Text Files & Energy in a Battery

ECE 476 Advanced Embedded Systems

Jake Glower - Lecture #20

Please visit Bison Academy for corresponding
lecture notes, homework sets, and solutions

Introduction:

The Pi-Pico has 264k on-chip SRAM. This allows you to

- Create a text file which controls the Pi-Pico's operation
- Write to a text file, saving your data

This lecture covers

- How to open and close text files
- Reading from text files
- String commands and parsing strings
- Reading a text file to play a tune
- Writing to text files, and
- Measuring the energy in a rechargeable battery

Rechargeable Batteries from Amazon: How much energy to they *really* have?

Opening & Closing Text Files

Opening a file: The general syntax to open a file in Python is:

```
file = open("File_Name", "Access_Mode")
```

Access Mode can take on several values:

Access mode	Function
"r"	Default mode Open a text file for reading. Pointer is placed at the start of the file. Results in an error if the file does not exist
"a"	Open a text file for appending. Pointer is placed at the end of the file. Creates a new file if it does not exist
"w"	Open a text file for write-only. Create a new file if one does not already exist. Clear out the contents of the existing file.
"x"	Create a new file Returns an error if the file already exists

The file can also be specified as a text file or a binary file (i.e. an image)

File Type	Function
"t"	Text file (default)
"b"	Binary file (image)

Closing a file: Once finished, files should always be closed

```
file.close()
```

Reading From a Text File

Text files are read as strings - regardless of whether the contents are actually numbers or text. When you read a text file, you can read some or all of the file

Command	Result
`Data = f.read(5)`	Read the next five characters into text string *Data*
`Data = f.readline()`	Read the next line into *Data*
`Data = f.readlines()`	Read the entire file into an array *Data*. Each line is stored in a different entry: Data[0], Data[1], etc
`Data = f.read()`	Read the entire file into a text string, *Data* Carriage returns and line feeds show up as /n/r

For example, assume a text file contains the following information:

readme.txt

```
Three rings for the Elven-kings under the sky
Seven for the Dwarf-lords in their halls of stone,
Nine for the Mortal Men doomed to die
```

This file can be read in its entirety

Program Window

```
f = open("readme.txt", "rt")
Data = f.read()
print(Data)
f.close
```

Shell

```
Three rings for the Elven-kings under the sky
Seven for the Dwarf-lords in their halls of stone,
Nine for the Mortal Men doomed to die

>>> Data
'Three rings for the Elven-kings under the sky\r\nSeven
for the Dward-lords in their halls of stone\r\nNone for
the Mortal Men doomed to die\r\n'
```

Note that

- The file is stored as a text string
- \r is a carriage return
- \n is a newline command

You can also read this file line by line

Program Window

```
f = open("readme.txt", "rt")
Data = f.readlines()
n = len(Data)
for i in range(0,n):
    print(i, Data[i])
f.close
```

Shell

```
0 Three rings for the Elven-kings under the sky

1 Seven for the Dwarf-lords in their halls of stone,

2 Nine for the Mortal Men doomed to die

>>> Data[0]
'Three rings for the Elven-kings under the sky\r\n'
```

String Commands and Parsing Strings

https://www.w3schools.com/python/python_strings.asp

One way to pass data to a Python program is through a text file.

- list of numbers to graph
- list of music notes to play a song.

Typically, the data is separated by commas, spaces, or tabs

- Search for these to find the fields

Example: National Sea and Ice Data Center (NSIDC):

```
# Arctic Sea Ice Extent
# https://nsdic.org/arcticseaicenews/sea-ice-tools/
1979  7.051  16.342
1980  7.667  16.041
1981  7.138  15.632
:
```

Parsing Text Files

- pull out fields

readlines()

- reads in an entire line

strip()

- removes spaces at the start and end

replace()

- replaces tabs and commas with spaces
- replaces double spaces with single

find()

- locate where the spaces are
- determines the fields

X[0:m]

- field (type-string)

float()

- Convert to a floating point number

```
def Parse(X):
    X = X.strip()
    X = X.replace(',',' ')
    X = X.replace('\t',' ')
    for i in range(0,10):
       X = X.replace('  ',' ')
    ncol = X.count(' ') + 1
    Y = [0]*ncol

    for i in range(0,ncol):
        m = X.find(' ')
        if(m>0):
            Y[i] = float(X[0:m])
        else:
            Y[i] = float(X)
        X = X[(m+1):]
    return(Y)

Data = '1979  7.051  16.342'
Y = Parse(Data)
print(Y)
```

```
[1979.0, 7.051, 16.342]
```

Plotting a Text File

- SeaIce.txt has three columns
  - Year, min(Ice), max(Ice)
- Read in the file
- Plot ice level vs. year

Y is read as a Nx3 matrix

- Transpose to pull out columns

Plot displays the data

- scaled to max & min

```
import LCD
import matrix

def Parse(X):
    :

f = open("SeaIce.txt", "rt")
Data = f.readlines()
f.close()

n = len(Data)
Y = []
for i in range(0,n):
    Y.append(Parse(Data[i]))

Y = matrix.transpose(Y)

Navy = LCD.RGB(0,0,5)
White = LCD.RGB(100,100,100)
LCD.Init()
LCD.Clear(Navy)
LCD.Plot(Y[0],[Y[1],Y[2]])
LCD.Title('Arctic Ice', White, Navy)
```

Resulting Plot

- Arctic sea ice
- Plotting data from a text file

Playing a Tune from a Text File

- Example: Mario Brothers Tune

Text files can also contain a tune to play

Each line contains

- The note,
- The octave, and
- The duration of the note
  - in 16th's of a beat:

file Mario_Bros.txt

```
E4,2
E4,2
E4,4
0,2
C4,2
E4,4
G4,4
0,4
G3,4
0,4
```

Parse Routine

Pull out the note

- First field

Pull out the duration

- Second field

Write a test program

- Check each element of Y
- Contains note and duration

Parse subroutine

```
def Parse(X):
    X = X.strip()
    X = X.replace(',',' ')
    X = X.replace('\t',' ')
    for i in range(0,10):
       X = X.replace('  ',' ')
    m = X.find(' ')
    Note = X[0:m]
    Dur = int(X[(m+1):])
    return([Note,Dur])

f = open("Mario_Bros.txt", "rt")
Data = f.readlines()
f.close()

n = len(Data)
Y = []
for i in range(0,n):
    Y.append(Parse(Data[i]))
print(Y)
```

Shell

```
[['E4',2], ['E4',2], ['E4',4],
['0',2], ['C4',2], ['E4',4],
['G4',4], ['0',4], ['G3',4],
['0',4]]
```

Convert notes to frequency

Note	C0	C#0	D0	E0	F0	F#0	G0	G#0	A0	A#0	B0
Hz	16.35	17.32	18.35	20.6	21.83	23.12	24.5	25.96	27.5	29.14	30.87

Start with the frequency

- Assume zeroth octave

Scale by 2**n

- n = octave

Write a test routine

- Verify frequencies are correct

Freq subroutine

```
def Freq(a):
    n = len(a)
    Note = a[0:n-1]
    Octave = a[n-1]
    Hz = 0
    if(Note == 'C'):
        Hz = 16.35
    elif(Note == 'C#'):
        Hz = 17.32
    elif(Note == 'D'):
        Hz = 18.35
    :
    if(Hz > 0):
        Hz = Hz * (2 ** int(Octave))
    return(Hz)

print('A3 = ', Freq('A3'), ' Hz')
print('D4 = ', Freq('D4'), ' Hz')
print('G#5 = ', Freq('G#5'), ' Hz')
```

shell

```
A3 = 220.0 Hz
D4 = 293.6 Hz
G#5 = 830.72 Hz
```

Playing a Tune

Finally, reuse the *Play(Hz, Dur)*

- Hz = frequency
- Dur = duration in 1/16 beat

Spin through a text file to play a tune

- More impressive in the video

```
    :
def Play(Hz, Eighths):
    if(Hz > 0):
        Spkr.freq(round(Hz))
        Spkr.duty_u16(32768)
    else:
        Spkr.duty_u16(0)
    sleep_ms(75 * Eighths - 50)
    Spkr.duty_u16(0)
    sleep_ms(50)

f = open("Mario_Bros.txt", "rt")
Data = f.readlines()
f.close()

n = len(Data)
Y = []
for i in range(0,n):
    Y.append(Parse(Data[i]))

for i in range(0,n):
    Hz = Freq(Y[i])
    Dur = Y[i][1]
    print(i, Hz, Dur)
    Play(Hz, Dur)
```

Writing to a Text File

You can also write to a text file

- Save data for later analysis
- Voltage of a discharging battery

Open Options

- "a" append to file.
  - Create new file if needed
- "w" clear current file
  - Create new file if needed

File Write

- Writes a text string to a file
- \n = carriage return
- \t = tab

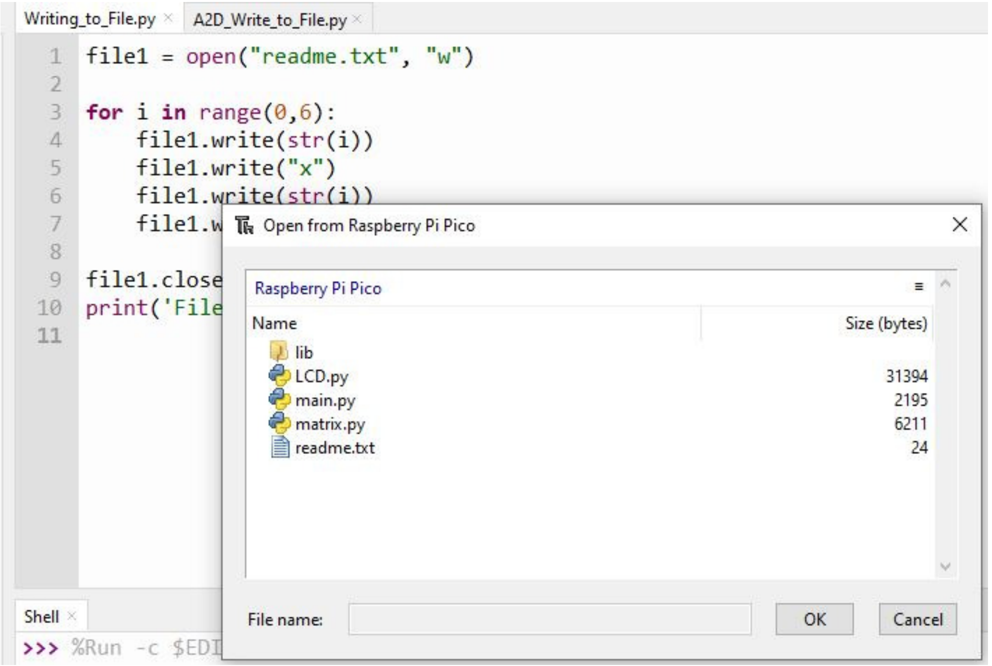
```
file1 = open("readme.txt", "w")
print('File Opened')

for i in range(0,6):
    file1.write(str(i))
    file1.write("x")
    file1.write(str(i))
    file1.write("\n")

file1.close()
print('File Closed')
```

readme.txt

```
0x0
1x1
2x2
3x3
4x4
5x5
```

Note: To open readme.txt, from Thonny,

- Click on File Open
- Select Raspberry Pi Pico
- Select readme.txt



After writing to a file, the file on the Pi-Piico board can be opened using Thonny

Example: Reading A/D channels

- Read three A/D inputs
- Sample every 100ms
  - A/D read takes 100us
  - Write to file takes 1770us
- Write these to a file
  - Separate data with spaces
- Terminate with a carriage return
  - "\n"

```
import machine, time

a2d0 = machine.ADC(0)
a2d1 = machine.ADC(1)
a2d2 = machine.ADC(2)

kV = 3.3 / 65535

file1 = open("readme.txt", "w")

for i in range(0,10):
    V0 = a2d0.read_u16() * kV
    V1 = a2d1.read_u16() * kV
    V2 = a2d2.read_u16() * kV
    file1.write(str(i) + " ")
    file1.write(str(V0) + " ")
    file1.write(str(V1) + " ")
    file1.write(str(V2) + "\n")
    time.sleep(0.1)
file1.close()
```

file readme.txt

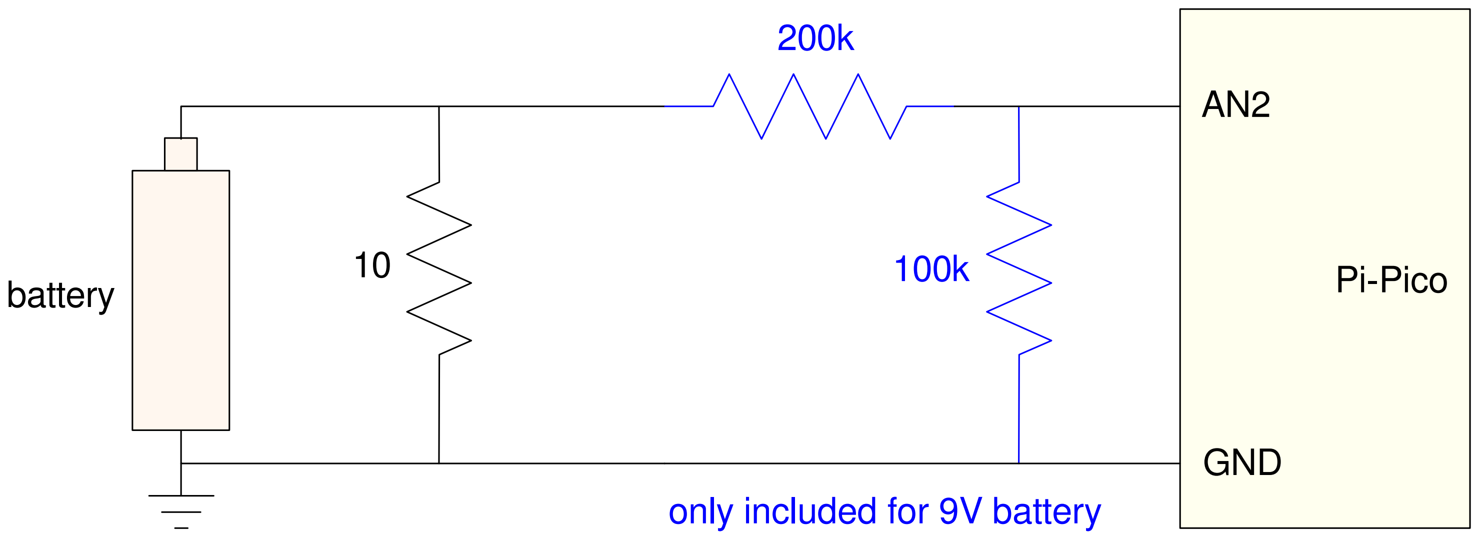
```
0  1.3925  1.4231  0.0556
1  1.3893  1.4215  0.0548
2  1.3869  1.4231  0.0556
3  1.3901  1.4231  0.0548
:
```

Energy in a Battery: Hardware

Next, to measure the energy in a rechargeable battery,

- Connect the battery to a 10 Ohm resistor, and
- Measure the voltage with a Pi-Pico
- For 9V batteries
  - Add a divide by 3 circuit
  - increase R to 47 Ohms

$$I = \frac{V}{R} \qquad P = \frac{V^2}{R}$$



Expected Battery Life:

Based upon battery rating, the time of the experiment should be:

Battery Type	Voltage	mAh (rated)	R (Ohms)	mA @ R	Hours
AAA	1.5V	750	10	150	5.00h
AA	1.5V	2,400	10	150	16.0h
9V	9.0V	600	47	191.48	3.13 h

One reason to save the data to a text file

- I don't want to wait around for 16 hours
- If you save data every second, 16 hours is 57,600 data points
- Pi-Pico doesn't have that much memory

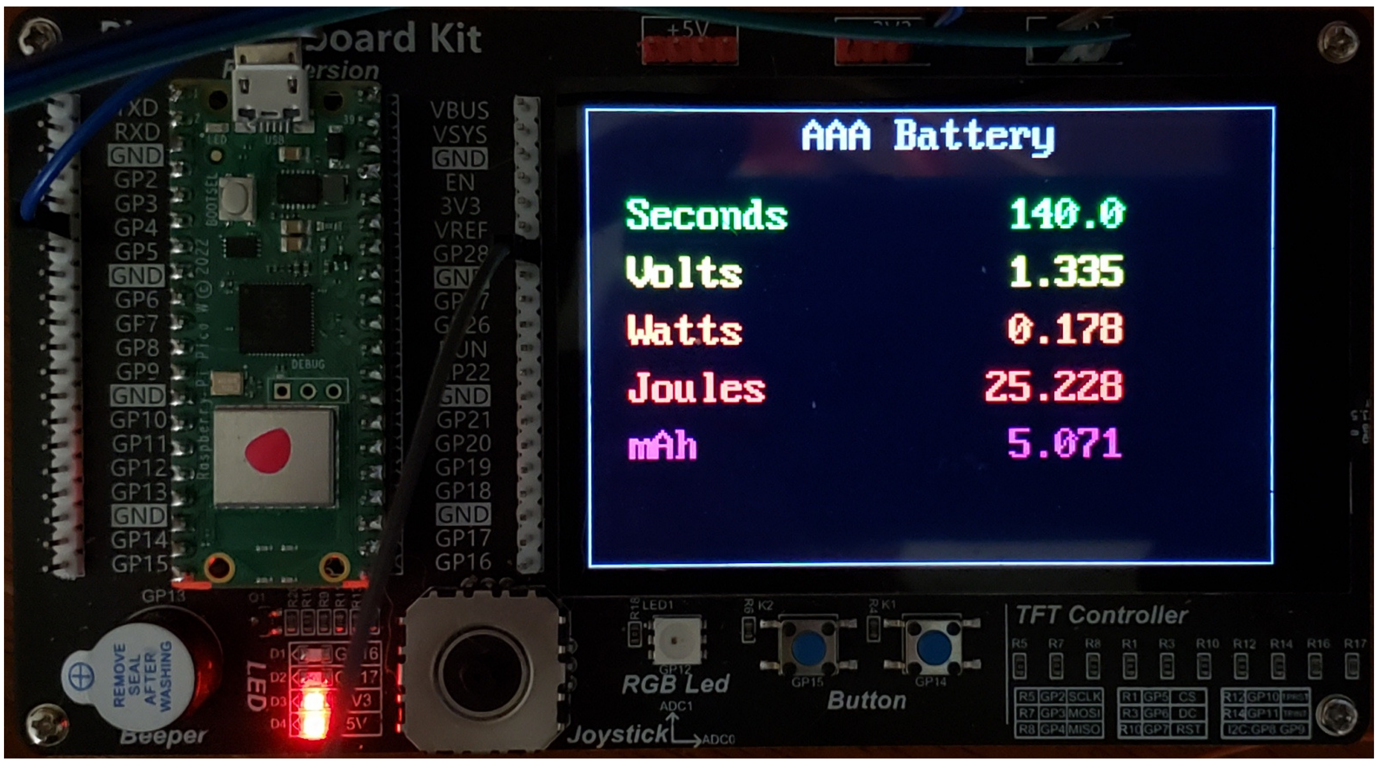
Energy in a Battery: Software

Measure the voltage every second

- Display on the Pico board
- Save to a file every 60 seconds

Stop when < 0.5V

- Battery is discharged



Code Listing

- abbreviated
- full code on Bison Academy

Interupts set the sampling rate

- one second

Every second

- Measure the voltage
- Compute Watts, Joules, mAh
- Display every second
  - Write to file every 60 seconds

When done

- Close the file

```
def tick(timer):
    global flag
    flag = 1

T = 1
Time = Timer()
Time.init(freq = 1/T,
mode=Timer.PERIODIC, callback = tick)

file1 = open("Battery_Test.txt", "w")

while(Volts > 0.5):
    while(flag == 0):
        pass
    flag = 0
    Volts = (a2d2.read_u16() * kV)
    mA = Volts / 10 * 1000
    mAh += mA * T / 3600
    Watts = ( Volts ** 2 ) / 10
    Joules += Watts * T
    file1.write(str(time))
    file1.write(str(Volts))
    file1.write(str(mAh))
    file1.write(str(Joules))
    file1.write("\n")
    time += 1
file1.close()
```

Rechargeable AAA Battery

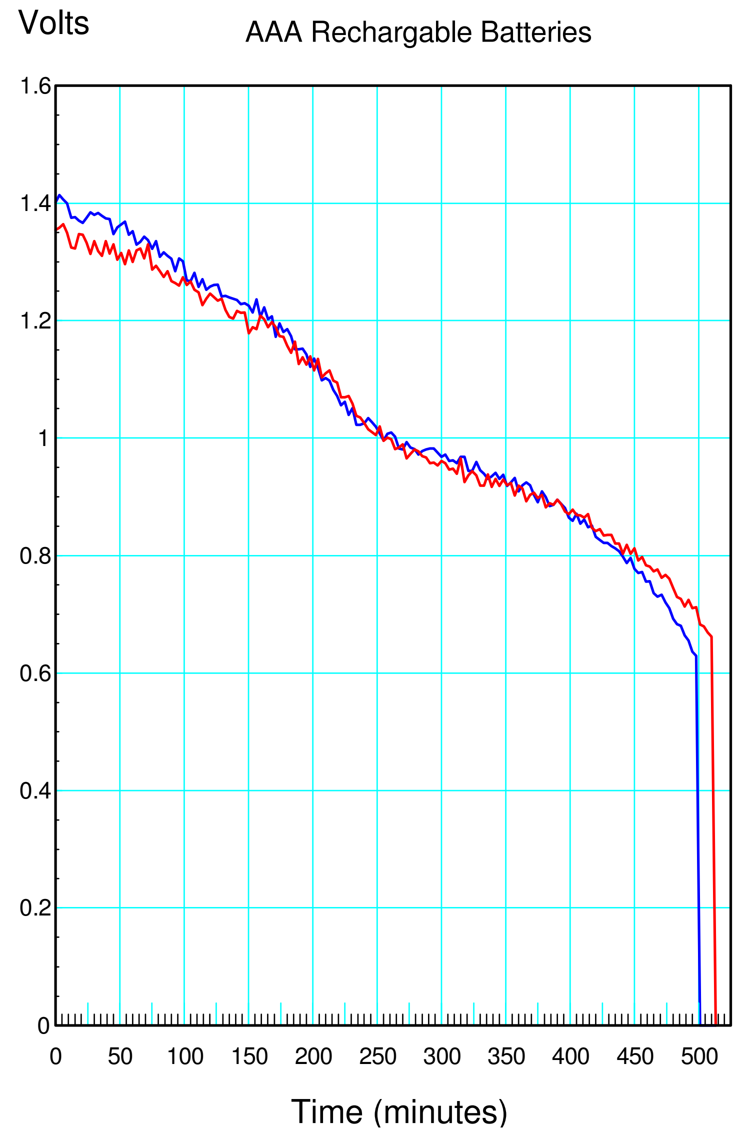
- Rated Energy: 750mAh

Experiment:

- Fully chargethree AAA batteries
- One by one, place in a battery holder
  - Discharge across a 10 Ohm resistor
- Record the voltage until it drops below 0.5V

Results:

Battery	mAh	Joules
1	879.53	3486.54
2	886.80	3444.23
3	880.86	3547.31



Analysis: Find

- The 90% confidence interval for mAh
- The probability mAh > 750
  - Battery meets manufacturer's claims

Student t-Test

- Analysis of lab data using Matlab
- Finite sample size
- Data from a normal distribution

Step 1: Collect the data (done)

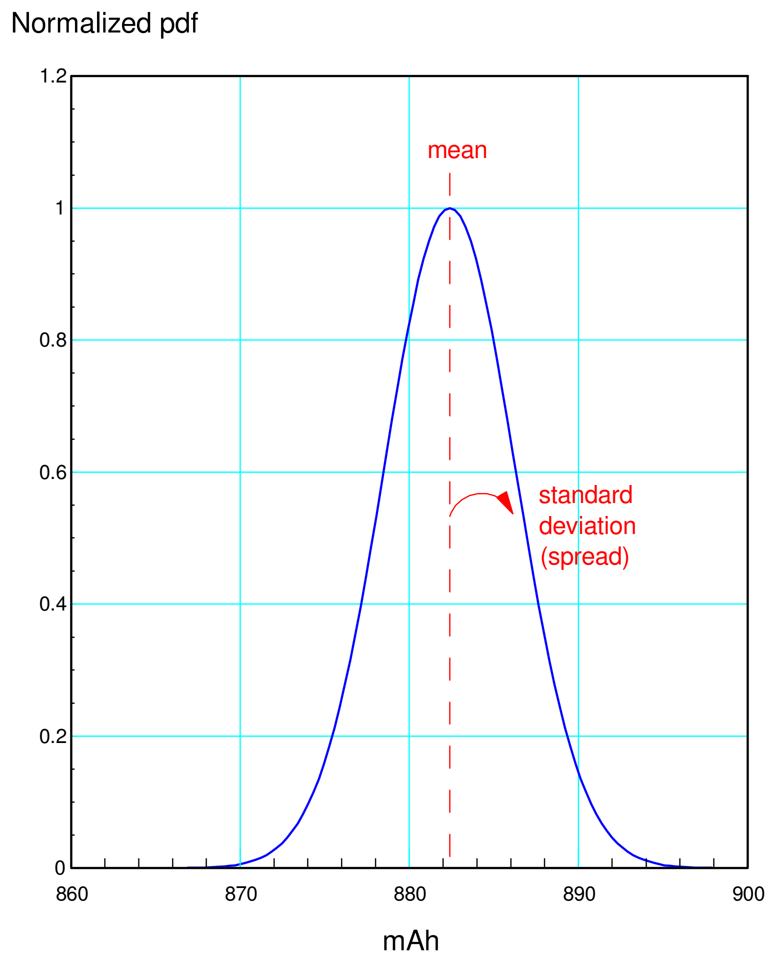
```
>> mAh = [879.531, 886.804, 880.86];
```

The mean and standard deviation are:

```
>> X = mean(mAh)
X =  882.3983

>> S = std(mAh)
S =   3.8729
```

This tells you the pdf

- shown to the right



90% Confidence Interval:

- Two-sided test (you have two tails)
- Each tail is 5%

From StatTrek (Student t-Table)

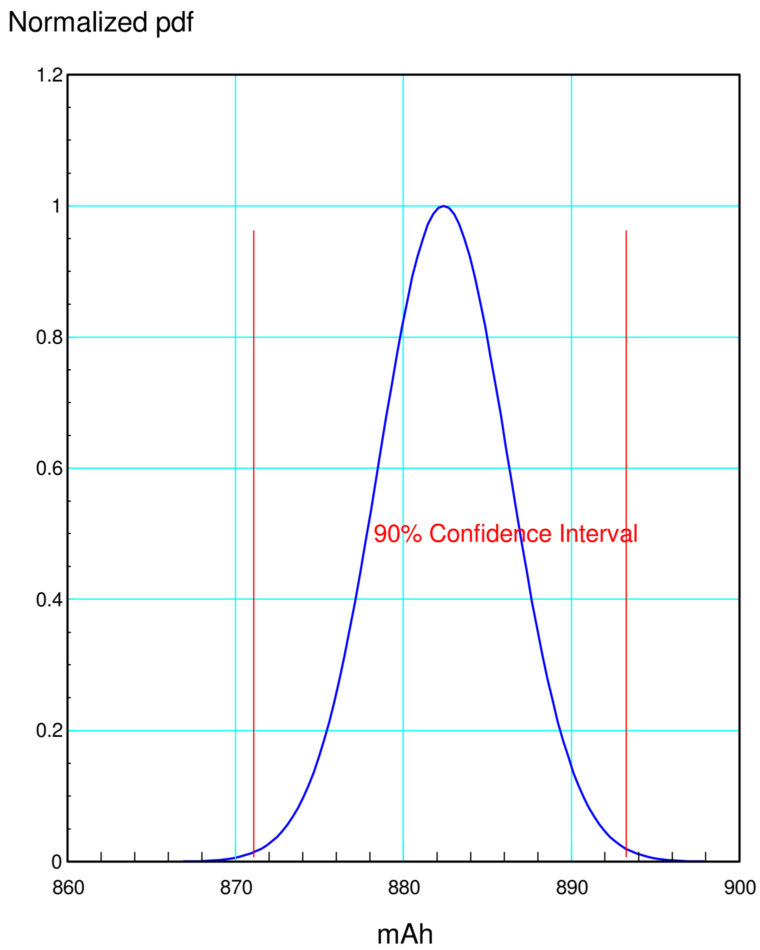
- 5% tails with
- Two degrees of freedom
  - dof = sample size minus one
- t-score = 2.920.

Translation:

- 90% confidence interval:
  - (mean $\pm$ 2.92 st dev)

```
>> X + 2.920*S
ans =  893.7071

>> X - 2.920*S
ans =  871.0896
```

90% of AAA batteries should have (871.09 to 893.71) mAh.



How many batteries meet specs?

- Energy > 750mAh

This is a single-sided t-test

- Find the area of the tail (< 750mAh)

Step 1: Find the t-score

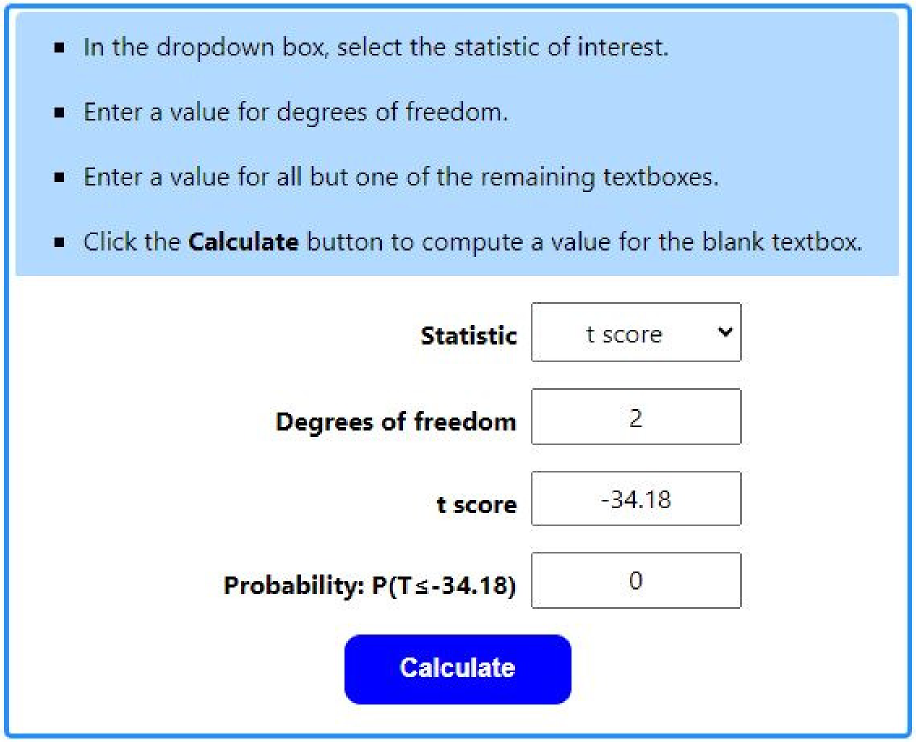
```
>> t = (750 - X) / S
t =   -34.1863
```

Step 2: Convert to a probability

- 2 degrees of freedom
  - sample size minus one
- t-score = -34.1863

From StatTrek

- $p < 0.0005\%$
- Rounded to 0%
- The manufacurer's claim is valie (!)



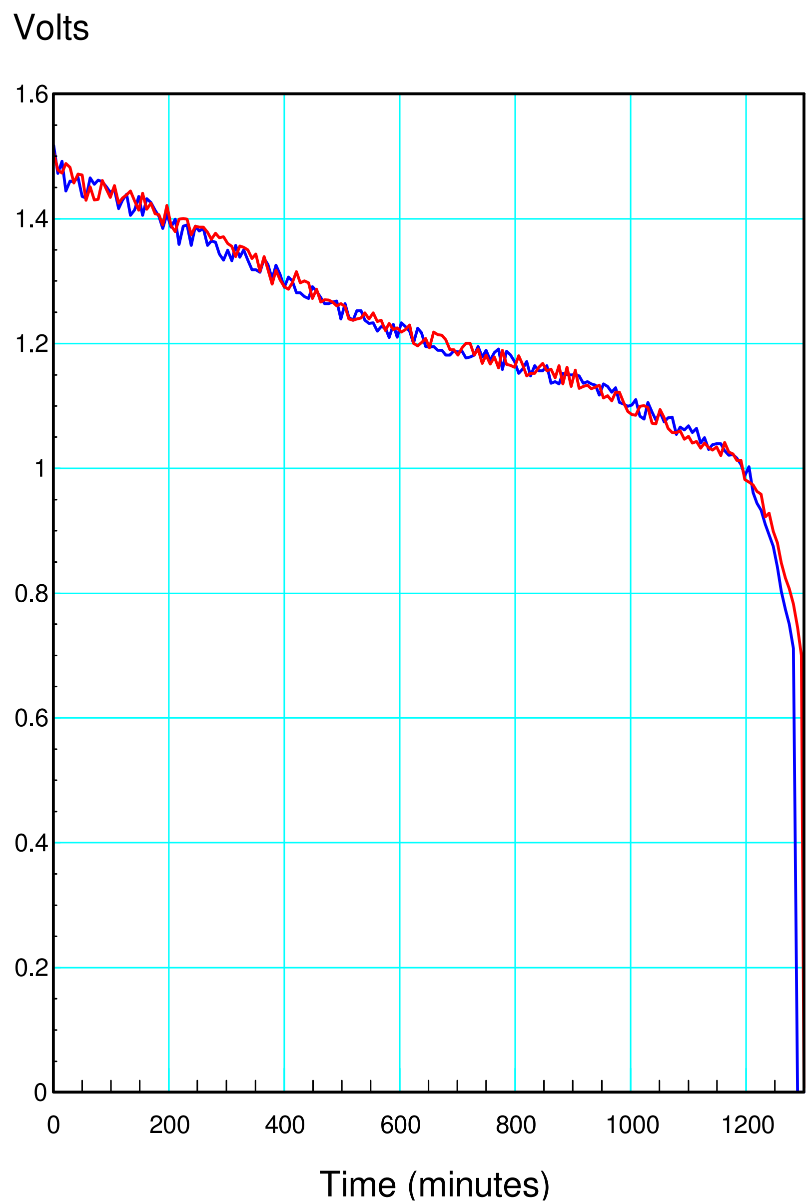
Rechargeable AA Battery

Repeat with a rechargable AA battery

- Rating = 2400mAh
- Discharge across 10 Ohms
- Record votlage, Joules, mAh

Results:

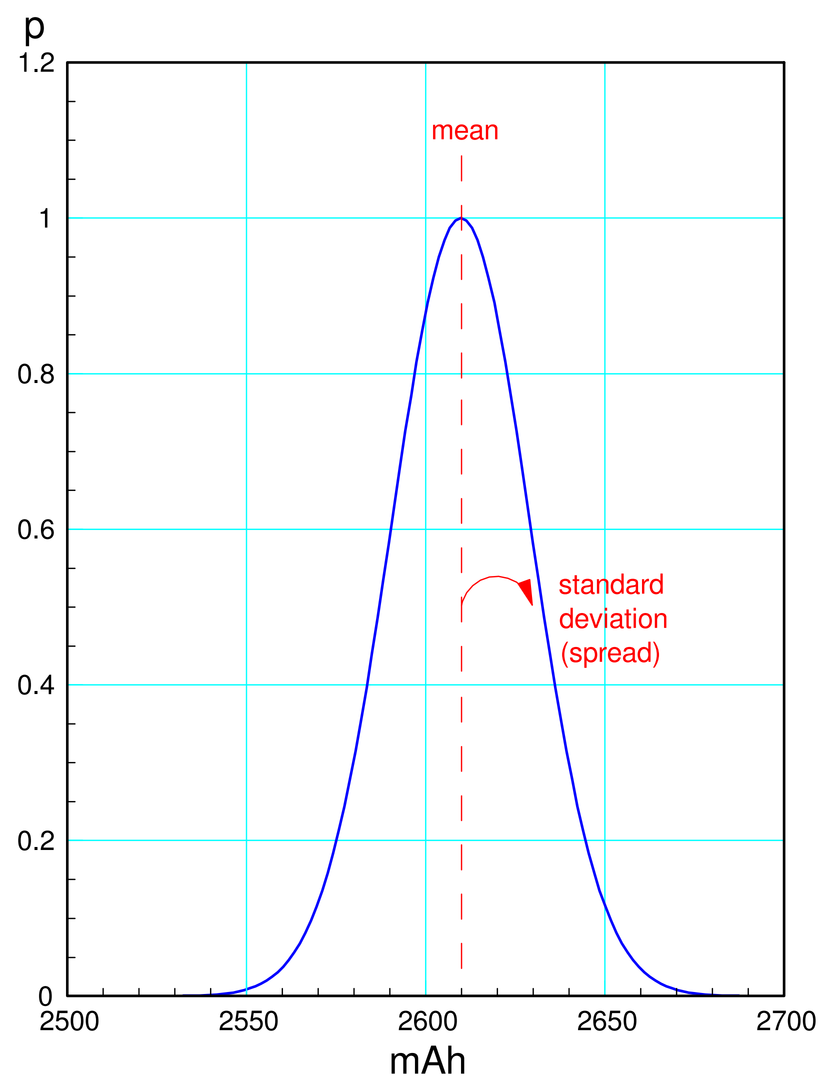
Battery	mAh	Joules
1 (blue)	2,596.1	11,512.723
2 (red)	2,623.5	11,632.354



Following the same procedure as before

- Sample size = 2
- t-score for 5% tails = 6.314

```
>> mAh = [2596.1, 2623.5];
>> X = mean(mAh)
X =  2.6098e+003

>> S = std(mAh)
S =   19.3747
```



90% confidence interval

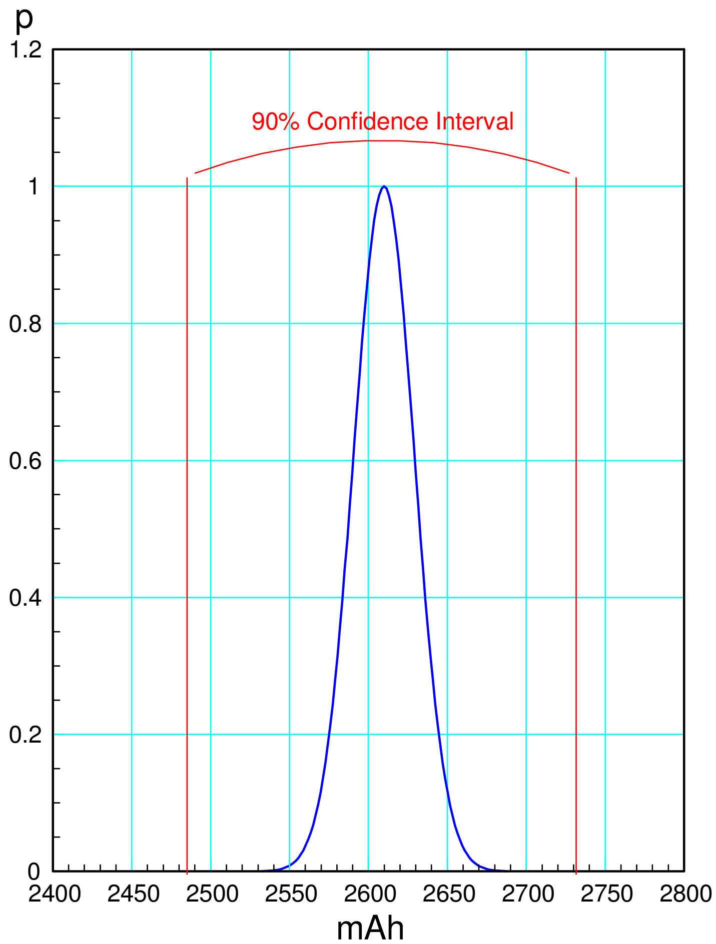
- t-score = 6.314
- Energy: (2487.5mAh - 2732.1mAh)

```
>> X + 6.314*S
ans =  2.7321e+003

>> X - 6.314*S
ans =  2.4875e+003
```

t-score for 2400 mAh

- t-score = -10.8285
- p(tail) = 2.9%
- 97.1% of batteries meet specs

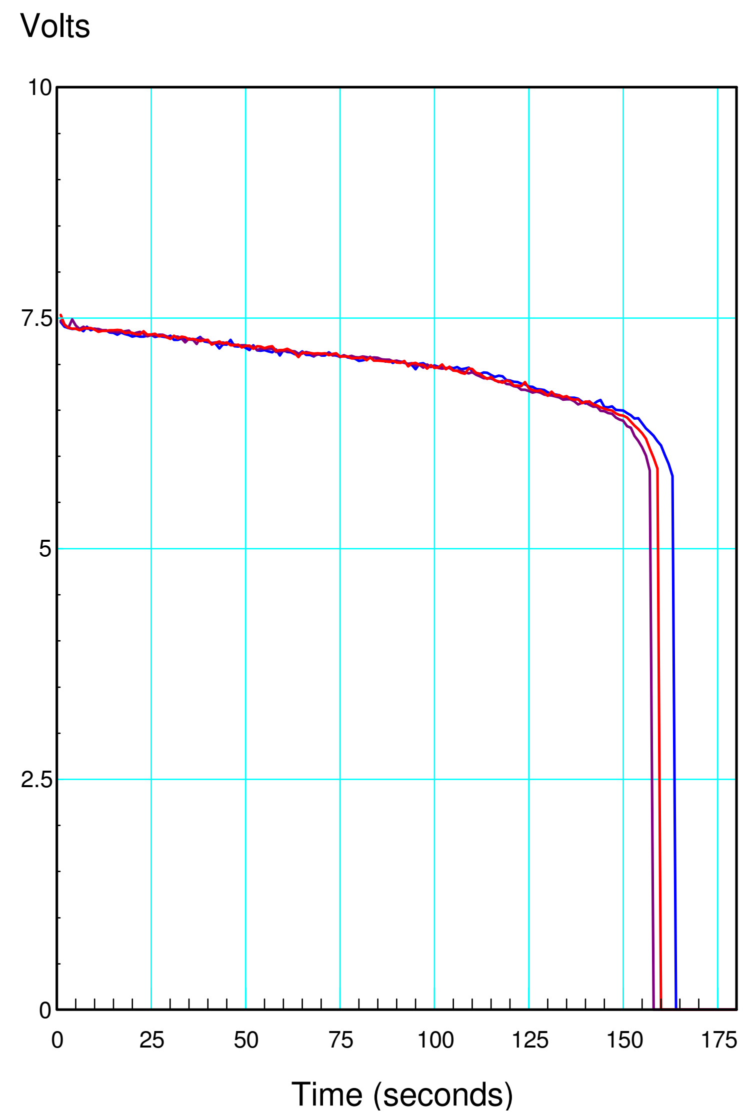
```
>> t = (2400 - X) / S
t =  -10.8285
```



A larger sample size would give better results.

Rechargeable 9V Battery

- Raten energy = 600mAh
- Discharge across 47 Ohms
- Data for three batteries



Results:

Battery	mAh	Joules
1	402.354	10,128.095
2	388.744	9,809.798
3	393.570	9,924.400

t-Tests:

- Energy = (374.74, 415.04) mAh
  - 90% confidence interval
- p(< 600mAh) > 99.9995%
  - Manufacturer's claim is a bit generous.

Summary

Python is able to read from and write to text files fairly easily. With this, you can

- Plot data you recorded earlier,
- Play different tunes by saving data to a given text files, and
- Save data when you collect it for later analysis.

References

Pi-Pico and MicroPython

- https://github.com/geeekpi/pico_breakboard_kit
- https://micropython.org/download/RPI_PICO/
- https://learn.pimoroni.com/article/getting-started-with-pico
- https://www.w3schools.com/python/default.asp
- https://docs.micropython.org/en/latest/pyboard/tutorial/index.html
- https://docs.micropython.org/en/latest/library/index.html
- https://www.fredscave.com/02-about.html

Pi-Pico Breadboard Kit

- https://wiki.52pi.com/index.php?title=EP-0172

Other

- https://docs.sunfounder.com/projects/sensorkit-v2-pi/en/latest/
- https://electrocredible.com/raspberry-pi-pico-external-interrupts-button-micropython/
- https://peppe8o.com/adding-external-modules-to-micropython-with-raspberry-pi-pico/
- https://randomnerdtutorials.com/projects-raspberry-pi-pico/
- https://randomnerdtutorials.com/projects-esp32-esp8266-micropython/