Instrumentation Lab   Lab 1 – Introduction to the PicoW   Fall 2024

Names ____

Introduction

My edit of the Wikipedia entry for *Instrumentation* is

> *Instrumentation* is a collective term for measuring instruments, used for indicating, measuring, and recording physical quantities. It is also a field of study about the art and science about making measurement instruments. The term has its origins in the art and science of scientific instrument-making.
>
> Instrumentation can refer to devices as simple as direct-reading thermometers, or as complex as multi-sensor components of complex control systems.

The focus of this course is on building scientific instruments. Almost all modern scientific instruments have a microcontroller at their core. Any instrument with a control panel, a display, WiFi, Bluetooth, etc. has a microcontroller in it. In this course you will learn to one way of making instruments with a microcontroller.

This course focuses on the Raspberry Pi Pico W (I call a Pico W) microcontroller as the main device that you will use to control experiments, and take data. In the first few labs you will work through the book *Get started with MicroPython on Raspberry Pi Pico, The Official Raspberry Pi Pico Guide, 2nd edition*. I will call this book *Getting Started*. It is available through Amazon as a Kindle, so you need an Amazon account to get the book. You can read it on a laptop at the *Cloud Kindle* website **`https://read.amazon.com/?asin=B0CW1DRBTV`** or on the Kindle app on you laptop or smarthphone. The Pico W that has WiFi and Bluetooth capability.

You will be programming the Pico W in a version of Python called MicroPython. Yes, these small boards can run a limited version of Python. This makes the programming much easier to learn. Two more advantages of MicroPython is you can run it interactively using the REPL prompt, **`>>>`**, and it has a nice **`help`** function to give you information.

Today you will work through the first three chapters

1. Soldering pins to your PicoW and mounting it on a breadboard. Chapter 1.
2. Installing MicroPython (latest version) on your PicoW. Chapter 1.
3. Start programming your PicoW using MicroPython. Chapter 2.

Activity 1 – Soldering the Headers to the PicoW

Note: you may have a PicoW with the headers already connected. Please read the pages anyway.

- First read locations 65 – 240, stopping before *Installing MicroPython*.
- Solder the header pins to your Pico W.
- When you are finished, show the instructor the PicoW soldered to pins. Figure 1-14 and the paragraphs below shows how soldering can go wrong and how to correct it.
- Mount it in a breadboard with row 1 of the breadboard and pin 1 on the PicoW in the same row. The USB connector should be sticking out at the of the breadboard.

Activity 2 – Programming Your PicoW with MicroPython

- Download the latest version of µPy (MicroPython) from **`https://micropython.org/download/RPI_PICO_W/`**
  - It will be the first link under **Firmware** *Releases*. The name will be something like **`v1.23.0 (2024-06-02).uf2`** / [Release notes] (latest)
- Follow the directions to load MicroPython on you Pico W.
- The next step, in Chapter 2, is installing the *Thonny* IDE (Integrated Development Environment.) It is a small stand-alone editor that allows you to read and write files to the PicoW. Follow directions in the box at location 328 to change the mode to **Regular**.

Names ____

- Continue reading.
- Write your first µPy program on the PicoW. It will print the message *Hello, World!*. Save it on the Pico W with the name **`hello_world.py`**.

Important Notes on saving your code files:

- Give your programs names that mean something related to what the program does. Don't name them **`program1.py`**, etc.
- I don't like spaces in my file names, so use underscores, _, instead of spaces.
- *Never have the only copy of your program only on the Pico W! Make a folder on your laptop, navigate Thonny to that folder and save a copy of your program on your laptop, also. You especially want to do this at the end of lab, and after you have a working copy of a program.*

THE CHALLENGES

Throughout the book there are *Challenges* you have to complete.

Important Tip: You *always* want to save your most recent code *both* on the *MicroPython* device *and* on your laptop! So, here are my suggestions:

1. Make the file names descriptive, for example **`Hello.py`**, or **`Challenge1.py`**.
2. Save the first time on the PicoW.
3. Make a folder on your laptop for your code. Make a subfolder named **`Lab01`**. Save the files for Lab 1 in this folder.
4. Make sure you save the file with the same name in your Lab 1 folder.

Challenge 1 (loc. 476): *Loop the Loop*. Try printing the numbers from 1 to 9 in a 3x3 square using a loop inside a loop. If you want more than one print statement on the same line, add , **`end=' '`** at the end of the print statement and **`print()`** at the end of the inner loop.

Challenge 2 (loc. 510): *Add More Questions*. Ask the user for a number, then print out too high or too low, or end the game if they guess 5. Hint: To get an integer from the user you need to change the input line to

```
num_guessed = int(input("Enter a number: ")
```

The **`int`** function converts the text to a number.

Challenge 3 Complex numbers. Yes, these little µPy microcontrollers know complex numbers! You enter a complex number, for example, as **`2-3j`**, which has a real part of **`2`** and a complex part of **`-3`**. To enter a pure complex number, put a **`j`** at the end, like **`3.14j`**. To input a complex number use a statement like:

```
cmplx2 = complex(input("Enter a complex number: "))
```

Write a program that inputs two complex numbers and prints their sum, product, and magnitude (try **`print(abs(1+1j)`**.)