic and calculation code was correct. With this information the team was able to confirm internal design functionality.

The team then moved on to testing live data that must be read from an Arduino and imported through MATLAB libraries. This test was to confirm the compatibility of our MATLAB script and Arduino signaling to ensure smooth transitions between the real-time I/O types. Using a simplified code scheme (testing a single pin reading) the team verified the ability of the MATLAB libraries to accurately read the dispatched Arduino signals. While this test did not assess the full functionality of transmitting 12 analog signals at the same time, it did verify that the script could read and interpret one signal. Known that the script could perform its tasks with constants and that it could read in Arduino signals allowed the team to sufficiently determine it could read and process 12 signals the same way.

Unfortunately, team 2 did not create the necessary interface needed to acquire the other groups’ signals into our own script, and thus we were unable to see a live verification of successful demonstration. It is believed however, that the tests performed by the team sufficiently prove its efficacy in performing these tasks.

There are however other weaknesses present throughout this project. For example, this code is currently developed particularly for these engines, if another engine is to be added each of the linked MATLAB functions must be adjusted to make this change. Additionally, there were some UI design limitations the team encountered that affected the aesthetic nature and usability of the project. Such limitations included the team’s ability to programmatically set the axis titles, unit type, axis tick increment, and axis information orientation of plots. This led to the produced plots having inconsistent visuals as their axis information became nearly unreadable. While this does not necessarily affect the functionality of this project (as these plotted results are also stored in a .csv for later more technical review), it would have made for a better user – experience had the team managed a way to control these behaviors.

Outside of these design weaknesses there are two features the team desired to implement but did not have the ability or time to execute. The first of these changes includes the ability to control the engines activation from the included start/stop button. With the right hardware (i.e. a remotely activated ignition system) the engines could be started from the same button click as the data collection to ensure no data is lost during collection. The next design change includes the ability to transition between metric and English units. The team desired to implement this functionality, however due to time constraints it was not developed. With this improvement the project could easily display the same values in different unit systems for greater usability by the operator.

While these weaknesses and improvements would provide increased usability, the team ensured to design the script with accurate and maximized functionality such that it could be quickly utilized in the existing system.

# Conclusion

After completing the required files and goals for this project the team’s resultant takeaways were the following: increased understanding of MATLAB/Simulink functionality, greater UI/UX planning and design, and improved I/O handling through this software.

Several members of the team were experienced programmers in multiple text-based languages, and other graphical programing methods such as LabVIEW. However, the team as a whole lacked the background in MATLAB/Simulink to rapidly develop this script. Instead, the team learned through a multi-week process various functional abilities available in Simulink that aided them in their design process. The team learned how to create sub-systems of larger code segments to enhance usability, what options are available for real-time data displays in Simulink, and how to implement complex equations into this visual development environment. With this knowledge, the team was confident in their ability to design and develop data driven logic systems such as this live engine measurement display.

This data driven logic was built upon by the team as knowledge learned through lectures, fundamental to the application of many necessary performance equations, was utilized in the implementation of the source code. The team utilized this knowledge to gather all necessary inputs to the series of equations, gather through device (Arduino) I/O, and then implement these values in the theoretical equations learned this semester. Beyond this application, the team learned how to leverage the data-drive nature of Simulink as the output equation values could not be determined until the received signal-based values were gathered. Additionally, this data-driven nature helped the team to learn how to implement system controls (such as the Start/Stop) that activated and deactivated the entire I/O and calculation sequences. There were additional nuanced learning opportunities throughout the development of this script such as the encoding of filenames to ASCII for file I/O, and action event triggers for logic control that helped the team become more familiar with this language and confident in the overall result of this project.

Once the team had determined how to approach the foundational functionality behind the backend of this dashboard, the next learning process came through creating usable and visually appealing UI/UX designs. With the plethora of dashboard options available to the team through Simulink’s toolkits, some creative design to determine the representation type most suitable for each data type. When assessing this most suitable choice, not only was the practical consideration of how the data varies over a load cycle considered, but also the team desired to design the UI to fill a computer’s screen in its entirety while providing quick to analyze data displays. Therefore, the team learned how to select and configure a display to optimize a user’s experience whilst maintaining data trend depictions for all qualifying measurements.

In summation, the team’s development process with this project enhanced their overall understanding of Simulink functionality, I/O handling, and UI/UX design.

# Works Cited

|  |  |
| --- | --- |
| [1] | J. Heywood, Internal Combustion Engine Fundamentals, New York: McGraw Hill, 1988. |

# Appendix A: Usage Manual

The usage manual has been generated as a PDF and includes instructions for continued development of the dashboard as well as the manual. See the Usage.pdf document for the manual.