IMPLEMENTING A COHESIVE PROGRAMMING ECOSYSTEM IN MECHANICAL ENGINEERING

by

ZAKARY CHRISTIAN OSTER

B.S., Kansas State University, 2022

A THESIS

submitted in partial fulfillment of the requirements for the degree

MASTER OF SCIENCE

Department of Mechanical Engineering
College of Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas
2024

Approved by:

Major Professor
Jeremy A. Roberts

Abstract

Talk about the how programming is important and people need to learn it. Then talk about how many do not seem confident in programming and avoid it like the plague. One fix could be a more cohesive and structured approach to programming throughout the program. This research proves it is possible to have one language, environment, and hardware set for the entire degree.

Contents

List of Figures

List of Tables

Chapter 1
Introduction and Background

1.1 Introduction

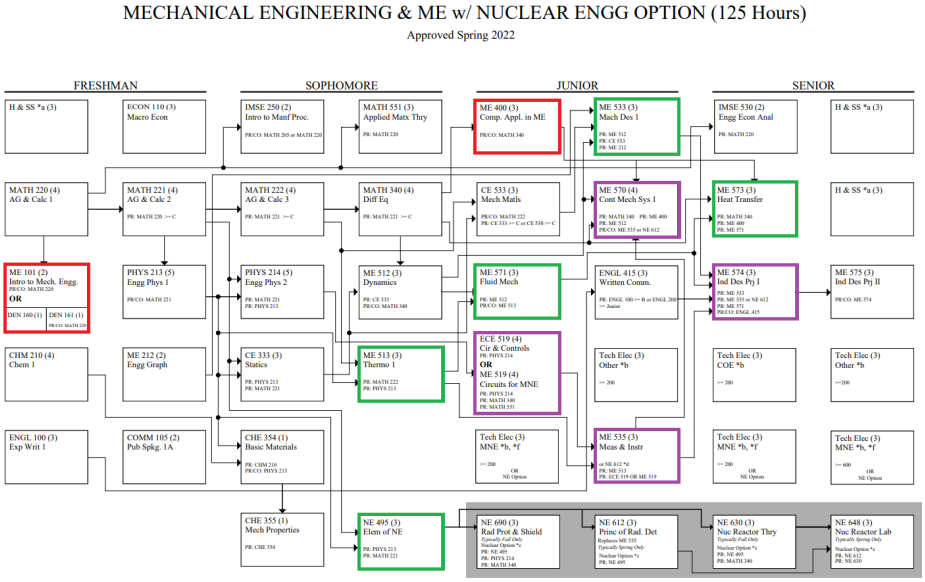
Engineers solve problems. Regardless of field, discipline, and generation, every engineer strives to solve problems. While the need for problem solvers does not change with time, the methods for solving problems certainly do. Consider the process of creating detail drawings. Engineers needed a way to communicate the size, shape, and finish of parts to other engineers, so they hand drew and dimensioned parts. But in the modern day, the thought of hand drafting a detail drawing is laughable. And education has adjusted in kind. While students are taught to draw straight lines (which all lines are) and visualize without the aid of CAD, no class spends time teaching students how to use a drafting table or create detail drawings by hand. Rather, they are taught the basics of visualization and then taught how to use SOLIDWORKS.

The same could be said for setting the state in a thermodynamics problem, graphing deflection in a load-bearing beam, or finding the Q-value of a fission reaction. At one point, looking up values in a table and plugging them into a calculator or using a ruler and lined paper were state-of-the-art methods for solving these problems. However, as the times change, so do the methods, and with the advent of programming and its ever lowering barrier to entry, these problems can more easily, and more accurately, be solved with a simple script in any number of programming languages.

Despite this, most classes continue to have students spend hours thumbing through tables and solve every problem by hand. In contrast to this, other classes have recognized the value of modern solution methods and rely heavily on them, only to find that students lack sufficient training to effectively utilize them. Because of this dichotomy, a cohesive and consistent programming ecosystem should be implemented throughout the curriculum of the Mechanical and Nuclear Engineering department.

1.2 Background

The first introduction to programming in MNE curriculum is DEN 161: Engineering Problem Solving, the first-semester introduction-to-engineering class. The class spends a few weeks introducing Python and then a week introducing the Arduino Uno. After this, no programming is used in the curriculum until first semester Junior year, as seen in figure 1.1.



Red : Teaches Programming
Purple : Uses Programming
Green : Does not utilize Programming, but could

Figure 1.1: MNE Curriculum Flowchart

The next class to use programming is also the only class dedicated to teaching programmig. ME 400: Computer Applications in Mechanical Engineering begins with an introduction to C++ but then quickly moves to teaching embedded C++ for use on an ESP32. While the class contains phenomenal material as it relates to embedded programming, its face paced nature often leaves students confused on the fundamentals of programming.

The next two classes to use programming are ME 519: Cirucuits for MNE and ME 400: Control of Mechanical Systems 1. Both classes use MATLAB for system analysis and only use MATLAB's introductory Onramp to teach

students how to use the language and environment. Due the inadequate foundations in programming, many students struggle with the new language, making it difficult for them to understand the new concepts being taught, such as a PID controller.

The last, non-project based class to use programming is ME 535: Measurements and Instrumentation. This class is a bit of an outlier, because it uses the graphical programming language known as LabVIEW. Rather than writing lines of code, users can add and edit "blocks" that have predefined functionality. The class teaches how to use every block that is needed to complete the labs, but much of the details are left unexplored.

The final class that utilizes programming is ME 574: Interdisciplinary Design Project 1, better known as Senior Design 1. This class revolves around the completion and fabrication of a single product. This product always has some electromechanical aspects that need to be controlled by a microcontroller. Neither the device not language used is moderated by the class instructors.

As it currently stands, the MNE department makes use of four different programming languages in required classes: C++, MATLAB, Python, and LabVIEW. While each of these languages and environments come with their own benefits, and there is certainly a benefit to using multiple languages, many students find themselves confused by the variety. Rather than feeling confident in one language, they feel unconfident in four languages.

In addition to the weak foundation and lack of consistency, many classes are not using programming that would benefit from using it. Any class that uses tables to look up values or iterative design methods, such as Thermodynamics, Heat Transfer, or Fluid Mechanics, would make good use of programming.

Based on experience as an instructor of ME 570 and ME 574 for the better part of four years, the majority of students struggle with using MATLAB and C++. This makes it difficult for students to learn how to design a control system when they do not understand the tool they are using to analyze it.

1.3 Scope of Work

While the motivation for this work relies on the idea that a more consistent approach to learning and applying programming would benefit students and improve their problem solving abilities, this project does not seek to prove

or verify this claim. Neither does this work set out to prove that the proposed solution is the best solution for the assumed problem. This project merely seeks to identify a potential path forward while demonstrating both the advantages and disadvantages of the curriculum.

This work will take time to point out benefits and drawbacks compared to the current curriculum, but given that no other possible solution is considered, it would be premature to draw conclusions relating to the "best" path forward. However, there will be a brief discussion of alternative options in the concluding chapter of the paper.

1.4 Structure of Work

Since this body of work does not set out to prove a hypothesis, opting instead to demonstrate a consistent usage and application of programming across the MNE curriculum, the format of this paper will be adjusted accordingly. The next chapter, titled "A Cohesive Programming Curriculum," will discuss the foundation of the new implementation, such as the language, hardware, and development environment. It will also discuss the general goal of the changes.

Subsequent chapters will each be dedicated to a class in the MNE curriculum. These chapters will give a course overview, describe the new or altered assignments, and give a list of deliverables for that course. The actual assignment descriptions and implementations are not contained in the chapters themselves but rather in a repository on GitHub. Each chapter will have a folder of the same name dedicated to it in the repository. This folder will contain everything an instructor would need to assign and grade the new assignments.

Chapter 2
A Cohesive Programming Curriculum

2.1 Curriculum Proposal

The primary concerns addressed in Chapter 1 are as follows:

1. Students lack a solid foundation in the fundamentals of programming.
2. Programming is under utilized as a method of problem solving.

To address the first issue, this paper would like to make two proposals: the adoption of a single programming language and the restructuring of electronics classes in the department. First, the goal of adopting a single programming language is to give students a more thorough understanding in a single language rather than a shallow understanding of four languages. The language chosen in this project is Python. In conjunction with Python, a consistent environemnt should be used as well. A consistent environment will prevent unnecessary confusions from IDE specific features and setup requirements. The choosen environment for this project is Visual Studio Code. To replace the work done on the Arduino Uno and ESP32, a Raspberry Pi Pico will used thanks to its seemless integration with both Python and Visual Studio Code. For more details regarding these choices, see Sections 2.2-2.5.

Second, the goal of restructuring electronics classes is to give students a dedicated "fundamentals of programming" class rather than a microcontroller class that teaches programming out of necessity. This change, however, will not be addressed for the bulk of this work. Significant changes to the courses required in the curriculum are difficult to pass, and always come with tradeoffs. As such, discussion of this topic will be saved for reflection in the conclusion of this paper.

To address the second primary concern, the under-utilization of programming in the curriculum, two more changes are being proposed: the addition of assignments or projects that make use of programming software and the creation of a MNE library for use in most, if not all, MNE classes. The majority of this paper, as well as the supporting repository, is dedicated to creating assignments that utilize programming for classes that do not have one, translating assignments from other languages in classes that do utilize

programming, and creating a functional, though incomplete, Python package that could be used throughout the department. The goal of these changes is twofold: teaching students how to use modern problem solving techniques to find solutions and giving students a chance to reinforce the programming skills they were taught. By giving students a more consistent approach to programming, as well as more use cases for the value of programming, students will become more confident and more capable as problem solvers and engineers.

Through the course of this project, many different Python packages and Visual Studio Code extensions will be utilized. Thankfully, the installation and usage steps are simple, and can be combined into a single step in both cases. The Visual Studio Code extensions can all by downloaded at once using an extension pack called KSU Mechancial Engineering Extension Pack. The necessary Python packages can also all be installed at once from a requirements file. In some cases, it may be preferable to use Anaconda, a Python distribution that comes many standard packages, to prevent the need for students to install packages on their own. To see a full list of the applications, packages, and extensions and their versions, see Appendix B. For installation instructions, see the git repository in Appendix A.

2.2 Python

Python is a multipurpose, interpreted programming langauge that emphasises code readability. The langauge is was originally introduced by Guido van Rossum in 1991 and has gone through three major versions, the most recent being the aptly named Python 3. Using Python, particularly for first time programmers, comes with the following benefits:

- Python is an interpreted language; therefore, users do not need to install and configure a compiler. The setup process for a compiler can often be confusing and a major barrier to entry for new users. Since Python does not have a compiler, the only download that is necessary to run a .py file is Python itself. The lack of a compiler does come with some downsides, namely speed, but these will be addressed later on.

- Python is designed to be human readable. This means that the syntax of the language closely matches the structure of the plain English. This emphasis on readability greatly reduces the burden on new users and

bolsters code comprehension, even in more experienced coders. Consider a program that prints the phrase "Hello world!" to the terminal. In C++, this would be done with the following code:

```
#include <iostream>

int main() {
    std::cout << "Hello world!" << std::endl;
    return 0;
}
```

Not only is this code not human readable, but it requires the use of imports, functions, return values, namespaces, and non-obvious syntax. In Python, however, only one line of code is needed:

```
print("Hello world!")
```

- Python is a multipurpose, general-use language. The language is adept at everything from data science to building a webiste to powering a microcontroller (using MicroPython). With such a wide breadth of use cases, learning the language opens the door to countless different applications and projects.

- Python is open-source and currently one of the most used languages. Thanks to its popularity, many tools exist to make programming with Python easier, more powerful, and more accessible such as integrations with Jupyter Notebooks and Visual Studio Code. However, this can potentially be two edged sword, as will be discussed later.

- Python has an easy to use package manager called pip, which is a recursize acronym for pip installs packages. This package manager makes it incredibly easy to both install and use non-standard libraries in Python.

Python provides a solid starting point for new programmers while also being fully capable of high-level, advanced programming making it a good language for users of all skill levels. However, it does not come without concerns of its own, most noteably version controlling. This will be addressed at length in a future chapter.

2.3 MicroPython

MicroPython is a slimmed version of Python that is designed to run on microcontrollers, such as the Raspberry Pi Pico or ESP32. This Python derivative removes some high level features in exchange for a smaller interpreter and a machine library dedicated to interacting with the hardware of the microcontroller, which Python would normally not interact with.

While MicroPython comes with all the benefits of Python, it also comes with one of the drawbacks of Python: speed. Unlike traditional low level languages like C/C++, MicroPython is not compiled, making it significantly slower than comparable languages. In some applications, the difference in speed has no affect on the application. Others, like a control system, heavily depend on fast loop times and are greatly impacted by speed.

2.4 Raspberry Pi Pico

The Raspberry Pi Pico is a small-but-powerful microcontroller made by Raspberry Pi. A microcontroller is an electronic input-output device capable of running uploaded code. It is differentiated from a microprocessor, such as a Raspberry Pi 5 or a desktop computer, by the fact that it does not run an operating system.

The using a Pico, as a newer competitor in the saturated microcontroller market, begs the question of "why not use the Arduino Uno or ESP32?" While all three devices support ADC, SPI, I2C, and have around 40 GPIO pins, only the ESP32 and Pico support MicroPython, as well as built in wireless and Bluetooth connectivity.

A case could be made for using the ESP32 over the Pico, given its rise in popularity in the home automation space. However, the ESP32 comes with greater restrictions in GPIO usage and the majority of its documentation and drivers are for C++ programs rather than MicroPython, giving the Pico an edge in user friendlinesss.

As was the case with Python and MicroPython, the Pico does not come without drawbacks. Almost all sensors come with drivers written in C++ for the Arduino Uno or ESP32. This is not the case for the Pico. Some sensors made specifically to work with the Pico come with drivers, and others may be found on GitHub thanks to the work of another tinkering user, but many pieces of hardware do not have driver support for the Pico. This could end

up being a limiting factor for students trying to do unguided projects.

2.5 Visual Studio Code

Visual Studio Code is a cross-platform, highly customizeable development environment created by Microsoft. It is one of the most used and most accessible development environments available, and supports use with any language. VSCode can almost be thought of as a Word processor where spell check looks for syntax errors and keywords instead of grammatical mistakes. By installing extensions, which can be done from inside VSCode, developers can write code for nearly any microcontroller or processor and in nearly any language.

For the purposes of the Mechanical Engineering Department, extensions for Python, MicroPython and the Pico, and Jupyter Notebooks (a type of interactive programming environment that will be utilized later) will be used. The extensions are both easy to use and install, but share a downside with Python: version control. Since these extensions are always in active development and get updated to work with the newest version of Python and Visual Studio Code, which may not always be desireable. This issue will be discussed in greater detail in a later chapter.

Chapter 3
DEN 161: Engineering Problem Solving

3.1 Course Overview

DEN 161: Engineering Problem Solving is a lab-style course that complements the lecture-oriented DEN 160: Engineering Orientation. The class focuses on providing hands-on, problem solving experiences through projects from multiple engineering disciplines. While these projects serve as the students' introduction to different engineering disciplines, they also develop the core tools needed to be a successful engineer.

The current iteration of DEN 161 has three sections of interest to this body of work: data analysis using Microsoft Excel, data analysis using Python, and embedded programming using an Arduino Uno.

Microsoft Excel is used to introduce the concept of data analysis to students. Students are tasked with manipulating data and finding different statistical properties of given data. This proves to be an effective point of entry given to data analysis since many students are familiar with Microsoft Excel or Google Sheets.

Python is then introduced as an alternative method of solving the same problems. The lectures and assignments focus on data calculations to verify designs and general code inspection to understand the how the program works. These lectures are done using Jupyter Notebook in Anaconda's Spyder IDE.

Following the introduction to Python, a Stoplight Activity is assigned. This project introduces students to circuitry and microcontrollers through the creation of a stoplight using 3 LEDs and an Arduino Uno. The Arduino Uno is programmed using C++ in the Arduino IDE.

3.2 Project Redesign

Thanks to the solid programming core created by the instructors, the proposed changes are minor, and result in no changes to the current curriculum. Two changes are proposed: the adoption of Visual Studio Code as a development environment and the migration from Arduino Unos to Raspberry Pi Picos.

The class currently uses an IDE by the name of Spyder. Spyder is a popular IDE for data science applications and has to ability to seamlessly integrate with Jupyter Notebooks. However, Spyder does not have the ability to work with a MicroPython device, such as the RPi Pico. Visual Studio Code, on the other hand, has an extension that integrates Pico controls directly into the interface, making it a one-stop-shop for both the data analysis and embedded systems development in DEN 161. Visual Studio Code also has Jupyter Notebook extensions that allow for a first class experience.

The second proposed change is transitioning from the Arduino Uno to a Raspberry Pi Pico. The reason for this change is twofold. First, the Pico can run using MicroPython, a lightweight implementation of Python, which has the same syntax as Python. This allows students to focus on understanding one language, Python, rather than learning both Python and C++. Switching to the Pico also opens the door to using a single development environment. While the Arduino can be programmed using Visual Studio Code, the set up process is non-trivial, and requires a strong understanding of the operating and file system of the computer.

The proposed changes aim to increase student understanding by reducing the number of systems they are introduced to. Instead of two langauges and two editors, students will only need to learn one language in one editor. The work done by these projects directly correlates with Abet Student Outcomes 6 and 7 and weakly correlates with Outcomes 1 and 3, as seen in Appendix C.

3.3 Project Redesign Assessment

The project proposal for DEN 161 is unique from the other classes in this work because the proposed changes were implemented into the standard class curriculum. The second week of Python was replaced with the tire RPM problem and the Arduino Uno was replaced by the Raspberry Pi Pico in the stoplight project. The adoption was largely successful, and the instructors felt that it was a step in the correct direction. That is not to say that there were no pain points.

The biggest hinderance encountered in the Fall 2023 was general file system comprehension. Many students did not know where downloaded files go or how the file system (both File Explorer and Finder) were structured. This proved to be an issue for both the tire RPM question and the stoplight

example. For Python to locate the .csv file with the RPM data, either the full path to the file needs to be provided, relative to the current working directory, or the file needed to be in the same folder as the script. To get the MicroPico extension to work correctly in Visual Studio Code, a working directory needs to be set.

Both of these issues can likely be solved with the same two methods. First, time will be spent in a previous lecture to talk about the file system. This is basic information that is required to effectively use a computer. Second, files for assignments will be distributed in zip files. This will nearly guarantee that files have the correct relative location. This will introduce a new issue of trying to link and edit files in a zip folder, but hopefully it is an easier fix than trying to hunt down files on student's laptops.

The Fall 2023 semester also found that using Visual Studio Code for both the data analysis and Pico portions of the class is better for students than using Spyder for the data analysis and VSCode for the Pico.

An additional pre-class assignment will also be created that will require students to verify that they have done the necessary installations before coming to class. Though this is not an issue directly caused by the changes made, it did directly inhibit student learning.

Lastly, and most concerningly, this semester showed that the MicroPico extension can be unreliable when not used exactly as intended. While no connection issues ever occurred during testing, a non-trivial number of students experienced issues with the MicroPico extension. Since no issues have been encountered by the instructors at this point, it is hard to pinpoint the exact cause. Fixes for different issues can be found in the "usage-and-installation" folder in the GitHub Repository.

3.4 Project Deliverables

In the GitHub repository associated with this paper, which can be found in Appendix A, the folder titled "engineering-problem-solving," there are several folders containing different problems and examples. The folder also contains a README that details what is in each file and what software is needed to complete the assignments.

The first folder is a tire RPM example problem that is to be completed in class, after having completed the Excel lectures and the introduction to Python lectures. This will show students some of the benefits of using a

language like Python over Excel. To go along with the example problem is a similar tire RPM homework assignment. The code needed to complete the assignment is all provided to students, with a handful of tasks left up to them, as detailed in the homework file.

Finally, the stoplight folder contains the starter code to give students for the stoplight assignment, a solution key, and a file demonstrating morse code on the Pico. Installation instructions for the using the Pico can be found in the "usage-and-installation" folder in the repository.

Chapter 4
ME 513: Thermodynamics

4.1 Course Overview

ME 513: Thermodynamics is a lecture based class that focuses on the interplay of heat, work, and energy in both open and closed loop systems. While there is no lab portion to the class, significant lecture time is spent solving problems and working examples, and a new homework is assigned every lecture to give students a chance to apply the new concepts taught that day.

Most problems in thermodynamics follow a similar solution path. First, students must recognize the different states in the system. Once the different states are identified, the values for temperature, power, energy, entropy, etc. must be found for each state. The exact values needed, and retrievable, vary depending on the given system. This process is known as setting the state. Once the state has been set, known laws of thermodynamics can be applied and the question can be solved.

For most questions, the majority of time is spent setting the state. This involves using known properties, usually given in the problem statement, to look up values in property tables in a thermodynamics textbook. This often proves to be a very time consuming process, especially when interpolation is required to get property values. And once iteration is introduced, such as in a design problem, manually searching through tables becomes impractical, if not impossible.

This is where the use of a programming language would become highly beneficial. As it currently stands, the class makes no use of programming or any computer software for solving problems in thermodynamics.

4.2 New Assignments

Since no assignments currently make use of programming, two new asssignments have been created to demonstrate how programming could be beneficial to the course. Both assignments will make use of a Python library called PYroMat. This is a free, open source library dedicated to making thermodynamic properties readily available. They will also both make use of Jupyter Notebooks for cleanly presenting questions, commentaries, and code while

solving the question.

The first assignment, Question 8.23, is a five state system that does not require iteration or plotting. It follows the structure of a typical question in thermodynamics well. The student must identify the states, list the known properties for each state, and then either search for additional values in a table or calculate them with known values. After this, the laws of thermodynamics are applied, equations are balanced, and the question is solved.

The second assignment, Question 8.33, better showcases the benefits of programming. This question uses both iteration and plotting to show the changes in quality and thermal efficiency as the pressure increases. If done by hand, this would be a difficult and tedious task. But when programming, it only requires a few extra lines of code.

Using programming in this manner gives students a better idea of how real problems are solved by introducing them to a more efficient and powerful solution method. These questions would also directly correlate to Abet Student Outcomes 1, 6, and 7 and weakly correlate to Outcome 3, as seen in Appendix C.

4.3 Project Deliverables

In the GitHub repository associated with this paper, which can be found in Appendix A, the folder titled "thermodynamics" contains both the problem statements and solution guides for the two questions introduced in the previous section. The folder also contains a README that details what is in each file and what software is needed to complete the assignments.

For these projects to be added to the class, the instructor would simply need to give the skeleton files to students as a problem statement. It may be beneficial to use question 8.23 as an in-class example both to serve as a reminder of how to use Python with Jupyter Notebooks as well as a demonstration of how to use PYroMat to access material properties and change property units.

Chapter 5

NE 495: Elements of Nuclear Engineering

5.1 Current Implementation

No usage of programming (as of when I took it, need to reach out to current instructor). Q-value assignments were large emphasis and all done by hand

5.2 Project Redesign

Create new homework assignments that require interpolation and iteration of q-values and require programming to iterate through them. Have a standard library with valuesthey can import and use.

5.3 Redesigned Project Deliverable

Chapter 6

ME 400: Computer Applications in Mechanical Engineering

6.1 Course Overview

ME 400: Computer Applications in Mechanical Engineering is a lecture based class with one lab per week. The lectures initially emphasis learning the basic tools of programming, looping, functions, classes, etc., while the remainder of the class focuses on embedded programming and microcontroller hardware, which is the primary objective of the class.

To complete labs, students use an ESP32, a powerful microcontroller made popular by the home automation community, programmed with C++, an industry standard in embedded applications. Since the class does not have any circuit prerequisits and does not teach, nor have the time to teach, circuit theory, the class instructors have created PCBs that are plug-and-play for students to use. These custom PCBs are designed to have the ESP32 as well as several other electronic components plug directly into it to give them the ability to work together. Additional componenets include a small LCD touchscreen, buttons, LEDs, and buzzers.

The labs in ME 400 take an iterative approach where subsequent labs will add features and functionality on top of the previous week's lab assignment. This gives students the experience of completing a relatively complex project without overwhelming them from the start.

ME 400 serves as the primary introduction to programming in the mechanical department. It is also the only class that spends significant teaching about programming and the only class that gives instructions on microcontrollers. Unfortunately, the class comes late in the curriculum, after many classes that would benefit from using programming, and focuses the majority of its time on embedded programming, often to the detriment of a fundamental understanding of programming. This is a topic of much debate, and will be addressed in the concluding chapters.

6.2 Project Redesign

Since ME 400 is a programming course, no new assignments are being created for this project. Instead, a pre-existing lab has been translated from C++ on the ESP32 to MicroPython on the Raspberry Pi Pico. The translated lab exercise is the concluding assignment in a string of lab exercises dedicated to programming the game "Simon Says."

The goal of the exercise is to create a replayable, memory-based game that imitates the children's game Simon Says. TO do this, students have to use the LCD screen, five buttons, four LEDs, a buzzer, and an IR remote. The lab exercise chosen aims to demonstrate how the Pico and MicroPython can handle the same projects and electronics that the ESP32 and C++ can.

To use the LCD screen and the IR remote, drivers needed to be created for the Pico. The drivers provided in the repository are modified versions of existing drivers from https://github.com/rdagger/micropython-ili9341 and https://github.com/peterhinch/micropython_ir. The modified drivers would be given to students and no modifications would need to be made to the drivers.

As previously mentioned, the class uses several custom PCBs to facilitate the use of different circuit componenets. However, no circuit board was designed for this project, so the wirings were all done by hand. Only small changes to hole locations and traces would be needed to adapt the current PCBs into a useable form for the Pico.

Since no new assignments are being added to the class, the learning objectives for the class do not change.

6.3 Project Deliverables

In the GitHub repository associated with this paper, which cna be found in Appendix A, the folder titled "computer-applications-in-me" contains the rewritten lab assignment, code skeleton, and solution for the exercise mentioned above. The folder also contains a README that details what is in each file and what software is needed to complete the assignments. Installation instructions can be found in the "usage-and-installation" folder in the GitHub repository.

The integration of these curriculum changes would not be a simple task. Since the entire class is structured around C++ and the ESP32, every lecture,

assignment, quiz, and test would have to be translated. In addition to this, new PCBs would need to be designed for use with the Pico.

Chapter 7
ME 571: Fluid Mechanics

7.1 Current Implementation

Bonus section since it is an elective. Three main projects in the class: wall following, line following, and free drive. Favorite is the maze solver, but might be realistic to do free drive since the others require getting a set up to test them

7.2 Project Redesign

switch from the arduino mega to pico and use micropython instead. Only concern is the number of gpio pins on the pico. might not have enough for everything that has to be used for maze solver

7.3 Redesigned Project Deliverable

Chapter 8

ME 533: Machine Design

8.1 Current Implementation

Currently no usage of programming in machine design. Graphs are drawn by hand and tables are used to find youngs modulus or inertia values.

8.2 Project Redesign

Make homework assignments that require iteration to solve and would therefore be tedious without programming. Computer graphed functions for deflection curves

8.3 Redesigned Project Deliverable

Chapter 9

ME 535: Measurements and Instrumentation

9.1 Current Implementation

Nearly all labs use programming or hardware for something. Position and motion lab might be the most interesting to convert. Want to change one that uses labview because that requires significant change

9.2 Project Redesign

Much the same as before. Replace any arduino usage with pico and replace labview with python code. Write code functions for students or make them parse data themselves? XOD.io?

9.3 Redesigned Project Deliverable

Chapter 10
ME 570: Control of Mechanical Systems I

10.1 Course Overview

ME 570: Control of Mechanical Systems is a mixed class with two lectures and one lab every week. The lectures focus on teaching the principles and theory of controls while the lab focuses on designing and testing control systems. Most labs are dedicated to using and designing PID controllers using different methods, such as frequency analysis or pole anaylsis.

To complete the labs, the class makes use of a custom motor rig called the motorlab. To go along with the motorlab, a graphical user interface for controlling the motorlab was also created by Dale Schinstock. This GUI runs as an appliation through MATLAB and allows the students to both send commands to the motor as well as collect runtime data from the motor.

Controls also relies heavily on MATLAB for the completion of labs and homeworks. All work is done using .mlx files through a licensed copy of MATLAB for a more interactive working environment. To access the controls related commands, a package called the Controls Toolbox has to be purchased from MATLAB.

While Controls uses programming more than any class besides ME 400, little to no instruction is provided on how to program in MATLAB. The first lab is dedicated to taking the MATLAB Onramp starter course, which takes about 1 to 2 hours to complete. However, the Onramp does not address most of the work done in Controls, such as plotting and transfer functions. The Onramp does teach students the basics of MATLAB syntax, but most students come out of the Onramp just as confused by MATLAB as they were going in.

This disconnect between students and MATLAB becomes a barrier that prevents students from understanding controls. Many students spend more time trying to understand MATLAB than they do learning controls.

10.2 Lab Assignment Redesigns

Since ME 570 already fully utlilizes programming in the course, no new assignments are being created. Instead, three pre-existing labs have been trans-

lated from MATLAB to Python. While MATLAB is the industry standard when it comes to control systems, the license is expensive, students do not have a solid understanding of it, and we do not make use of its most powerful feature, Simulink. In addition to this, Python has a library, aptly named "control," which has a MATLAB module that imitates the Control Toolbox, both in form and functionality, from MATLAB. In combination with Jupyter Notebooks and a few custom plotting modifications, using Python will give a very similar expereince to MATLAB, and will give students that pursue the field of controls any easy transition to MATLAB.

All three translated labs make use of the motorlab, which has had its GUI turned into an application, courtesy of the MATLAB Compiler. This allows the application to be run from an app icon or from the terminal. These labs also make use of a custom plot command. The plot command highjacks the standard matplotlib plot command and adds the ability to create data tips. The direction of the datatip changes based on which mouse click is used, and a double click in the plot window will remove all datatips. More custom functions could be added later to imitate MATLAB functionality as needed, but this was the only one required for completing these labs.

The first translated lab, Lab 4, is a standard assignment that makes use of the motorlab, reading from the csv file output from the motorlab, and then plotting the results. The second lab, Lab 10, builds on Lab 4 by adding root locus plots and sisotool. While the sisotool command in MATLAB is not as powerful as the full designer in MATLAB, it does allow for interactive plots where poles can be moved and the graphs automatically update. The third lab, Lab 13, focuses on frequency response design methods by making use of Bode plots.

The labs chosen aim to showcase how the major features used in MATLAB can be emulated in Python. Some features, like Simulink, have no direct correlation. However, Simulink is only used in one lab, and students only need to make two small changes. Python also requires more library imports, but these can all be handled by the skeleton and do not pose much concern.

Since no new assignments are being added to the class, the learning objectives for the class do not change.

10.3 Project Deliverables

In the GitHub repository associated with this paper, which can be found in Appendix A, the folder titled "control-of-mechanical-systems" contains the lab assignments, code skeletons, and solutions for the three lab assignments explained above. The folder also contains a README that details what is in each file and what software is needed to complete the assignments.

The act of integrating these labs into the class would not be as simple as the integrations for other classes. Since every lab uses MATLAB, the other 11 labs would also need to be translated. In addition to this, every computer in the lab would need to get the correct applications, extensions, and motorlabGUI executable installed. Instructions for installing everything needed can be found in the "usage-and-installation" folder in the GitHub repository.

Chapter 11

ME 573: Heat Transfer

11.1 Current Implementation

Project for designing heatsink exists. Can be solved in any language, python is a solid contender. Every assignment requires looking through the back of a book for table values

11.2 Project Redesign

No change to project needed, can be solved with python as is. Show how tables can be made in a library

11.3 Redesigned Project Deliverable

Chapter 12
Conclusion and Future Work

12.1 Concerns

Something about all hardware used and all software packages used? That might be more appropriate somewhere else?

Future work includes creation of libraries that have all the table data from text books. Some exist already, but not all of them. Might be better to have an in-house collection.

12.2 Recommendations

Appendix A
Project Repository

https://github.com/mKiloLA/python-based-mne

This will also contain a printed copy of the readme from the git repo once I have written it.

Appendix B
Software Versions

Through the entirity of this paper, the following application, package, and extension versions are used:

- Applications
  - Python 3.12.0
    - Optionally, Python 3.11.5 through Anaconda 2023.09
  - Visual Studio Code 1.85
- Python Packages
- Visual Studio Code Extensions
  - Python v2023.22.1
  - Pylance v2023.12.1
  - MicroPico v3.5.0
  - Jupyter v2023.11.1003402403
  - Jupyter Keymap v1.1.2
  - Jupyter Notebook Renderers v1.0.17
  - Jupyter Cell Tags v0.1.8
  - Jupyter Slide Show v0.1.5
  - IntelliCode v1.2.30
  - IntelliCode API Usage Examples v0.2.8
  - vscode-pdf v1.2.2
  - Excel Viewer v4.2.58

Appendix C
Abet Student Outcomes

The following exerpt is taken directly from K-State's website:

Student outcomes describe what students are expected to know and be able to do by the time of graduation. These relate to the knowledge, skills and behaviors that students acquire as they progress through the program. The mechanical engineering program will enable students to attain the following, by the time of graduation:

1. an ability to identify, formulate, and solve complex engineering problems by applying principles of engineering, science, and mathematics

2. an ability to apply engineering design to produce solutions that meet specified needs with consideration of public health, safety, and welfare, as well as global, cultural, social, environmental, and economic factors

3. an ability to communicate effectively with a range of audiences

4. an ability to recognize ethical and professional responsibilities in engineering situations and make informed judgments, which must consider the impact of engineering solutions in global, economic, environmental, and societal contexts

5. an ability to function effectively on a team whose members together provide leadership, create a collaborative and inclusive environment, establish goals, plan tasks, and meet objectives

6. an ability to develop and conduct appropriate experimentation, analyze and interpret data, and use engineering judgment to draw conclusions

7. an ability to acquire and apply new knowledge as needed, using appropriate learning strategies.

https://www.mne.k-state.edu/academics/accreditation/