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

Talk about what classes currently utilize programming and electronics and what languages/hardware they use. Discuss experience as instructor of controls/design 1/2 where students do not know how to code at all - even simple scripting (this makes it hard to teach controls or design when students do not even know how to operate the platform they are learning on). Also talk about the main components that will be used to complete the projects: pico, vscode, python

Issues: instructor integration, instructors need to understand python and programming environments

1.2 Background

1.3 Scope of Work

1.4 Structure of Work

Since this body of work does not set out to

Chapter 2
A Cohesive Programming Curriculum

2.1 Curriculum Proposal

For the purposes of cohesive programming curriculum in the Mechanical and Nucelar Engineering Department at KSU, proposes the use of Python in conjunction with library created for the MNE department,

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 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

Dean of Engineering 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 Proposed Changes

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 Notebooks 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 student outcomes 6 and 7 and weakly correlates with outcomes 1 and 3 as seen in Appendix B.

3.3 Project Deliverables

The repository contains in-class examples, assignments, and solutions for data analysis in Excel, data analysis in Python, and programming the Raspberry Pi Pico. It also contains an installation guide, source code for the custom extension pack, and videos walking through the setup and completition of the different assignments. Included below is an instructors guide that walks through the assigments and files in the repository.

3.3.1 Instructors Guide

Change the order to start with basic Python, which teachers already have, then the excel problem and the python version of the csv file. Then move to using the pico and ciruit building

1. **Introduction to Excel**

    - The content for the introduction to Excel already exists and does not need to be updated.

2. **Introduction to Python**

    - The content for the introduction to Python already exists and does not need to be updated.

3. **Data Analysis with Excel**

    - **In-Class Example:** Using the data in tire_rpm_excel.csv, find the maximum, minimum, mean, median, and mode speed of the vehicle. Assume a tire diameter of 20 inches. Graph the speed of the vehicle at any given time.
    - **Homework:** Have students repeat the process using the data from tire_rpm_homework.csv and 18" wheels. Graph the speed of the vehicle at any given time. This should be a 1-to-1 copy of what was done in class, just with different numbers.

4. **Data Analysis with Python**

    - **In-Class Example:** Using the data from tire_rpm_example.csv and the Jupyter Notebook intro_to_python_example.ipynb, find the maximum, minimum, mean, median, and mode speed of the vehicle. Assume a tire diameter of 20 inches. Graph the speed of the vehicle at any given time. This should give identical results to the Excel example problem.
    - **Homework:** Have students create a .py file that finds the maximum, minimum, mean, median, and mode speed of the vehicle using tire_rpm_homework.csv. Assume a tire diameter of 18 inches. Graph the speed of the vehicle at any given time.
        - **Extra Credit:** How long did it take a car with 22" wheels to go 0-60 if the sensor data was taken at 300Hz?

5. **Programming the Raspberry Pi Pico**

    - **In-Class Example:** Walk through the code in example.py to show students how to blink the LEDs.

- **Homework:** Task students with altering the code provided in class to make the LEDs function as a stoplight. A potential solution is provided in stoplight.py.
  - **Extra Credit:** Make the LEDs spell your name in morse code. An example of looping morse code is shown in sos.py.

3.3.2 Description of Files in the Repository

See Appendix A for full source code and documentation. Videos have been removed from the repository due to file size limitations.

1. **installation_guides**
   - **Installation_Guide.pdf:** a guide that walks through downloading Anaconda, Visual Studio Code, and the KSU Extensions in Visual Studio Code.
2. **intro_to_excel**
   - **tire_rpm_example.csv:** a data file that contains RPM data for a car wheel. Use this data for the example questions.
   - **tire_rpm_homework.csv:** a data file that contains RPM data for a car wheel. Use this data for the homework questions.
   - **tire_rpm_example_solution.xlsx:** a potential solution to the in-class problem posed in Step 1.
   - **tire_rpm_homework_solution.xlsx:** a potential solution to the homework problem posed in Step 1.
3. **intro_to_python**
   - **tire_rpm_example.csv:** a data file that contains RPM data for a car wheel. Use this data for the example questions.
   - **tire_rpm_homework.csv:** a data file that contains RPM data for a car wheel. Use this data for the homework questions.
   - **intro_to_python_example.ipynb:** a Jupyter Notebook file that walks through solving the in-class example problem. This file is intended to bridge the gap between Excel and Python.

- **intro_to_python_homework_solution.py:** a Python script for solving the homework question from Step 2. This could also be done in a Jupyter Notebook, but using a .py file was used to showcase standard Python usage. This also contains the solution to the extra credit question.

4. **intro_to_pico**

    - **example.py:** LED flashing program for the RPi Pico. This file is intended to be used as the in-class example in Step 3 and the base code provided for the homework.
    - **stoplight.py:** this is a potential solution to the Stoplight Activity. Many different variations of this file could exist.
    - **sos.py:** this is an example of using morse code with the Pico. The solution utilizes looping and a function to reduce repeated code.

5. **ksu_den_161_extension_pack**

    - **package.json:** this file contains the code used to create the Extension Package in the Microsoft Marketplace. As it stands, this file (and folder) can be ignored. In the future, an instructor will need to make sure the extension pack stays up to date.

Chapter 4
ME 513: Thermodynamics

4.1 Current Implementation

No projects currently exist. Currently any assignments require looking through tables in back of book endlessly. Interpolation is done by hand.

4.2 Project Redesign

Create new homework assignments (or a semester project?) that requires iteration of a property that would typically be overly tedious without software help. Could potentially utilize curve fitting for better interpolation than straight linear interpolation.

4.3 Redesigned Project Deliverable

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 Current Implementation

Waiting on projects from Dr. Brockhoff still. Uses mostly C++ and Arduino Megas / ESP32s for projects. Minimal python to end the semester.

6.2 Project Redesign

Change the class to use only python, increase the learning period at the beginning. Pick one project (maybe the buzzer one because it would be fun) and replicate it using pico and micropython.

6.3 Redesigned Project Deliverable

Chapter 7

ME 533: Machine Design

7.1 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.

7.2 Project Redesign

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

7.3 Redesigned Project Deliverable

Chapter 8

ME 535: Measurements and Instrumentation

8.1 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

8.2 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?

8.3 Redesigned Project Deliverable

Chapter 9

ME 570: Control of Mechanical Systems I

9.1 Current Implementation

Interesting one. Probably want to change one assignment and one lab. The lab requires figuring out if the motorlab can be integrated into python and vscode through serial. Should be possible.

9.2 Project Redesign

Change a homework assignment that needs matlab and use python for it. Change a lab that needs the motorlab and use python instead of matlab

9.3 Redesigned Project Deliverable

Chapter 10

ME 573: Heat Transfer

10.1 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

10.2 Project Redesign

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

10.3 Redesigned Project Deliverable

Chapter 11

ME 615: Applications in Mechatronics

11.1 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

11.2 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

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

Appendix B
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/