Stephen Duncanson

Computer Engineer

Presentation Overview - 1

- My background, skills, interests.
- Complete with code/technology examples (often!)
- **Implementation**: Cross reference requirements (below) with a quantitative trek through my engineering background (college/research/projects)

Requirements

- BS in engineering or science; computer, software or systems engineering preferred
- Courses and development experience with several of the following: C/C++/C#, Python, VHDL, LabVIEW, MATLAB, Simulink
- Strong hands-on software and hardware debugging skills
- Excellent verbal and written communication skills
- Fast and eager learning, utilizing, and applying new technology

Presentation Overview - 2

Me: UConn Computer Engineering ‘22

Likes: Outdoors, reading, history, running, animals, computers, physics/applied math

Highlighted topics which I have experience with && will be touched on in talk

Requirements

- BS in engineering or science; computer, software or systems engineering preferred
- Courses and development experience with several of the following: C/C++/C#, Python, VHDL, LabVIEW, MATLAB, Simulink
- Strong hands-on software and hardware debugging skills
- Excellent verbal and written communication skills
- Fast and eager learning, utilizing, and applying new technology

Ask questions anytime!

Tell me to speed up/~~down~~/etc!

Summer 2019

Interested in Physics; RACE Coastal Engineering Internship

RACE COASTAL ENGINEERING

- New project management software (Deltek Ajera)
- Old disorganized project files from 1998-2019

openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files

openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files

```python
# Stephen Duncanson

from openpyxl import load_workbook
import waterp

def cols_to_tuples(name_of_file, col_tuple):
    tuple_array = []

    wb = load_workbook(filename = name_of_file)
    ws = wb.active

    # row 1 is header: (latitude, longitude)
    for row_number in range(2, ws.max_row+1):
        tuple_array.append((ws.cell(row=row_number,column=5).value,
                            ws.cell(row=row_number,column=4).value))

    return tuple_array


if __name__ == "__main__":
    coord_array = cols_to_tuples("SiteLocations1.xlsx", [2,3])
    print(coord_array)

    for coord in coord_array:
        if waterp.point_in_water(coord):
            print(coord, " Is in water!!")
        else:
            #print(coord, "is not in water!")
            pass
```

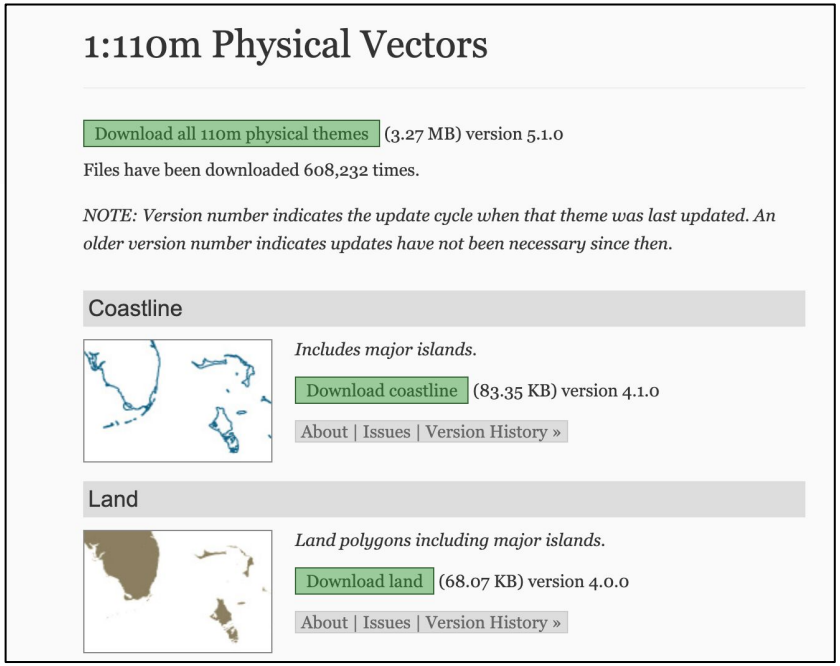
Was not allowed to keep Python scripts that contained information about company filesystem - but here is another example of a script I wrote that used openpyxl

- Highlights the power of Python (imo - pip!)

Brief sidetrack because this is interesting, then back to 2019

PyShp

The Python Shapefile Library (PyShp) reads and writes ESRI Shapefiles in pure Python.



https://www.naturalearthdata.com/downloads/110m-physical-vectors/

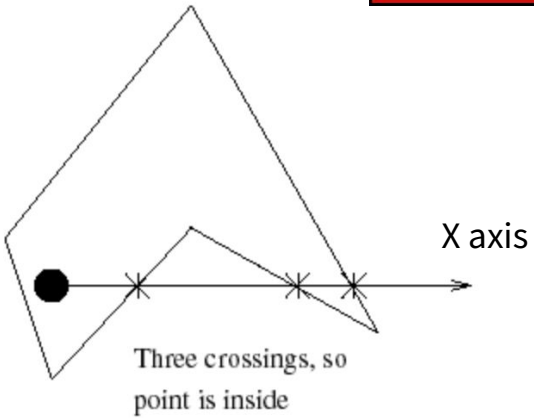


Figure 1 - Crossings Test

Haines, Eric, "Point in Polygon Strategies," *Graphics Gems IV*, ed. Paul Heckbert, Academic Press, p. 24-46, 1994.

PyShp

The Python Shapefile Library (PyShp) reads and writes ESRI Shapefiles in pure Python.

```
# load in landmass shapefile
try:
    # loading .shx file may improve read speeds.
    land_110m_shp, land_110m_dbf = open("ne_110m_land.shp","rb"), \
        open("ne_110m_land.dbf","rb")
    print("Landmass shape file and dbf loaded!")
except:
    print("Missing ne_110m_land.shp or ne_110m_land.dbf files")
    sys.exit(2)

# load the shapefile using pyshp
sf = shapefile.Reader(shp=land_110m_shp, dbf=land_110m_dbf)
shapes = sf.shapes()
# print(len(shapes)) # 127 <- how many polygons we need to iterate through
```

```
n = len(polygon)

for i in range(n):
    i1 = i+1
    if i == n-1: i1 = 0

    if polygon[i][1]*polygon[i1][1]<0:
        # only +*- multiplication yields a negative product.
        # edge crosses x axis
        f = lambda p1, p2 :  p1[0] + p1[1]*(p2[0]-p1[0]) / p1[1]-p2[1]
        r = f(polygon[i],polygon[i1])
        # r is x coord of intersection

        if r > 0: # crosses positive x-axis
            if polygon[i][1] < 0:
                w+=1
            else:
                w-=1
    elif polygon[i][1] == 0 and polygon[i][0] > 0:
        # v_i is on the positive x axis
        if polygon[i1][1] > 0:
            w+=0.5
        else:
            w-=0.5
    elif polygon[i1][1] == 0 and polygon[i1][0] > 0:
        # v_i+1 is on positive x axis
        if polygon[i][1] < 0:
            w+=0.5
        else:
            w-=0.5
return w
```

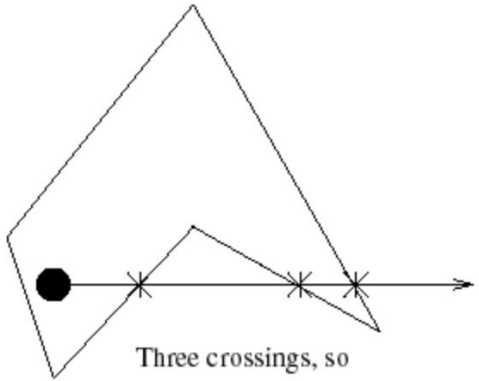


Figure 1 - Crossings Test

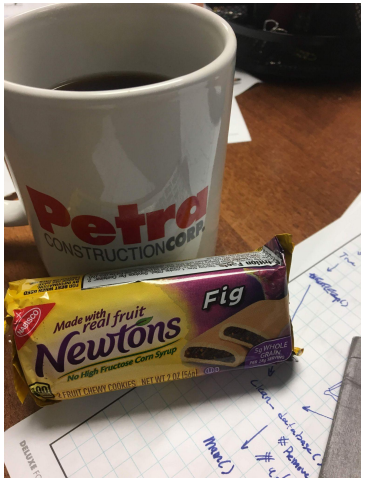
```
# we translate the point to be tested to (0,0)
# we iterate over all vertices v and translate them too
for v in poly:
    polygon.append((v[0]-x[0], v[1]-x[1]))

w = 0 # winding number
```

Back to Summer 2019

openpyxl - A Python library to read/write Excel 2010 xlsx/xlsm files

- New project management software (Deltek Ajera)
- Old disorganized project files from 1998-2019

New problem:
I now have a lot of free time



RACE GIS v1.28

File Help Dev

Years

From 2019 to 2019

Project Types

- [x] All
- [x] Dredging
- [x] Flood / Wave Study
- [x] Hydrographic Survey
- [x] Pier / Dock
- [x] Flood and erosion control structure
- [x] Upland Structure
- [x] Facilities Assessment
- [x] Beach / Dune / Living Shoreline
- [x] Construction Administration
- [x] Other

Client Types

- [x] All
- [x] Residential
- [x] Goverment
- [x] Marina / Yacht Club
- [x] Utility / Industrial
- [x] Consultant
- [x] Contractor

Extras

- [ ] Permit
- [ ] Boring

Location

- [ ] Specify Location — By ID

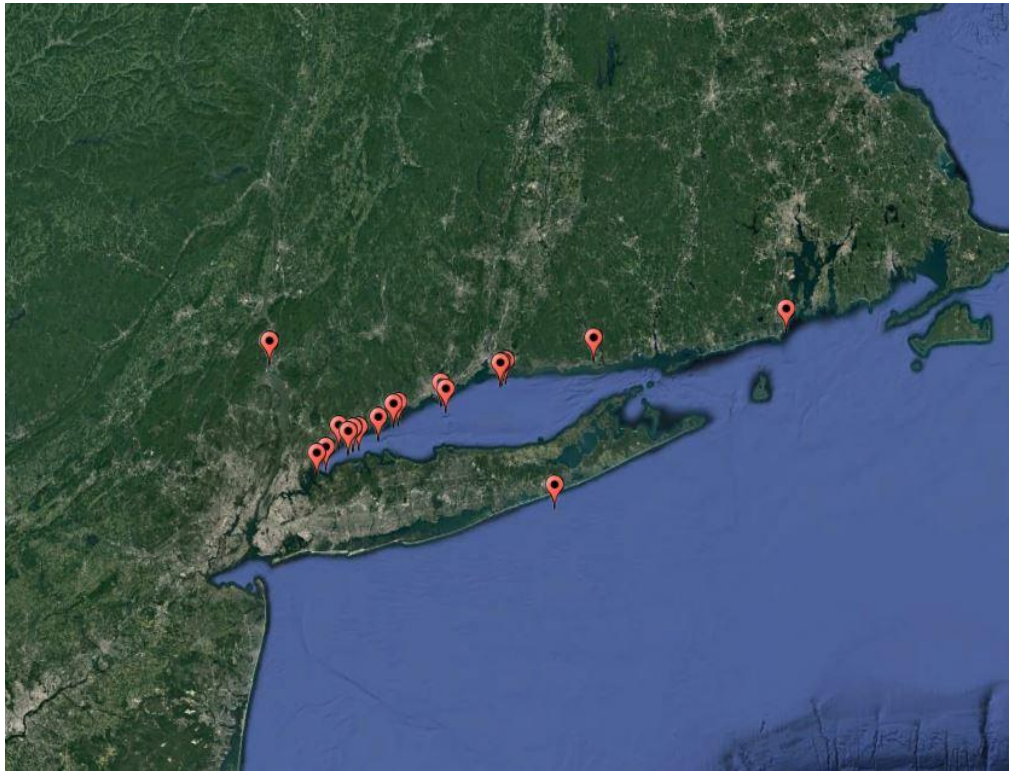
Starting Point

Radius (mi) 10

Google Earth

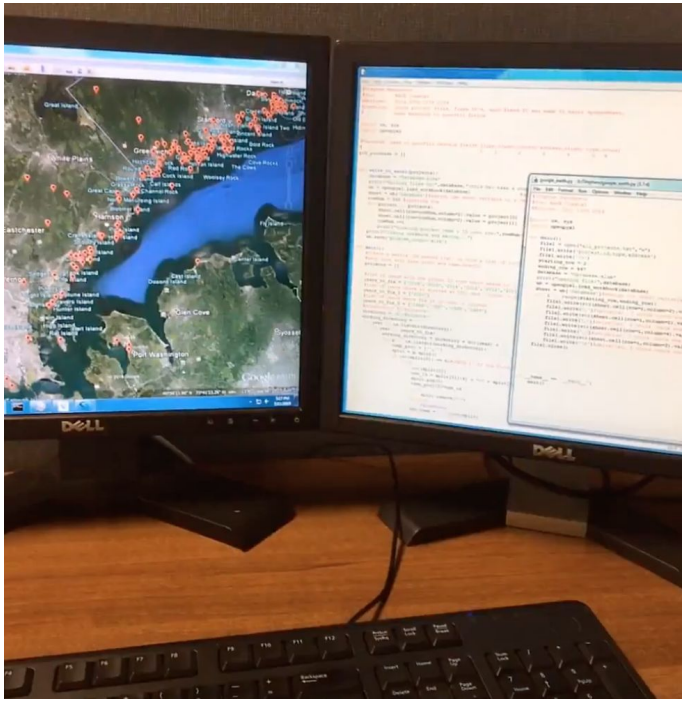
Selecting projects between 2019 and 2019

RACE
COASTAL ENGINEERING



RACE GIS Demo Video

Apologies for Camerawork + Vertical



If time permits:

https://youtube.com/shorts/zvWlKfNkCns?feature=share

https://youtube.com/shorts/xFQW-NPG84k?feature=share

RACE GIS - GUI

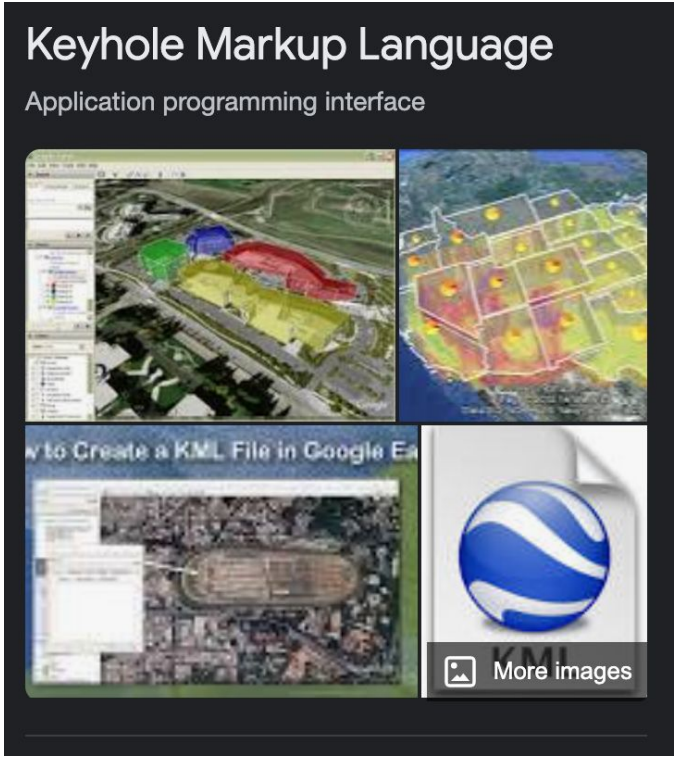
`tkinter` — Python interface to Tcl/Tk

RACE COASTAL ENGINEERING

Improvements? Threading…

```
1263    root = tk.Tk()                                  #Create the window
1264    root.title("RACE GIS v"+str(VERSION))           #Set the titlebar
1265    #YEARS
1266    year_choices = []                               #Create empty list to hold years
1267    year_var_1 = tk.StringVar()                     #Variable to hold start year
1268    year_var_1.set(str(now.year))                   #Set to current year
1269    year_var_2 = tk.StringVar()                     #Variable to hold end year
1270    year_var_2.set(str(now.year))                   #Set to current year
1271    for x in range(1996,now.year+1):                #For all years between 1996 and current year
1272        year_choices.append(x)                      #add to a list
1273    years = ttk.Labelframe(root,text="Years")       #Create the years labelframe
1274    years.grid(column=1,row=0,sticky=tk.W+tk.E,ipadx=2,ipady=2,padx=2,pady=2)                #with 2px of inte
1275    from_label = tk.Label(years, text="From")                                                #Create the 'from
1276    from_label.grid(column=1,row=0)                                                          #Place the label
1277    year_menu_1 = tk.OptionMenu(years,year_var_1, *year_choices,command=refresh_year_list) #Create drop down
1278    year_menu_1.grid(row=0,column=2)
1279    to_label = tk.Label(years, text="to")
1280    to_label.grid(row=0,column=3)
1281    year_menu_2 = tk.OptionMenu(years,year_var_2, *year_choices,command=refresh_year_list)
1282    year_menu_2.grid(row=0,column=4)
```

RACE GIS - Google Earth Export



Improvements? KML exports were hardcoded; would have been more elegant to have some cool parsing

```
<?xml version="1.0" encoding="UTF-8"?>
<kml xmlns="http://www.opengis.net/kml/2.2">
  <Placemark>
    <name>Simple placemark</name>
    <description>Attached to the ground. Intelligently places itself
       at the height of the underlying terrain.</description>
    <Point>
      <coordinates>-122.0822035425683,37.42228990140251,0</coordinates>
    </Point>
  </Placemark>
</kml>
```

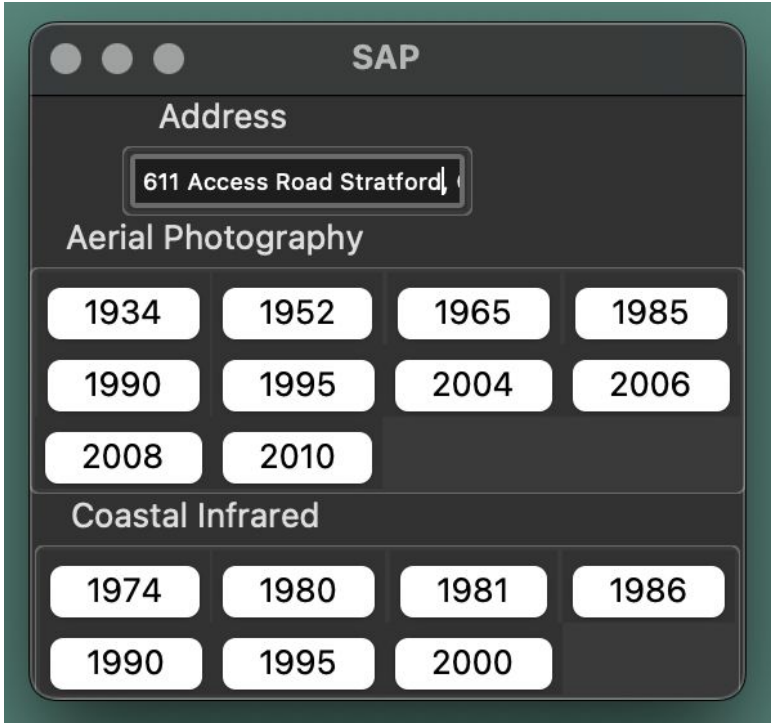
RACE GIS - Google Earth Export

RACE COASTAL ENGINEERING

```
652  def google_earth_button():
653      '''
654      When map button is pressed
655      '''
656      list_of_lat_longs = []
657
658      database_wb = openpyxl.load_workbook(DATABASE_FILE)
659      database_sheet = database_wb[DATABASE_SHEET_NAME]
660      geolocator = GoogleV3(api_key=GOOGLE_API_KEY)
661      export_file = open('RACEGIS.kml','w')#open a text file of the date + rest
662      projects = get_project_types()
663      years = get_years() #list of years to select
664      clients = get_client_types() #list of client types
665      export_file.write(kml_header_1)
666      export_file.write(kml_header)
```

Still Summer 2019

Once again: free time. Permitting is slow; layers, scrolling, can we do better?
“Stephen’s Aerial Program”




```
import os
import os.path
from geopy.geocoders import GoogleV3
from geopy.distance import geodesic

FILE_PATH      = 'txts/'
GOOGLE_API_KEY = ''
YEARS_NUMBER   = ['1934','1952','1965','1985','1990','1995','2004','2006','2008','2010']
years          = []

print('Building database...')
l1934 = []
t1934 = open(FILE_PATH+"1934.txt",'r')
for line in t1934:
    l1934.append(((line.split(' ')[0], line.split(' ')[1].strip('\n'))))
t1934.close()
years.append(l1934)
```

Got a free API key from Google
Used geopy library to calculate distances
Today? This can be queried online API

Aerial Photography

Today? Use a cool data structure like kd-tree to find nearest neighbors fast.

Fall 2019

Physics at UConn; Research in computational soft matter group

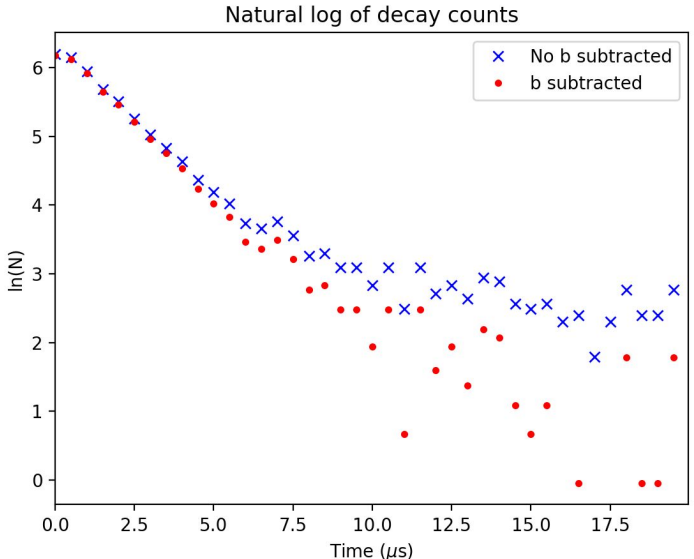
Course		Description	Attempted Credits	Earned Credits	Grade	Grade Points
ENGL	1616	Major Works of Eng & Amer Lit	3.00	3.00	A	12.000
MATH	1132Q	Calculus II	4.00	4.00	A-	14.800
PHYS	1602Q	Fundamentals of Physics II	4.00	4.00	A-	14.800
PHYS	2501W	Advanced Undergraduate Lab	4.00	4.00	A	16.000
PHYS	3989	Undergraduate Research	2.00	2.00	A-	7.400
Note:		*Computer Simulations of Associating Liquid Crystals*				

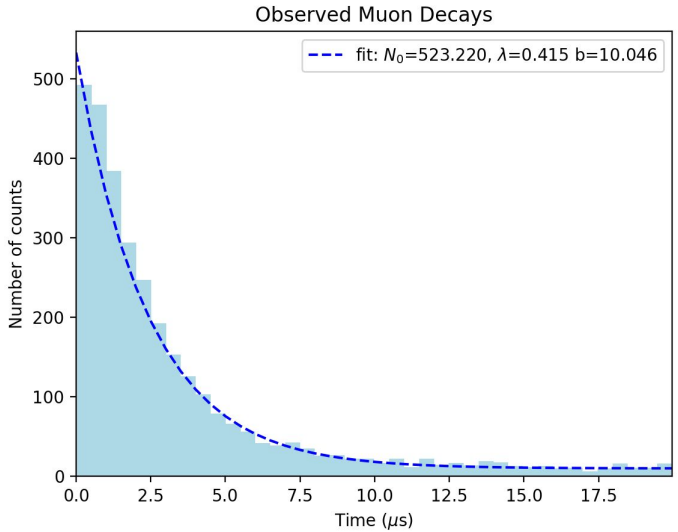
Fall 2019 Applicable Class: PHYS 2501W

I want to discuss my research a bit but first, I really liked this course.

Example from lab 5: Muon decay.

Started using matplotlib to make plots! Learned stats. LaTeX. Writing. Experimental setup. Patience.





Put together the intersection point of the lines of best fit with uncertainty was found to be:

$$t_0 = 1.9930 \pm 0.0008(s) \tag{26}$$

IV. Results

Using the intersection point t_0 calculated in equation 27, the local acceleration due to gravity can be calculated g using equation 12. For L, defined as the distance between suspension points A and B, or as the sum of the distances a and b, a value was stamped on the pendulum itself by the machinist who tooled the instrument. The given value was:

$$0.98783 \pm 0.00001(meters) \tag{27}$$

This given value was deemed more accurate than any measurements I would be capable of doing with the available equipment.

Applying the values from equations 26 and 27 into equation 12 gives a measurement for g.

$$g = 9.8179 m/s^2 \tag{28}$$

The uncertainty must be propagated to obtain the complete results of the experiment. The following equation was utilized to propagate the uncertainties associated with the measurement of length L and the value of the intersection point t_0.

$$\Delta g = g_{raw} \times \sqrt{\left(\frac{\sigma_{t_0}}{t_0}\right)^2 + \left(\frac{\sigma_L}{L}\right)^2} = 0.004 \tag{29}$$

Applying this uncertainty to the raw calculated value for g yields:

$$g = 9.818 \pm 0.004 m/s^2 \tag{30}$$

to minimize differences between experimental and theoretical models of periodic motion. Using this angle, data were collected to linearly relate period with a distance x between the pendulum's mass and end. A correction to compensate for the small angle approximation was applied to these data. A linear least squares fit was done to interpolate a line between the collected data points. From these lines an intersection point t_0 was obtained. This value and the length between the pendulum's pivots were the only requirements needed to obtain a result for gravity which agrees with our current models, some 202 years after the English physicist's original experiment.

[1] P. Deh
[2] D. DiF
gradio
meetin
[3] J. B. L
physics
[4] H. Kat
the len
Philos
pp. 33-
[5] P. Bev
the Ph
Educat

LaTeX

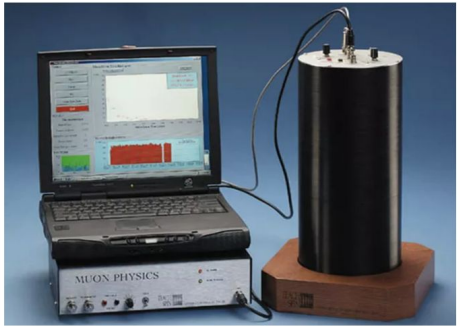


Figure 2. The TEACHSPIN Muon physics apparatus used during this determination.

Do to the fact that many muons will not stop in the detector, the count rate will be quite low, and the time interval to collect any statistically significant amount with grow quickly.

For this reason the muon detecting apparatus was left running for several days to obtain a usable amount of counts.

III. Data Analysis & Results

After two days of collecting the apparatus was shut off and the resulting data was *sifted* using a software program provided by TEACHSPIN. The purpose of this program was to separate erroneous counts from the data algorithmically. As is shown in later analysis, this did not remove all incorrect data.

Below in figure 3 is the raw obtained count data. 40 bins are used here with a bin width of .5 microsecond.

$$N(t) = N_0 e^{-\lambda t} + b \tag{1}$$

Where N_0 is a constant determined by the conditions, λ is the decay constant for the observed muons, and b is a background term helpful in correctly fitting the observed data.

The resulting graph is shown below in figure 4.

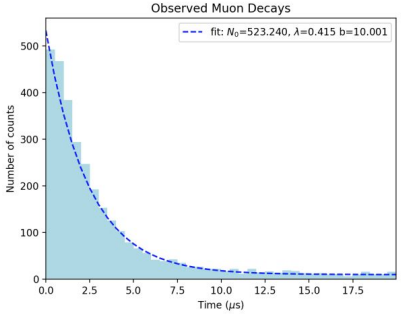


Figure 4. The result of fitting an exponential decay curve to the histogram shown in figure 3. $\chi^2 = 1.23$

The fit parameters are tabulated below with uncertainty.

Parameter	Value	Uncertainty
N_0	523.240	±1.2

Writing, cool physics, history! Ask questions if time permits

PHYS 2501W - Python Scipy & Matplotlib

```python
#stephen muon lab
#fall 2019

import numpy as np
import matplotlib.mlab as mlab
import matplotlib.pyplot as plt
import scipy, scipy.stats
from scipy.optimize import curve_fit

def exp_func(x,n,l,b):
    ''' x, n_0, lambda'''
    return n*np.exp(-l*x) +b

ofile = open('r.txt','r')
raws = []
for line in ofile:
    raws.append(float(line.rstrip('\n')))

num_bins = int(max(raws)+1)*2 #40 bins
n, bins, patches = plt.hist(raws, num_bins, facecolor='lightblue', alpha=1)

plt.xlabel(r'Time ($\mu$s)')
plt.ylabel('Number of counts')
plt.title(r'Observed Muon Decays')

plt.xlim(0, max(raws))

#get the x and y values for the hist
x_vals = np.linspace(0,20,41)
y_vals = []
for bar in patches:
    y_vals.append(bar.get_height())
y_vals.append(1)

popt, pcov = curve_fit(exp_func, x_vals, y_vals)
o_y_vals = [exp_func(x, *popt) for x in x_vals]
chi2 = scipy.stats.chisquare(y_vals, f_exp=o_y_vals)
print(chi2[0]/(len(x_vals-2)))
#print(pcov)
plt.plot(x_vals, exp_func(x_vals, *popt), 'b--',
         label=r'fit: $N_{0}$=%5.3f, $\lambda$=%5.3f, b=%5.3f' % tuple(popt))
```

Physics Research Context: Computer Simulations of Liquid Crystals

In media res: A multi-thousand line C++ simulation had caused two grad students to drop out already..

Professor recruited me and another undergrad Alice Hu to rewrite the simulation in Python. Alice handled math and stat mech side, I programmed.

The Metropolis Monte Carlo Method

Consider the following integral:

$$I = \int_a^b f(x)\,dx$$

$$I = \int_a^b \frac{f(x)}{\rho(x)}\rho(x)\,dx$$

$$I \approx \frac{1}{M}\sum_{i=1}^{M}\frac{f(x_i)}{\rho(x_i)}$$

Except… too costly!
Metropolis to rescue:

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. N. Teller, and E. Teller. Equation of state calculations by fast computing methods.
J. Chem. Phys., 21:1087-1092, 1953.

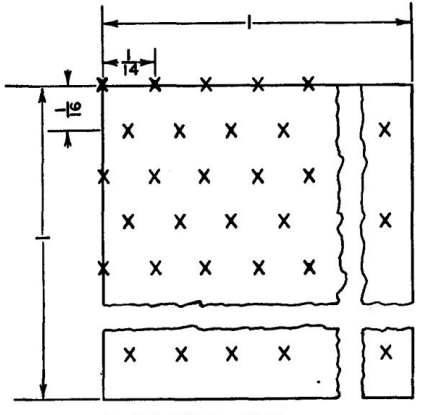


FIG. 2. Initial trigonal lattice.

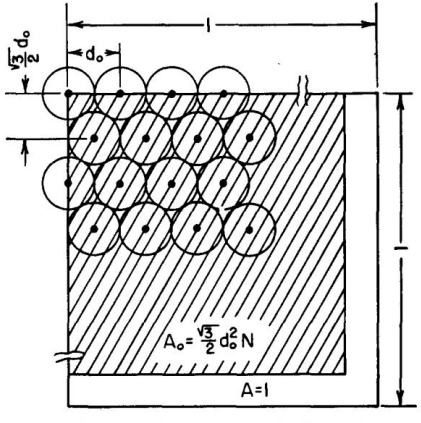


FIG. 3. The close-packed arrangement for determining A_0.

$r_{ij}=d_0$ and $\sum_j F_{ij}$ is given by Eq. (8), so we have

$$\langle \sum_i \mathbf{X}_i^{(\mathrm{int})}\cdot\mathbf{r}_i\rangle_{Av} = -(Nm\bar{v}^2/2)\pi d_0^2\bar{n}. \qquad (9)$$

Substitution of (9) into (7) and replacement of $(N/2)m\bar{v}^2$ by E_{kin} gives finally

$$PA = E_{\mathrm{kin}}(1+\pi d_0^2\bar{n}/2) \equiv NkT(1+\pi d_0^2\bar{n}/2). \qquad (10)$$

This equation shows that a determination of the one

The unit cell is a parallelogram with interior angle 60°, side d_0, and altitude $3^{\frac{1}{2}}d_0/2$ in the close-packed system.

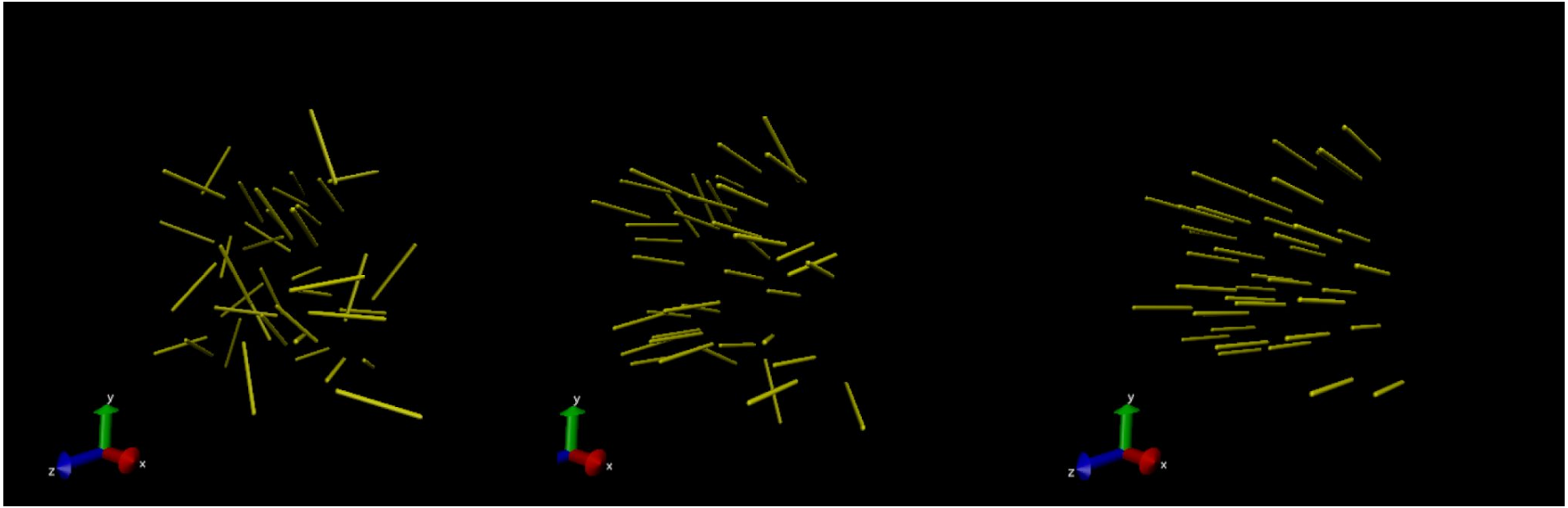
Every configuration reached by proceeding according to the method of the preceding section was analyzed in terms of a radial distribution function $N(r^2)$. We chose a $K>1$ for each ν and divided the area between πd_0^2 and $K^2\pi d_0^2$ into sixty-four zones of equal area ΔA^2,

$$\Delta A^2 = (K^2-1)\pi d_0^2/64.$$

The Metropolis Monte Carlo Method

Sphero-cylinders, torque when not aligned, hundreds of iterations.

What I would do differently today: Code in C, multithreading.



The Metropolis Monte Carlo Method

First object-oriented code I wrote.

Used numpy/scipy libraries for kd-tree

```python
# Stephen Duncanson & Alice Hu
# Fall 2019

import random
import math
import numpy as np
import constant as c
from numpy import linalg as LA
from scipy.spatial import cKDTree


class Simulation:
    def __init__(self, n_particles):
        self.n_particles = n_particles
        self.n_accepted_moves = 0
        self.n_rejected_moves = 0
        self.initial_system = generate_original_system(n_particles)
        self.initial_system_energy = calculate_energy(self.initial_system)
        self.initial_system_order_parameter = get_order_parameter(self.initial_system)
        self.current_system_energy = self.initial_system_energy    # originally
        self.current_order_parameter = self.initial_system_order_parameter  # originally
        self.system = self.initial_system   # originally
        self.trial_system = None    # originally
        self.trial_system_energy = None
        self.shifted_particle = None    # originally
        self.original_particle = None   # originally
        setup_vtf_output(n_particles*2)

    def get_initial_order_parameter(self):
        return self.initial_system_order_parameter

    def get_current_energy(self):
        return self.current_system_energy

    def get_acceptance_ratio(self):
        return self.n_accepted_moves / (self.n_rejected_moves + self.n_accepted_moves)

    def get_n_accepted_moves(self):
        return self.n_accepted_moves
```

The Metropolis Monte Carlo Method

Very poor data structure. Using mix of Python vanilla lists (issues) and np arrays (issues) Slow - but worked!

```
76  class Spherocylinder:
77      def __init__(self, p1, p2):
78          self.p1_position = p1
79          self.p2_position = p2
80          self.s = np.array([p2[0] - p1[0], p2[1] - p1[1], p2[2] - p1[2]])  # line segment
81          self.center_position = np.array([(p2[0] + p1[0]) / 2, (p2[1] + p1[1]) / 2, (p2[2] + p1[2]) / 2])
82          self.theta = get_theta(self.s)
83          self.phi = get_phi(self.s)
84          self.n = np.array([(p2[0] - p1[0]) / 2, (p2[1] - p1[1]) / 2, (p2[2] - p1[2]) / 2])
85          self.dipole_moment = 2 * c.Q * self.n
86          self.bond_exp = None
87          self.bond_angle = None
88          self.bonded = False
89
90      def get_bond_exp(self):
91          return self.bond_exp
92
93      def get_bond_angle(self):
94          return self.bond_angle
95
96      def set_bond_angle(self, bond_angle):
97          self.bond_angle = bond_angle
98
99      def set_bond_exp(self, bond_exp):
100         self.bond_exp = bond_exp
101
102     #def break_bond(self):
103         #self.bonded = False
104
105     def is_bonded(self):
106         return self.bonded
107
108     def bond(self):
109         self.bonded = True
110
111     def get_dipole_moment(self):
112         return self.dipole_moment
```

Hardest algorithm I wrote in 2019

Intersection for Metropolis et. al. was trivial (spheres)...

Would have been easier if I had taken linear algebra before writing..

```
199  def check_intersection(particle1, particle2):
200      small_num = 0.0000001  # number to check if they're closely parallel
201      u = particle1.get_s()  # s1
202      v = particle2.get_s()  # s2
203      p0 = particle1.get_p1_position()  # P0
204      q0 = particle2.get_p1_position()  # Q0
205      w = np.array([p0[0] - q0[0], p0[1] - q0[1], p0[2] - q0[2]])  # distance
206
207      a = np.dot(u, u)
208      b = np.dot(u, v)
209      f = np.dot(v, v)
210      d = np.dot(u, w)
211      e = np.dot(v, w)
212      D = (a * f) - (b * b)  # always >=0
213
214      # Set all to defaults
215      sc = D
216      sN = D
217      sD = D  # sc = sN / sD, default sD = D >= 0
218      tc = D
219      tN = D
220      tD = D  # tc = tN / tD, default tD = D >= 0
221
222      if D * D < small_num:  # checks if SCs are parallel
223          sN = 0.0  # force using point P0 on segment S1
224          sD = 1.0  # to prevent possible division by 0.0 later
225          tN = e
226          tD = f
227      else:  # get the closest points on the infinite lines
```

VMD
Visual Molecular Dynamics

Output to .vtf format for visualizing in vmd

```
517  def setup_vtf_output(num_of_atoms):
518      vtf_output_file = open('output.vtf', 'w')
519      for i in range(num_of_atoms):
520          if i % 2 == 0:
521              vtf_output_file.write('atom ' + str(i) + ' radius ' + str(c.RADIUS) + " name S1\n")
522          else:
523              vtf_output_file.write('atom ' + str(i) + ' radius ' + str(c.RADIUS) + " name S2\n")
524      for i in range(int(num_of_atoms/2)):
525          first_point = i+i
526          second_point = i+(i+1)
527          vtf_output_file.write('bond ' + str(first_point) + ':' + str(second_point) + '\n')
528      vtf_output_file.close()
529
530
531  def write_position(p):
532      vtf_output_file = open('output.vtf', 'a')
533      vtf_output_file.write('timestep\n')
534      vtf_output_file.write('PBC ' + str(c.X_BOUND) + ' ' + str(c.Y_BOUND) + ' ' + str(c.Z_BOUND) + '\n')
535      for o in p: #for every particle in the system
536          p1_pos = o.get_p1_position()
537          p2_pos = o.get_p2_position()
538          vtf_output_file.write(str(float(p1_pos[0])) + ' ' + str(float(p1_pos[1])) + ' ' + str(float(p1_pos[2])) + '\n')
539          vtf_output_file.write(str(float(p2_pos[0])) + ' ' + str(float(p2_pos[1])) + ' ' + str(float(p2_pos[2])) + '\n')
540      vtf_output_file.close()
```

Spring 2020

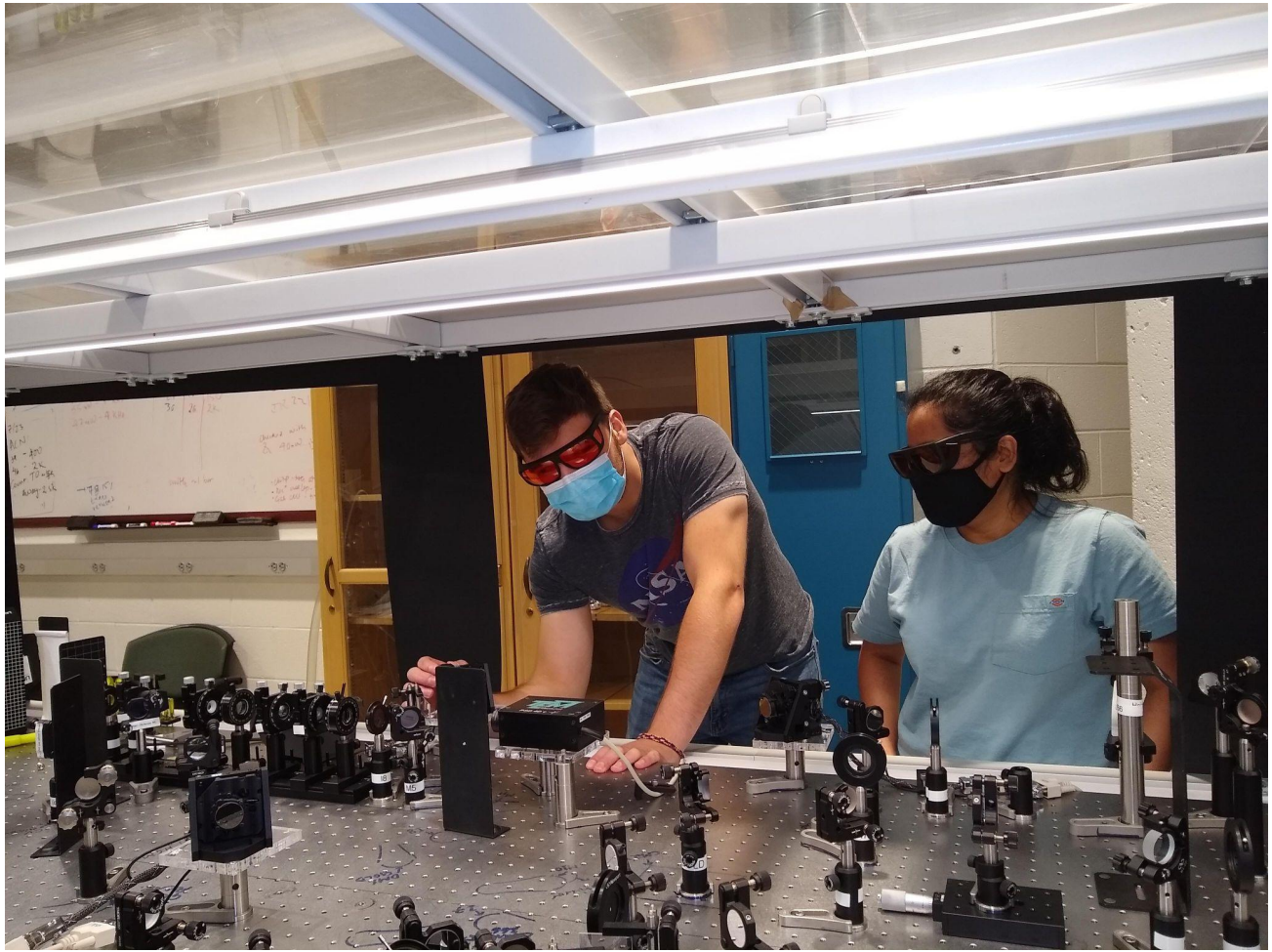
Physics; Optical physics research; Lotsa math; Covid sends the school home

Course		Description	Attempted Credits	Earned Credits	Grade	Grade Points
MATH	2110Q	Multivariable Calculus	4.00	4.00	A	16.000
MATH	2210Q	Applied Linear Algebra	3.00	3.00	A	12.000
MATH	2410Q	Elem Differential Equations	3.00	3.00	A	12.000
PHYS	2300	Development of Quantum Physics	3.00	3.00	A	12.000
PHYS	3989	Undergraduate Research	1.00	1.00	A	4.000

Summer 2020

I move back to UConn to continue Optical Physics Research

New Lab



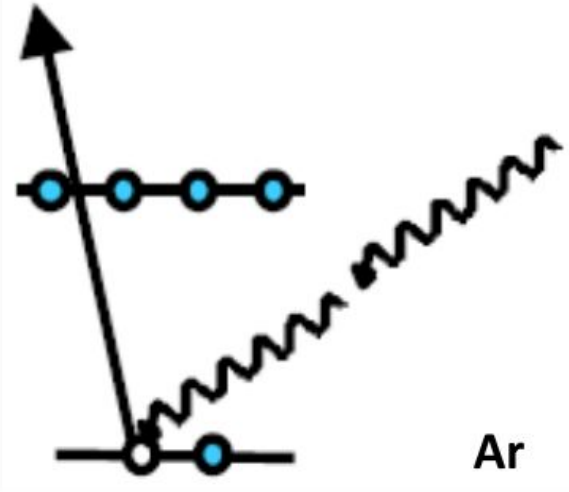
Goal: Produce Ar^{17+} by Simultaneous Absorption of 2 Photons--- Nonlinear Strong Field Response using

October 23, 2019

Stephen; Ionization potential; all energies below we get them from the NIST tables in their websites

IP of Ar^{16+}= 4087 eV.

Ar K-shell= 3205 eV.

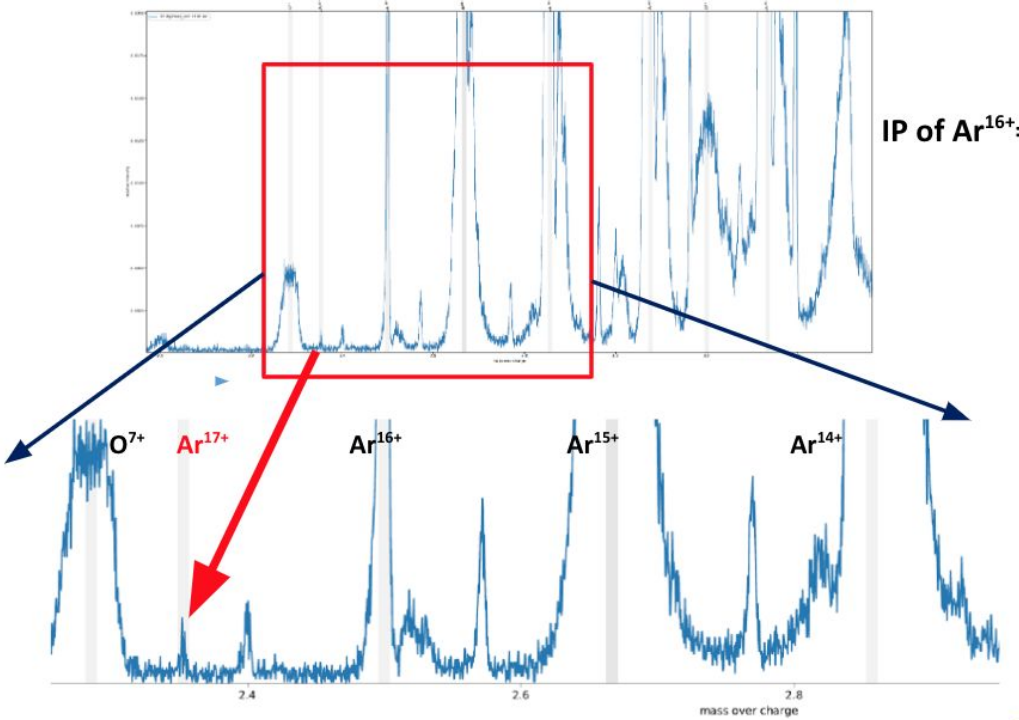


Ar

<u>Li-Like Ar</u>: 3100 eV

Set hγ@1550 eV; Half the energy

Ar @ hγ=1550 eV



We observe Ar^{17+} at photon energies:1520-1569 eV!!!

Resonances region: $1s2s^2$, $1s2s2p$, $1s2p^2$ (3040 eV-3140 eV)

2hγ =3100; 3hγ =4650 eV

Resonance-enhanced multiphoton ionization in the x-ray regime

Aaron C. LaForge,[1,*] Sang-Kil Son (손상길),[2,3,†] Debadarshini Mishra,[1] Markus Ilchen,[4,5] Stephen Duncanson,[1] Eemeli Eronen,[6] Edwin Kukk,[6] Stanislaw Wirok-Stoletow,[2,7] Daria Kolbasova,[2,7] Peter Walter,[8] Rebecca Boll,[4] Alberto De Fanis,[4] Michael Meyer,[4] Yevheniy Ovcharenko,[4] Daniel E. Rivas,[4] Philipp Schmidt,[4] Sergey Usenko,[4] Robin Santra,[2,3,7] and Nora Berrah[1]

[1]*Department of Physics, University of Connecticut, Storrs, Connecticut 06269, USA*
[2]*Center for Free-Electron Laser Science CFEL, Deutsches Elektronen-Synchrotron DESY, 22607 Hamburg, Germany*
[3]*The Hamburg Centre for Ultrafast Imaging, 22761 Hamburg, Germany*
[4]*European XFEL, 22869 Schenefeld, Germany*
[5]*Institut für Physik und CINSaT, Universität Kassel, 34132 Kassel, Germany*
[6]*Department of Physics and Astronomy, University of Turku, 20014 Turku, Finland*
[7]*Department of Physics, Universität Hamburg, 22607 Hamburg, Germany*
[8]*Linac Coherent Light Source, SLAC National Accelerator Laboratory, Menlo Park, California 94025, USA*

(Dated: November 9, 2021)

Here, we report on the nonlinear ionization of argon atoms in the short wavelength regime using ultraintense x rays from the European XFEL. After sequential multiphoton ionization, high charge states are obtained. For photon energies that are insufficient to directly ionize a $1s$ electron, a different mechanism is required to obtain ionization to Ar^{17+}. We propose this occurs through a two-color process where the second harmonic of the FEL pulse resonantly excites the system via a $1s \rightarrow 2p$ transition followed by ionization by the fundamental FEL pulse, which is a type of x-ray resonance-enhanced multiphoton ionization (REMPI). This resonant phenomenon occurs not only

Berrah Laser Lab: REMPI Research

Lots of people worked on the paper.

My role was generating graphs and usable data for analysis from the raw data.

What was the raw data? How was it stored?

Like earlier examples:

import h5py as h5

Huge HDF5 Files, 18 MHz ADC is just one of these instrument sources

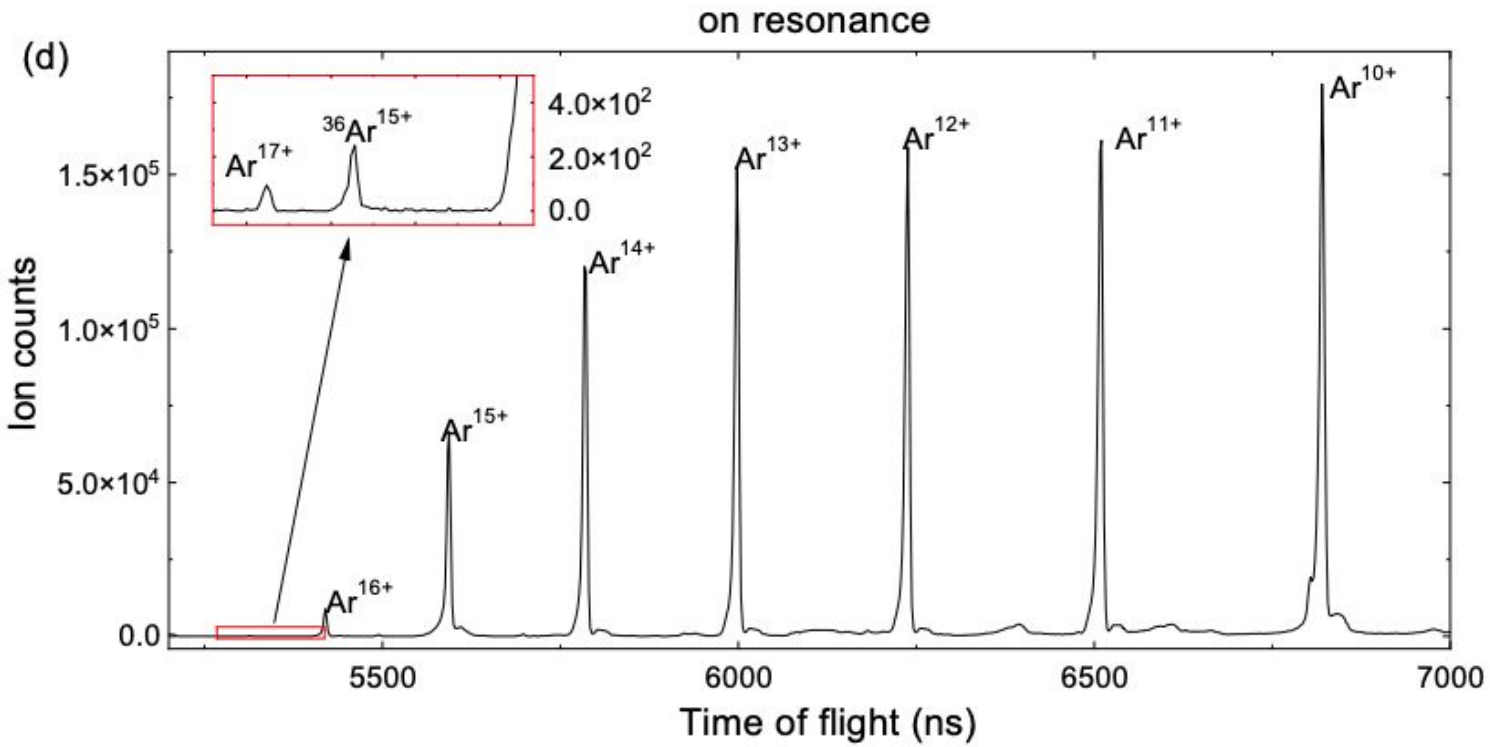
DA01	SA3_XTD10_XGM/XGM/DOOCS:output	'data.intensitySigmaTD', 'data.ySa3SigmaTD', 'data.intensityTD', 'data.xSa3TD', 'data.trainId', 'data.xTD', 'data.ySa1TD', 'data.ySa3TD', 'data.ySigmaTD', 'data.intensityAUXTD', 'data.xSa1TD', 'data.intensitySa1SigmaTD', 'data.ySa1SigmaTD', 'data.intensitySa1TD', 'data.intensityAUXSa3TD', 'data.intensitySa3TD', 'data.xSa1SigmaTD', 'data.yTD', 'data.intensitySa3SigmaTD', 'data.intensityAUXSa1TD', 'data.xSigmaTD', 'data.xSa3SigmaTD'
DA02	SQS_DIGITIZER_UTC1/ADC/1:network	'data.trainId', 'data.pos2', 'data.y4', 'data.pos0', 'data.pos3', 'data.pos1', 'data.y3', 'data.y2', 'data.y5', 'data.y0', 'data.y1'
DA04	SQS_DAQ_SCAN/MDL/KARABACON:output	'digitizers.channel_1_B.apd.pulseIntegral', 'digitizers.channel_1_A.apd.pulseIntegral', 'digitizers.channel_2_B.zero.samples', 'digitizers.channel_2_B.zero.lengths', ..., 'digitizers.channel_1_A.apd.triggerId', 'digitizers.channel_2_B.zero.pulseNum'

How do we detect ions? Time of Flight

Simple linear fit with atomic mass of most common argon isotope, if we know the timing of the ADC we can predict where ions will hit the ADC

```
260  def get_tof(charge, mass=40):
261      '''
262      The result of the linear calibration
263
264      accepts the charge: int in range [2,17]
265      as well as mass (defaults to Argon 40) the most common isotope
266      and returns tof in ns/2
267
268      this is used to count the number of ions in certain charged states
269      '''
270      slope = 6685.782541703501
271      intercept = 269.49516087660595
272      return slope*np.sqrt(mass/charge)+intercept
```

Time of Flight Spectrum



Cool way to get count with numpy array slice :

```
715  def get_count_and_error_per_pulse_at_intensity(peaks_at_int):
716      '''
717      takes the dictionary of peaks at intensity and returns
718      a dictionary of the count mode at each intensity
719      '''
720      pulse_time = get_pulse_time()
721      bin_size = 1 # 5 ns, 1 = ns/2
722      # want to use a bigger bin to make the shape a little more square
723      #print(pulse_time+1)
724      # are the zeros the same shape?
725      count_at_intensity = {}
726      for intensity_bin in peaks_at_int:
727          # check if there are 0 pulses
728          if peaks_at_int[intensity_bin][0] == []:
729              # there are no peaks at this intensity (even if there are pulses!)
730              count_at_intensity[intensity_bin] = np.zeros(pulse_time)
731          else:
732              # there are peaks at this intensity
733              all_peaks = peaks_at_int[intensity_bin][0]
734              count = np.histogram(all_peaks, bins=range(0, pulse_time+1, bin_size))[0]
735              # divide by number of pulses
736              ar17_count = np.sum(count[int(get_tof(17)-12):int(get_tof(17)+12)])
737              ar16_count = np.sum(count[int(get_tof(16)-12):int(get_tof(16)+12)])
738              ar15_count = np.sum(count[int(get_tof(15)-12):int(get_tof(15)+12)])
739              ar14_count = np.sum(count[int(get_tof(14)-12):int(get_tof(14)+12)])
740              ar13_count = np.sum(count[int(get_tof(13)-12):int(get_tof(13)+12)])
741              ar12_count = np.sum(count[int(get_tof(12)-12):int(get_tof(12)+12)])
742
743              count = count/peaks_at_int[intensity_bin][1]
744              count_at_intensity[intensity_bin] = count
745      return count_at_intensity, peaks_at_int[intensity_bin][1]
```

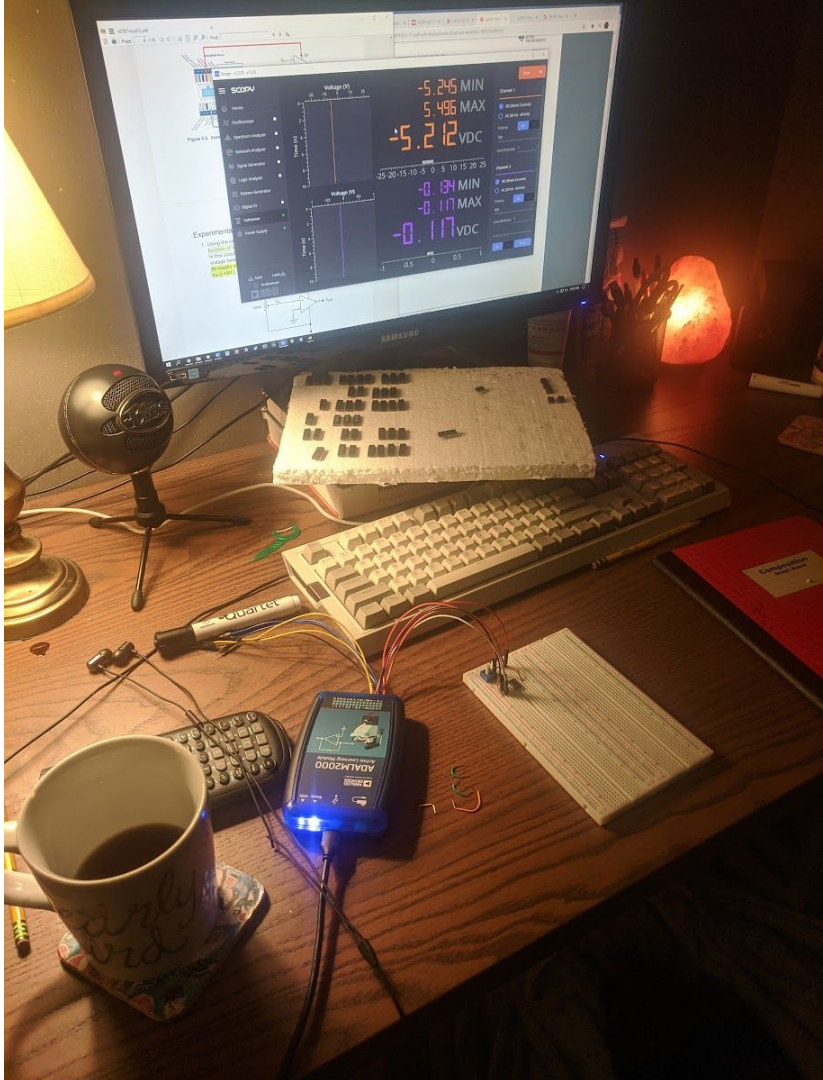
Fall 2020

1st online semester;
Switching to Computer Engineering

Course		Description	Attempted Credits	Earned Credits	Grade	Grade Points
CSE	1729	Intro to Principle Programming	3.00	3.00	A	12.000
CSE	2301	Prin of Digital Logic Design	4.00	4.00	A-	14.800
ECE	2001	Electrical Circuits	4.00	4.00	A	16.000
HIST	2222E	Global Environmental History	3.00	3.00	A	12.000
STAT	3345Q	Probability Models Engineers	3.00	3.00	A	12.000

CSE 2301: Digital Logic
ECE 2001: Circuits
Desk looked like:

I'll go into one project for each of these hardware classes. Ask me about other projects if time, or ask me to skip if no time :)



CSE 2301 Lab 6

detecting the sequences ‘0001’ and ‘1110’ using 74 series logic chips.

1) Create states and assign to flip flops. (The sort of thing VHDL will take care of for you)

State	Detects
C	0
D	00
E	000
F	1
G	11
H	111

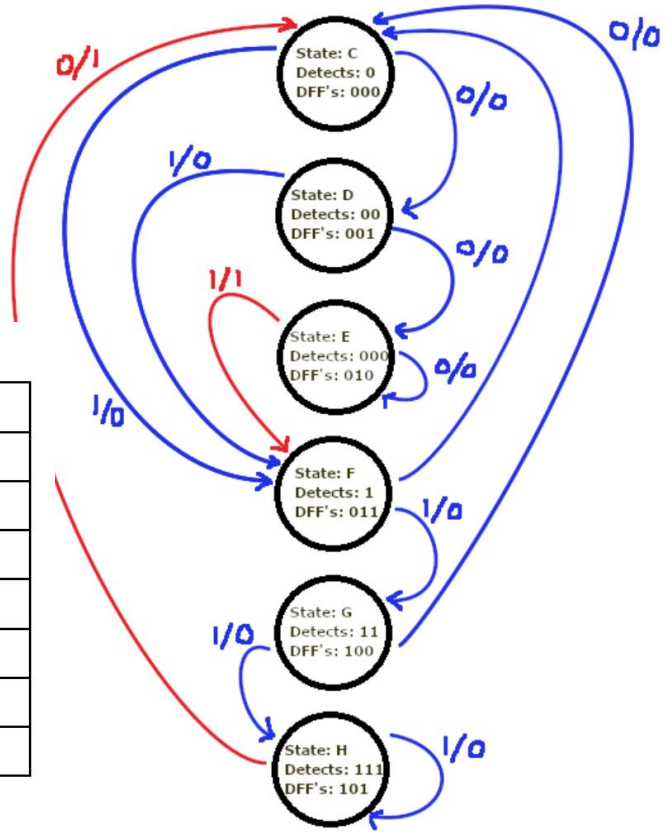
The information as to which state we are in can be stored in a set of 3 D flip flops. If we define state C = 000, D = 001, etc., we can rewrite this table using the contents of the 3 D flip flops. This is shown below.

State | $Q_c Q_b Q_a$	Detects
C | 000	0
D | 001	00
E | 010	000
F | 011	1
G | 100	11
H | 101	111

detecting the sequences ‘0001’ and ‘1110’ using 74 series logic chips.

CSE 2301 Lab 6

2) State diagram -> state table



This state diagram can be converted into a state table (here A is the input).

$Q_c(t)Q_b(t)Q_a(t)$	$Q_c(t+)Q_b(t+)Q_a(t+) / B$	
	A=0	**A=1**
000	001/0	011/0
001	010/0	011/0
010	010/0	011/1
011	000/0	100/0
100	000/0	101/0
101	000/1	101/0

CSE 2301 Lab 6

detecting the sequences ‘0001’ and ‘1110’ using 74 series logic chips.

3) K-Maps from state table

2/4 shown, point is illustrated however. We can now directly implement this.

The k-map for the Q_b flip flop:

$AQ_c \setminus Q_bQ_a$	00	01	11	10
00	0	1	0	1
01	0	0	X	X
11	0	0	X	X
10	1	1	0	1

Which can be expressed as:

$$Q_b(t+) = Q_bQ_a' + AQ_c'Q_b' + Q_c'Q_b'Q_a$$

The final D flip flop Q_a:

$AQ_c \setminus Q_bQ_a$	00	01	11	10
00	1	0	0	0
01	0	0	X	X
11	1	1	X	X
10	1	1	0	1

Which is written:

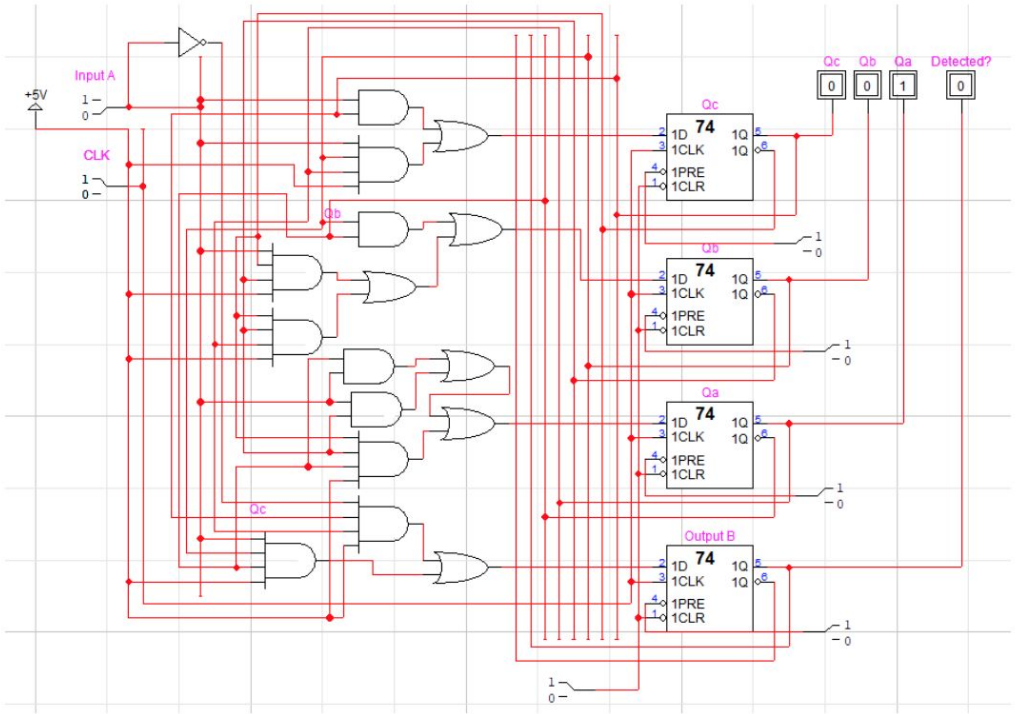
$$Q_a(t+) = Q_c'Q_b'Q_a' + AQ_b' + AQ_a'$$

CSE 2301 Lab 6

detecting the sequences ‘0001’ and ‘1110’ using 74 series logic chips.

4) Simulate logic (using logic works 5)

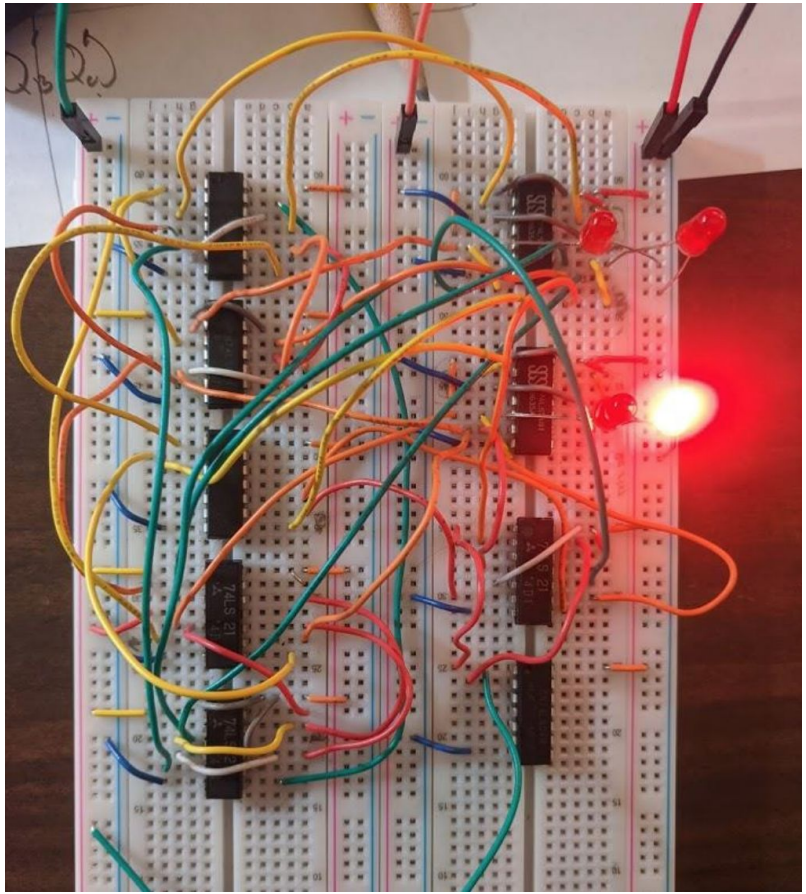
Below is an image of the sequence detector, built in LogicWorks.



CSE 2301 Lab 6

5) Construct on breadboard from schematic

detecting the sequences ‘0001’ and ‘1110’ using 74 series logic chips.

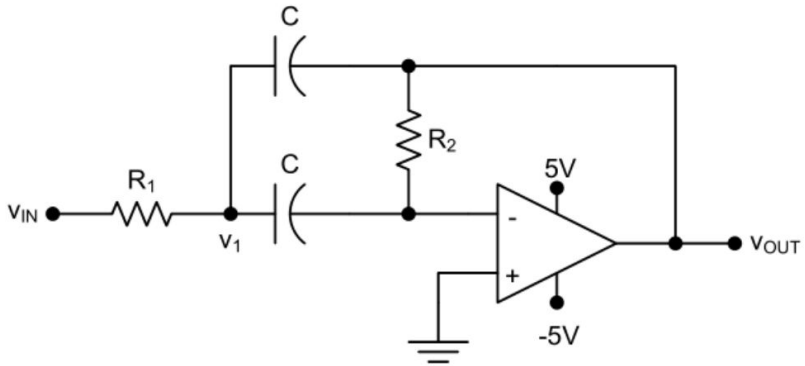


ECE 2001: Audio Equalizer

We begin by reviewing required specifications for the three-band audio equalizer.

Parameter	**Value**
Supply Voltages	±5V
Bass Band	Center frequency = 180 Hz ± 10% Quality factor = 0.7 ± 10% Range of adjustable gain = [0.4, 8]
Midrange Band	Center frequency = 1 kHz ± 10% Quality factor = 0.7 ± 10% Range of adjustable gain = [0.2, 4]
Treble Band	Center frequency = 5.5 kHz ± 10% Quality factor = 0.7 ± 10% Range of adjustable gain = [0.1, 2]

ECE 2001: Audio Equalizer



The center frequency for this bandpass filter is given by:

$$f_o = \frac{1}{2\pi C\sqrt{R_1 R_2}}$$

The resonant gain is given by:

$$A_r = -\frac{R_2}{2R_1}$$

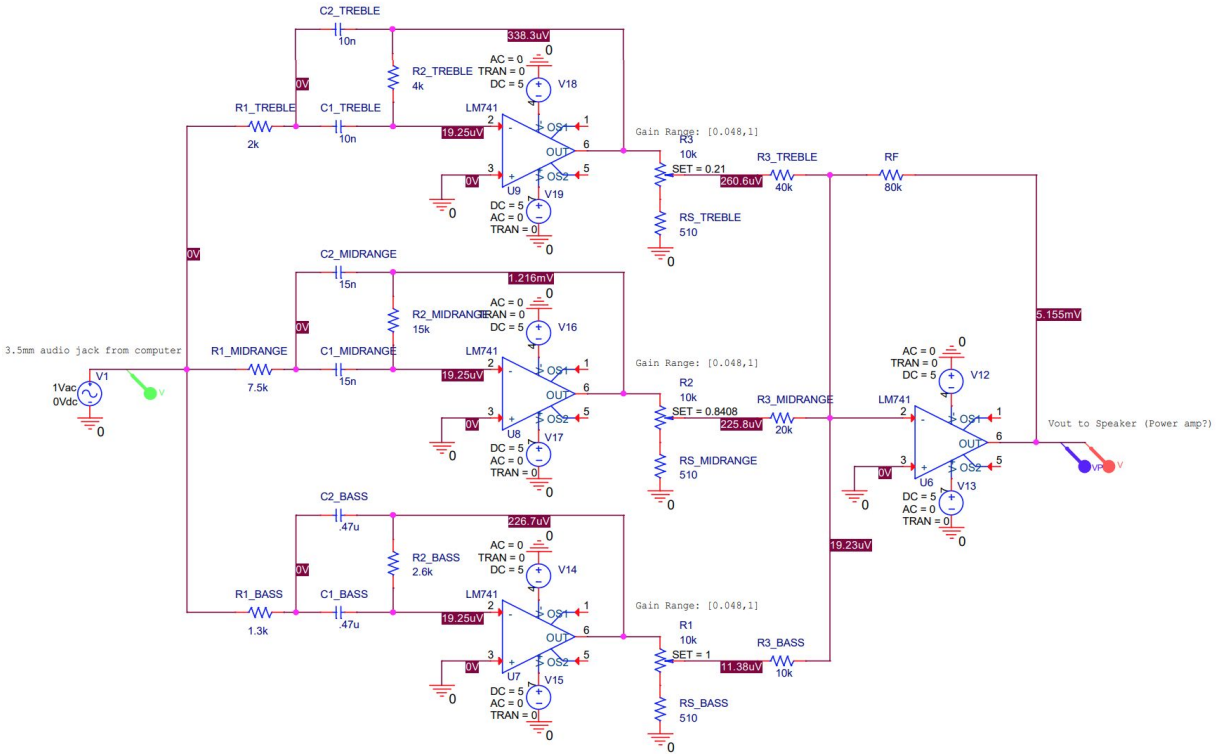
And the Q-Factor by:

$$Q = \frac{1}{2}\sqrt{\frac{R_2}{R_1}}$$

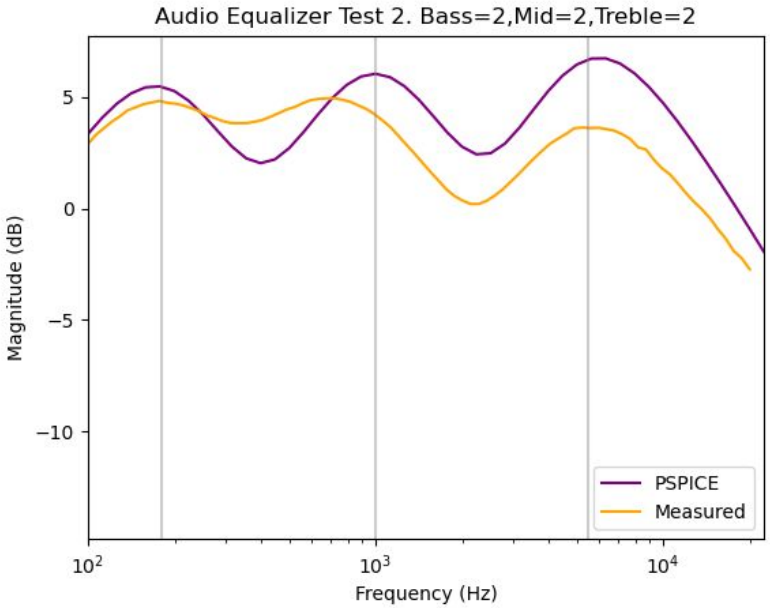
ECE 2001: Audio Equalizer

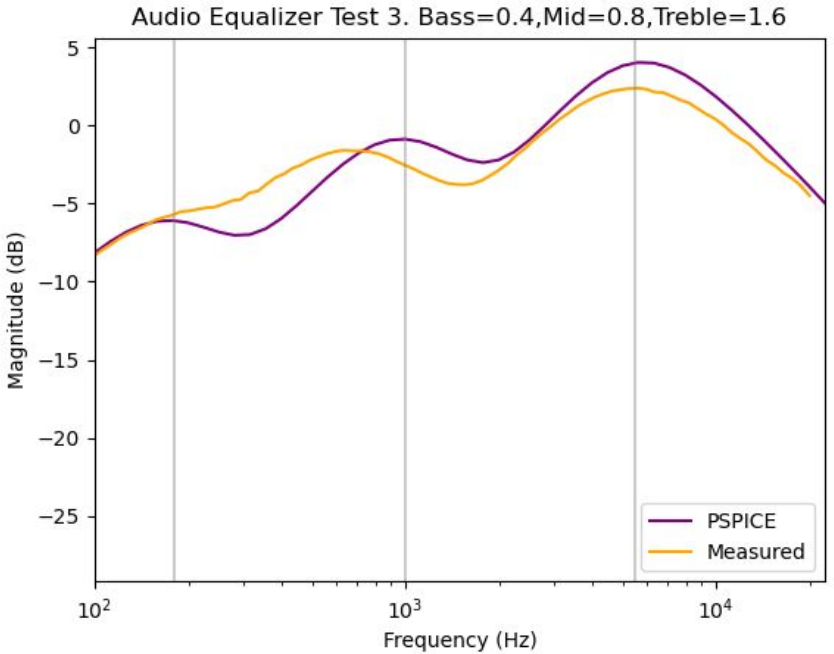
```
1   # stephen duncanson
2   # used to find values of C and R1, R2 for DP2
3   import numpy as np
4   resistors =
5   [1000,1100,1300,1600,2000,2200,3300,5100,7500,10000,15e3,20e3,33e3,51e3,75e3,
6   100e3,200e3]
7
8   capacitors = [1e-9, 1.5e-9, 2.2e-9, 3.3e-9, 4.7e-9, 10e-9, 15e-9, 22e-9, 33e-9,
9                 47e-9, .1e-6, .15e-6, .22e-6, .33e-6, .47e-6]
10
11  # 180, 1000, 5500 (Hz)
12  goals = [180, 1000, 5500]
13  # we may need to combine resistors to get the values we need.
14  for goal in goals:
15  print("current goal: "+str(goal))
16  current_distance = 300
17  for r1 in resistors:
18  for c in capacitors:
19  r2 = r1*2
```

ECE 2001: Audio Equalizer, OrCAD Model



Testing the Audio Equalizer





Spring 2021

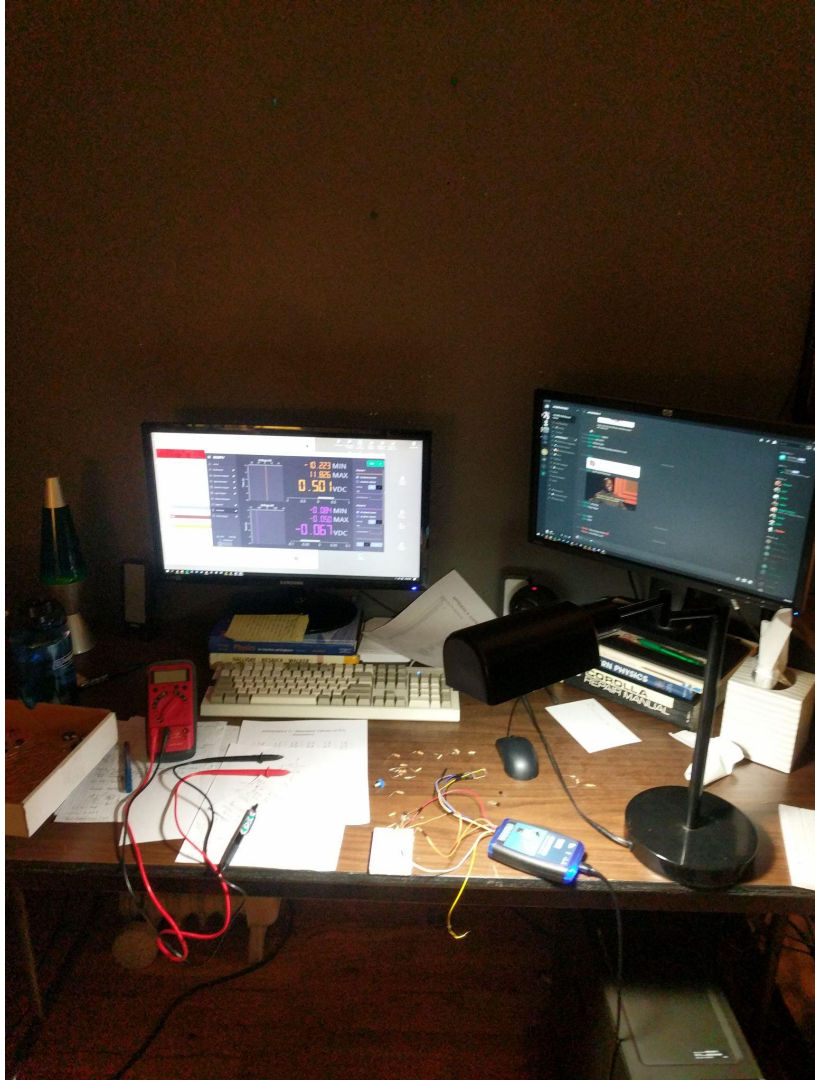
2nd online semester; Computer Engineering

CSE	2050	Data Struc and Obj-Orient Des	3.00	3.00	A-	11.100
CSE	2500	Introduction to Discrete Syste	3.00	3.00	B	9.000
CSE	3302	Digital Systems Design	3.00	3.00	A	12.000
ECE	3101	Signals and Systems	3.00	3.00	A	12.000
ECE	3201	Elec Circ Design & Analysis	4.00	4.00	B-	10.800
ECE	4900W	Comm Engr Solutions Soc Cntxt	1.00	1.00	A	4.000

Spring 2021: An overview

I'll devote one slide to each course, ask questions if you would like to know more.

Bonus: I've got a second monitor since last semester, and the desk has got messier.



Spring 2021:
Data Structures & OOP

Binary search tree pictured.
All sorts of, you guessed it, data structures and algorithms in Python.

This is my first Python class.

```python
from BSTNode import BSTNode

# Public interface: users only interact with the class BSTMap.
# Methods in BSTMap often call BSTNode methods, which do the heavy lifting.
class BSTSet:
    def __init__(self):
        self._head = None
        self._len = 0

    def put(self, key):
        if self._head is None:
            self._head = BSTNode(key)
            self._len += 1

        else:
            self._head.put(key)
            self._len += 1

    def __len__(self): return self._len

    def __iter__(self): return iter(self._head)

    def in_order(self): yield from self._head.in_order()

    def pre_order(self): yield from self._head.pre_order()

    def post_order(self): yield from self._head.post_order()

if __name__ == '__main__':
    bst = BSTSet()
    n = 4

    # Create a balanced tree with 7 nodes
    #       3
    #    /     \
    #   1       5
    #  / \     / \
    # 0   2   4   6
    for i in [3, 1, 0, 2, 5, 4, 6]:
        bst.put(i)
```

Spring 2021:
Discrete Systems

Applied math student does proofs class.

Let A(n) = "n is prime"

B(n) = "n is odd"

C(n) = "n = 2"

$\forall n \in \boldsymbol{Z}, if\ A(n)\ then\ B(n)\ or\ C(n)$

The negation of a universal statement is in the form of an existential statement:

$\exists n \in \boldsymbol{Z}\ if\ A(n)\ then \neg(B(n)\ or\ C(n))$

By DeMorgan's law:

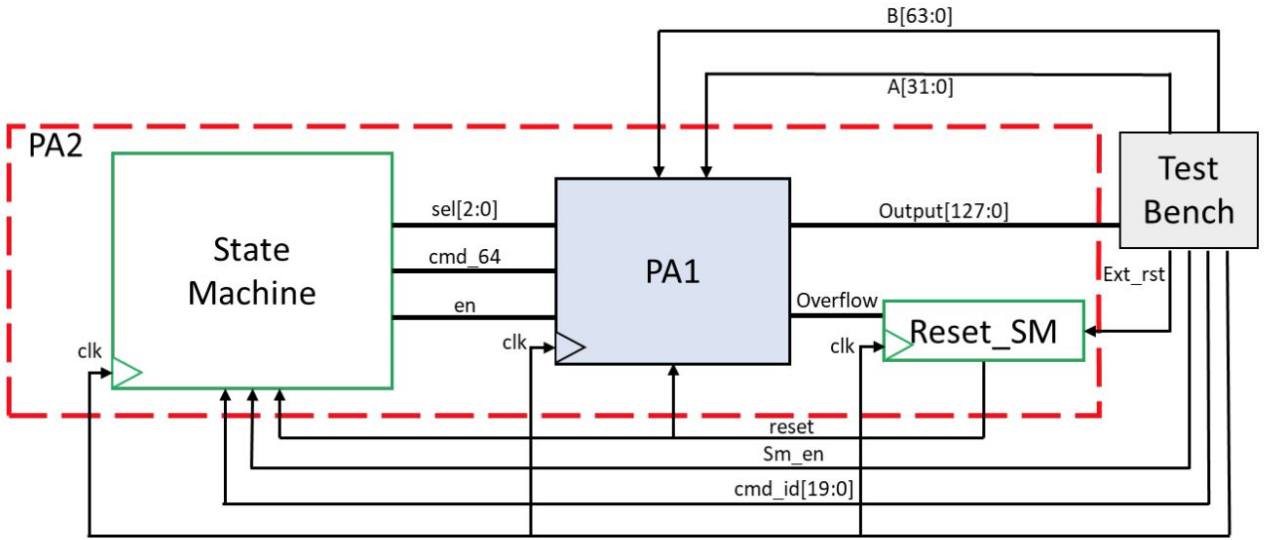
$\exists n \in \boldsymbol{Z}\ if\ A(n)\ then \neg B(n)\ and \neg C(n)$

Therefore: $\exists n \in \boldsymbol{Z}\ such\ that\ n\ is\ prime\ and\ n\ is\ not\ odd\ and\ n \neq 2.$

Spring 2021:
Digital Systems Design, PA2

VHDL Class!

This programming assignment introduces you to finite state machine design. The programming assignment will take the “ALU” module from the previous programming assignment (pa1) and inject a sequence of commands into the design. To complete this programming assignment, you will need to design *two state machine* modules, one for PA1 ALU and other for system reset. You will also need to develop a test bench to test your design.



Spring 2021:
Digital Systems Design, PA2

VHDL Class!

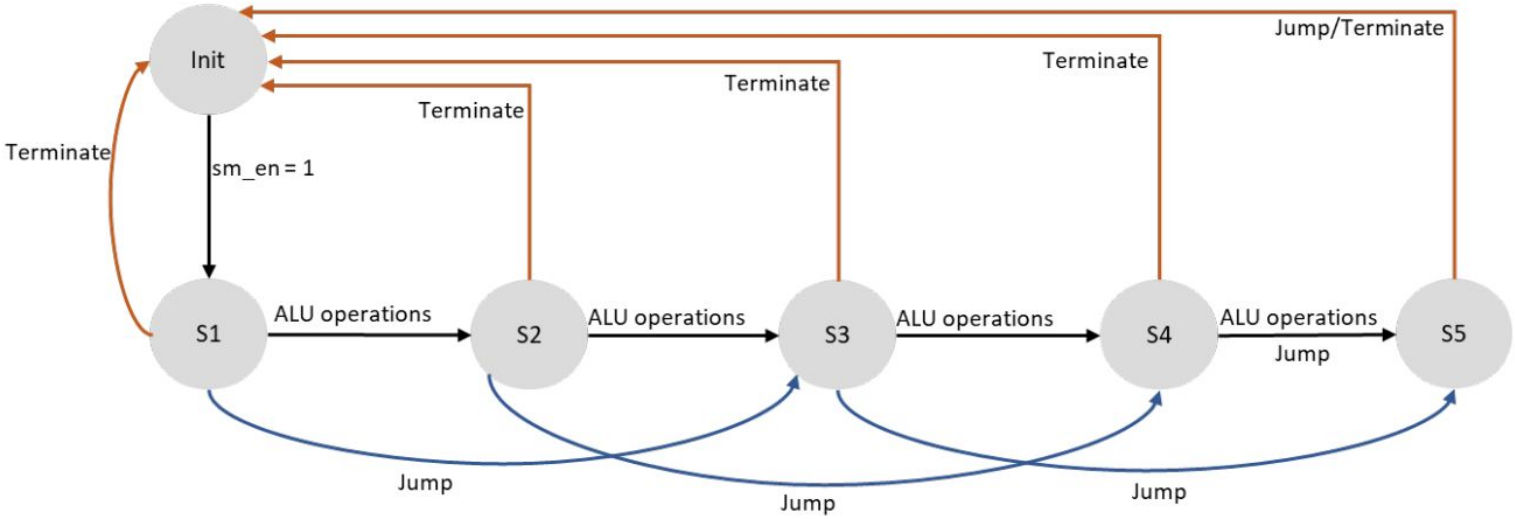


Figure 2. ALU State Sequencing Flow Chart

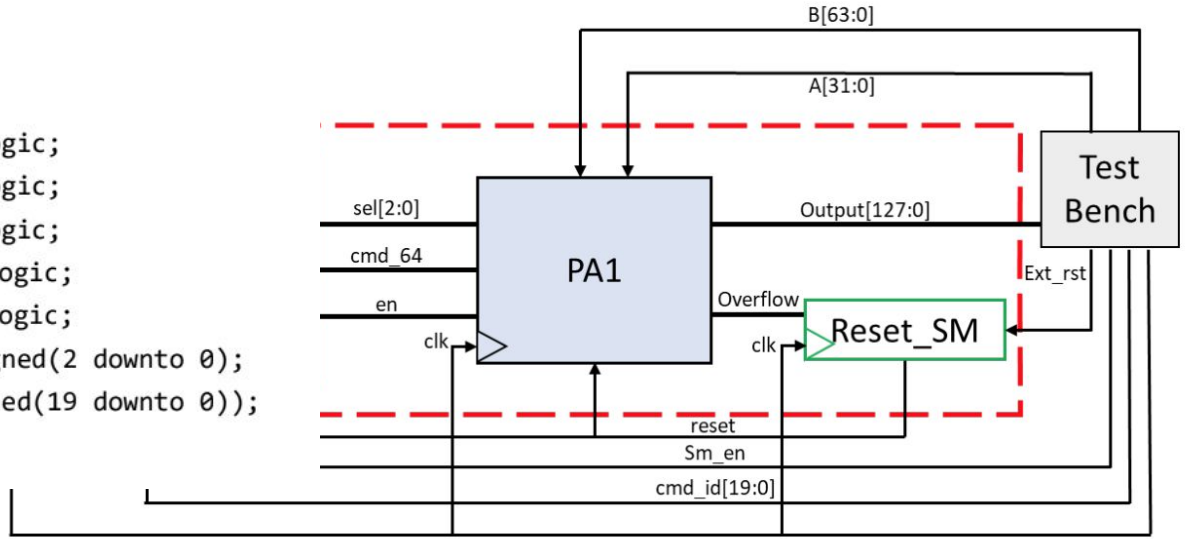
Spring 2021:
Digital Systems Design, PA2

VHDL Class!

State_machine.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity state_machine is
port(
        clk     : in std_logic;
        reset   : in std_logic;
        sm_en   : in std_logic;
        en      : out std_logic;
        cmd_64  : out std_logic;
        sel     : out unsigned(2 downto 0);
        cmd_id  : in unsigned(19 downto 0));
end state_machine;
```

oduces you to finite state machine design. The programming assignment will revious programming assignment (pa1) and inject a sequence of commands s programming assignment, you will need to design *two state machine* er for system reset. You will also need to develop a test bench to test your



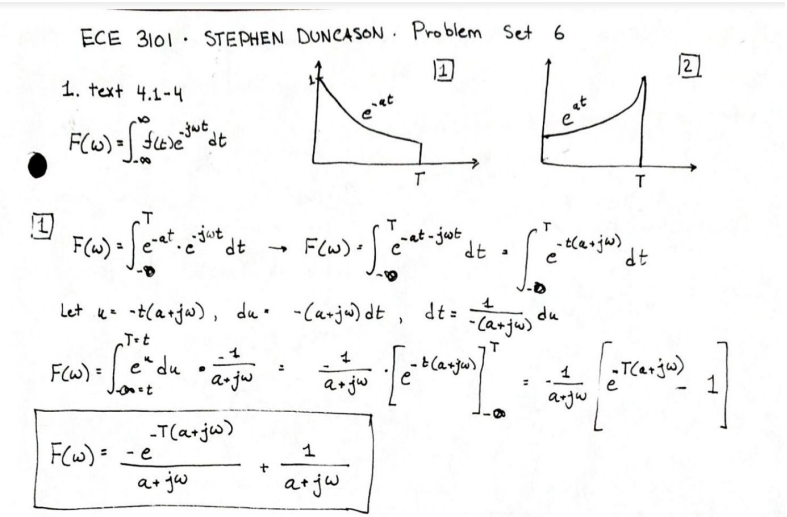
Spring 2021: Digital Systems Design, PA2

```
architecture Behavioral of state_machine is
type state_type is (START, ONE, TWO, THREE, FOUR, FIVE);
signal state : state_type;
signal cmd_id_r: unsigned(19 downto 0) := (others => '0');
begin

process(clk, reset) begin
    -- async reset
    if reset = '1' then state <= START; en <= '0';
    -- otherwise on clock event
    elsif clk'event and clk='1' then
    case state is
        when START =>
            if sm_en = '1' then cmd_id_r <= cmd_id; state <= ONE;
            else state <= START;
            end if;
        when ONE =>
            if cmd_id_r(19 downto 16) < "1000" then
                en <= '1'; cmd_64 <= cmd_id_r(18); sel <= cmd_id_r(18 downto 16); state <= TWO;
            elsif cmd_id_r(19 downto 16) = "1000" then en <= '0'; state <= THREE;
            else en <= '0'; state <= START;
            end if;
        when TWO =>
            if cmd_id_r(15 downto 12) < "1000" then
                en <= '1'; cmd_64 <= cmd_id_r(14); sel <= cmd_id_r(14 downto 12); state <= THREE;
            elsif cmd_id_r(15 downto 12) = "1000" then en <= '0'; state <= FOUR;
            else en <= '0'; state <= START;
            end if;
        when THREE =>
            if cmd_id_r(11 downto 8) < "1000" then
                en <= '1'; cmd_64 <= cmd_id_r(10); sel <= cmd_id_r(10 downto 8); state <= FOUR;
            elsif cmd_id_r(11 downto 8) = "1000" then en <= '0'; state <= FIVE;
```

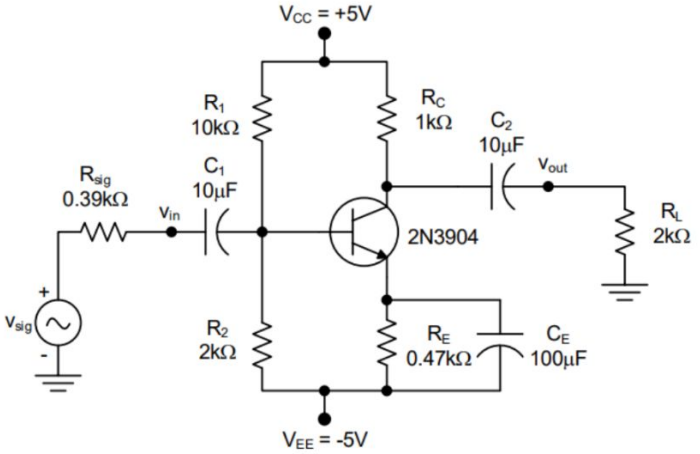
Spring 2021: Signals and Systems

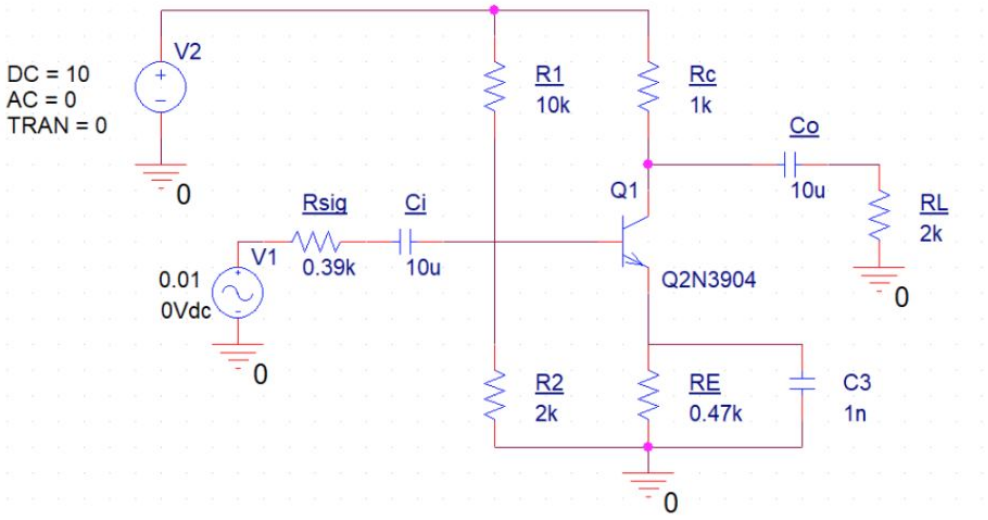
Fourier Analysis Class. Matlab used, though I am not great at it.

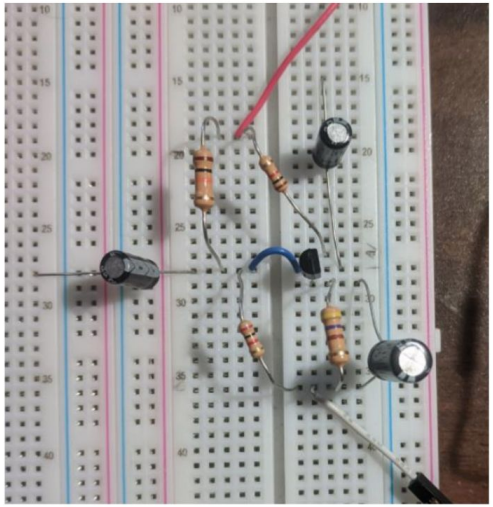
What I remember most is doing homework like-> and scanning it in



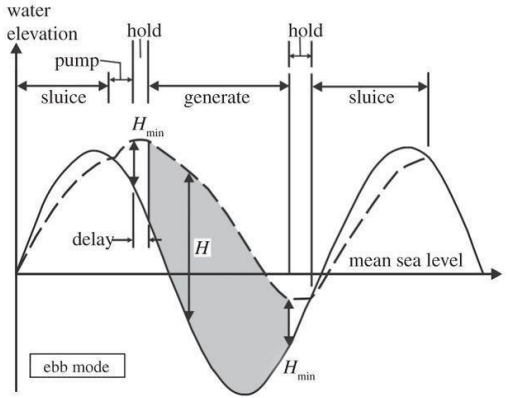
Spring 2021:
Circuit Design & Analysis
BJT CE

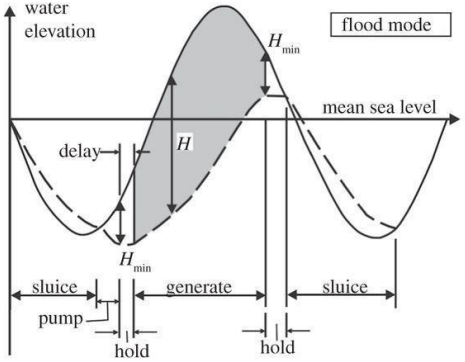


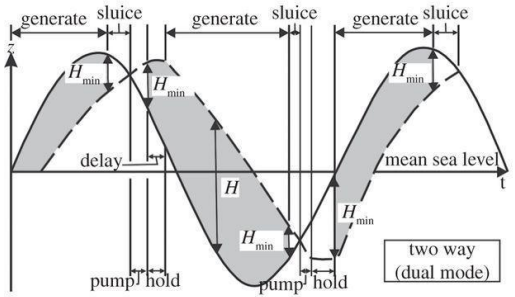


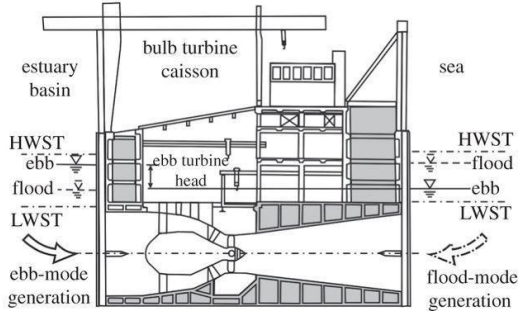


Spring 2021:
Engineering Writing









Fall 2021

Back on Campus; Computer Engineering; Lots of classes

CSE	3100	Systems Programming	3.00	3.00	A	12.000
CSE	3666	Intro to Computer Architecture	3.00	3.00	B+	9.900
CSE	4302	Computer Organization&Architec	3.00	3.00	A	12.000
ECE	3221	Digital Integrated Circuits	3.00	3.00	B	9.000
ECE	4112	Digital Communicat & Networks	3.00	3.00	B+	9.900
ECE	4401	Digital Design Laboratory	3.00	3.00	B+	9.900
ECE	4901	ECE Design I	2.00	2.00	A	8.000
ENGR	3201	Undergrad Teach, Mentor, Lead	3.00	3.00	A	12.000

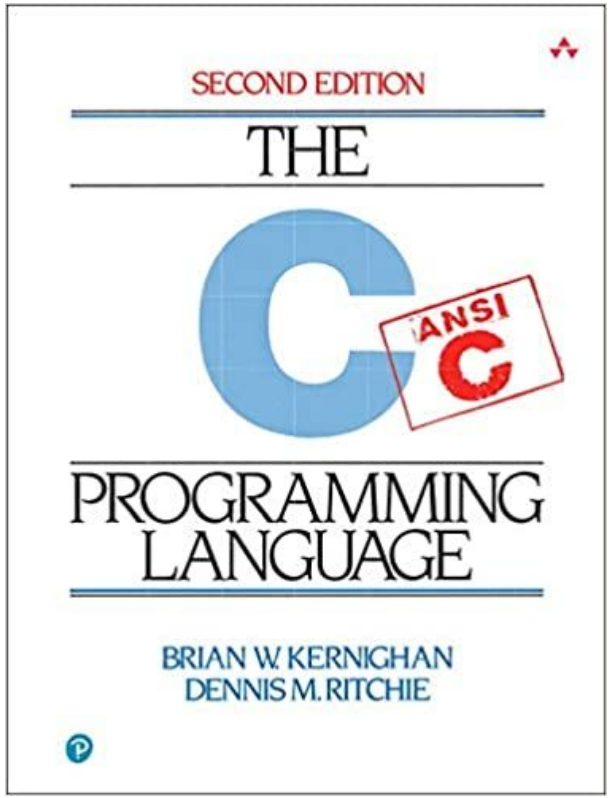
Systems Programming, C!

C is my favorite programming language.

3 part course covered

1) Language
2) Sockets, stdlib
3) Pthreads

Used gnu toolchain (gcc, make, emacs)



Systems Programming, C!

Structured programming.

Implementing data structures and creating an interface to the data structure with functions.

```c
#include <stdio.h>
#include <stdlib.h>

typedef struct Heap {
  int msz; /* the max size of the heap */
  int* t;  /* t points to an array of integers */
  int sz;  /* how many we have */
} Heap;

Heap* makeHeap(int msz);
void freeHeap(Heap* h);
void insertHeap(Heap* h, int v);
int emptyHeap(Heap* h);
int extractMin(Heap* h);
void printHeap(Heap* h);

int main()
{
  Heap* h = makeHeap(128);
  for(int i=0; i < 10; i++)
    {
      insertHeap(h,10 - i);
      printHeap(h);
    }
  while(!emptyHeap(h))
    {
      int v = extractMin(h);
      printf("Got %d\n",v);
    }
  freeHeap(h);
  return 0;
}
```

Structured Programming in C

Create a hashmap interface built atop out dllist structure, but without worrying or needing to care about how dll is implemented- abstraction.

```c
#include "hashMap.h"
#include "dllist.h"
#include <string.h>

unsigned long long hash(char* key)
{
  const int p = 31;
  const int m = 1e9 + 9;
  unsigned long long hash_value = 0;
  long long p_pow = 1;
  for (int i = 0; i < strlen(key); i++) {
    char c = key[i];
    hash_value = (hash_value + (c - 'a' + 1) * p_pow) % m;
    p_pow = (p_pow * p) % m;
  }
  return hash_value;
}

void initHashMap(HashMap* hm, unsigned int nb)
{
  hm->numBuckets = nb;
  hm->numItems = 0;
  hm->buckets = malloc(sizeof(Bucket)*nb);
  for (int i=0; i<nb; i++) {
    hm->buckets[i].list = malloc(sizeof(DLList));
    initList(hm->buckets[i].list);
    pthread_mutex_init(&hm->buckets[i].lock);
  }
  pthread_mutex_init(&hm->tableLock);
}

void clearHashMap(HashMap* hm)
{
  for (int i=0; i<hm->numBuckets; i++) {
    free(hm->buckets[i].list);
    pthread_mutex_destroy(&hm->buckets[i].lock);
  }
  free(hm->buckets);
  pthread_mutex_destroy(&hm->tableLock);
  hm->numItems = 0;
```

```c
int main(int argc,char* argv[])
{
  int n = atoi(argv[1]); // number of philosophers
  int c = atoi(argv[2]); //number of cycles to be completed by each philosopher

  Table* t = makeTable(n); // shared table
  pthread_t threads[n]; // array of thread ids for each philosopher
  Philosopher thinkers[n]; // array of philosopher structs

  for(int i=0; i<n; i++) { // set up each philo
    thinkers[i].pid=i;
    thinkers[i].ate=0;
    thinkers[i].cycle=c;
    thinkers[i].state=EAT;
    thinkers[i].t=t;
    pthread_create(&threads[i], NULL, (void*)muse, &thinkers[i]);
  }

  for(int i = 0; i < n; i++) {
    pthread_join(threads[i], NULL);
    printf("%d ate %d times\n",i,thinkers[i].ate);
  }

  clearTable(t);
  return 0;
}
```

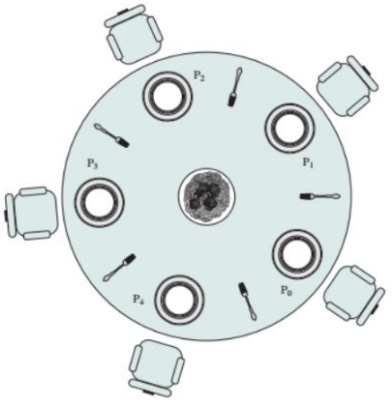


Figure 1: Dining Philosophers' problem with 5 philosophers

3 (40 points) Dining philosophers' problem

```c
typedef struct Table {
  pthread_cond_t* done_eating;
  int* forks;
  pthread_mutex_t table_lock;
  int n;
} Table;

/* This is the ADT to store information for each philosopher */
typedef struct PhiloTag {
  int pid; //ID
  int state; //activity state
  int ate; //number of times eaten
  int cycle; //number of cycles left
  Table* t; //shared Table ADT
} Philosopher;

Table* makeTable(int n)
{
  Table* t = (Table*)malloc(sizeof(Table));
  pthread_mutex_init(&t->table_lock,NULL);
  t->n = n; // n seats at table
  t->forks = malloc(sizeof(int)*n);
  t->done_eating = malloc(sizeof(pthread_cond_t)*n);
  for (int f=0; f<n; f++) {
    t->forks[f]=1; // all forks are available at the start
    pthread_cond_init(&t->done_eating[f],NULL);
  }
  return t;
}
```

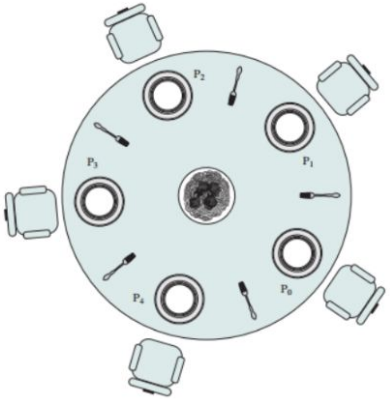


Figure 1: Dining Philosophers' problem with 5 philosophers

3 (40 points) Dining philosophers' problem

```
75  void* muse(Philosopher* p)
76  {
77    unsigned seed = (unsigned)pthread_self(); //seed used to put each
78    int cycle = p->cycle;
79    int pid = p->pid;
80    int n = p->t->n;
81
82    while (cycle) {
83      if (p->state==THINK)
84        doActivity(HUNGRY,p,&seed);
85      else if (p->state==HUNGRY) {
86        pthread_mutex_lock(&p->t->table_lock);
87        while (p->t->forks[pid]==0 && p->t->forks[(pid+1)%n]==0)
88    pthread_cond_wait(&p->t->done_eating[p->pid], &p->t->table_lock);
89        p->t->forks[p->pid] = 0;
90        p->t->forks[(p->pid+1)%(p->t->n)] = 0;
91        pthread_mutex_unlock(&p->t->table_lock);
92        doActivity(EAT,p,&seed);
93        p->t->forks[p->pid] = 1;
94        p->t->forks[(p->pid+1)%(p->t->n)] = 1;
95        pthread_cond_signal(&p->t->done_eating[p->pid]); // signal we are done with lower fork
96        pthread_cond_signal(&p->t->done_eating[(p->pid+1)%(p->t->n)]);
97        cycle--;
98      }
99      else
100       doActivity(THINK,p,&seed);
101   }
102   return NULL;
103 }
```

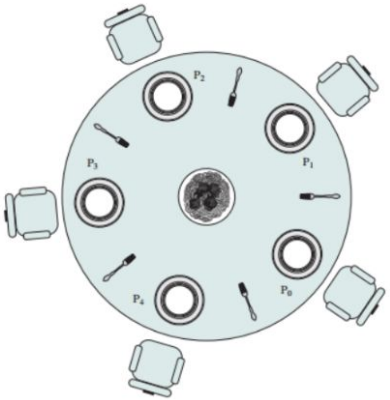
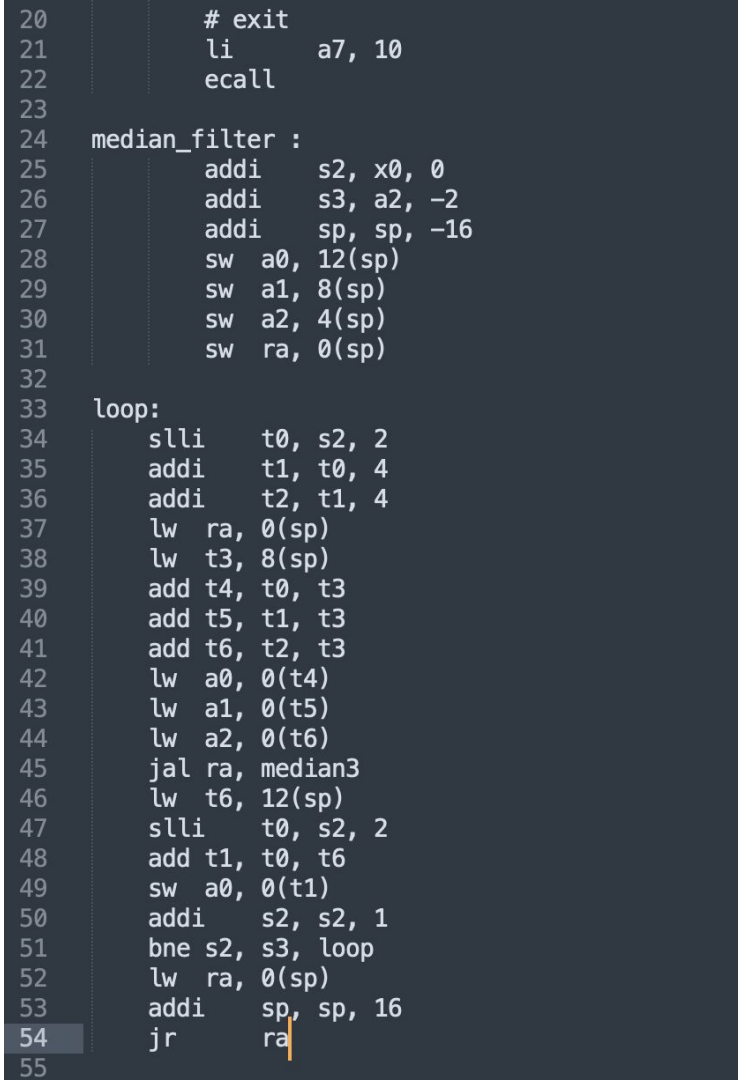


Figure 1: Dining Philosophers' problem with 5 philosophers

3 (40 points) Dining philosophers' problem

Computer Architecture: RISC-V

Learned RISC-V ASM, pipeline, I-Cache, D-Cache



```
20          # exit
21          li      a7, 10
22          ecall
23
24  median_filter :
25          addi    s2, x0, 0
26          addi    s3, a2, -2
27          addi    sp, sp, -16
28          sw  a0, 12(sp)
29          sw  a1, 8(sp)
30          sw  a2, 4(sp)
31          sw  ra, 0(sp)
32
33  loop:
34      slli    t0, s2, 2
35      addi    t1, t0, 4
36      addi    t2, t1, 4
37      lw  ra, 0(sp)
38      lw  t3, 8(sp)
39      add t4, t0, t3
40      add t5, t1, t3
41      add t6, t2, t3
42      lw  a0, 0(t4)
43      lw  a1, 0(t5)
44      lw  a2, 0(t6)
45      jal ra, median3
46      lw  t6, 12(sp)
47      slli    t0, s2, 2
48      add t1, t0, t6
49      sw  a0, 0(t1)
50      addi    s2, s2, 1
51      bne s2, s3, loop
52      lw  ra, 0(sp)
53      addi    sp, sp, 16
54      jr      ra
55
```

Computer Architecture: Riscy-uconn (CSE4302)

Implemented a virtual machine for Riscy-uconn architecture in C99.

Superscalar, OOO Execution, Cache, Hazards

```c
/**
 * University of Connecticut
 * CSE 4302 / CSE 5302 / ECE 5402: Computer Architecture
 * Fall 2021
 * Stephen Duncanson
 *
 * Programming Assignment 2: Pipelined Simulator
 *
 * riscy-uconn: sim_stages.c
 *
 */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include "instruction_map.h"
#include "sim_core.h"
#include "sim_stages.h"


/**
 * Debug flags
 */
int debug = 1;          // Set to 1 for additional debugging information.
int pipe_trace = 1;     // Set to 1 for pipe trace.

/* Pipeline related */
int pipe_stall  = 0;
int br_taken    = 0;
int lw_in_exe   = 0;
int we_exe      = 0, ws_exe = 0, dout_exe = 0;
int we_mem      = 0, ws_mem = 0, dout_mem = 0;
int we_wb       = 0, ws_wb  = 0, dout_wb  = 0;
```

Computer Architecture: Riscy-uconn (CSE4302)

Implemented a virtual machine for Riscy-uconn architecture in C99.

Stage 1) Fetch

```
36  /**
37   * Fetch stage implementation.
38   */
39  unsigned int fetch(unsigned int fetch_in) {
40    unsigned int inst = 0;
41    if (pipe_stall==1)
42      return fetch_in;
43    else if (br_taken==1)
44      return 0x00000000;
45    else
46      return inst = memory[pc / 4];
47  }
```

Computer Architecture: Riscy-uconn (CSE4302)

Implemented a virtual machine for Riscy-uconn architecture in C99.

Stage 2) Decode

```
/**
 * Decode stage implementation
 */
struct State decode(unsigned int fetch_out) {
  struct State decode_out = {0};

  /* Local signals: Assert when all registers except $0 are read. */
  int re1 = 0; // assert when rs is read
  int re2 = 0; // assert when rt is read
  int fwd_reg1 = 0; // assert if we have gotten a forwarded value in re1
  int fwd_reg2 = 0; // assert if we have gotten a forwarded value in re2
  int fwd_buf1 = 0; // buffer to hold forwarded value for rs
  int fwd_buf2 = 0; // buffer to hold forwarded value for rt
  pipe_stall = 0; // we assume no stall until we find one
  br_taken = 0; // assume branch is not taken unless it is

  /* Bitmasks to extract bits in fetch'd instruction */
  unsigned int opcode_mask = 0xFC000000; // RIJ type

  unsigned int func_mask = 0x0000003F;   // RIJ type
  unsigned int addr_mask = 0x03FFFFFF;   // J type
  unsigned int imm_mask = 0x0000FFFF;    // I type
  unsigned int rs_mask = 0x03E00000;     // RI type
  unsigned int rt_mask = 0x001F0000;     // RI type
  unsigned int rd_mask = 0x0000F800;     // R type
  unsigned int sa_mask = 0x000007C0;     // R type

  decode_out.inst = fetch_out;

  /* Decode: Put all fields into State struct.
     Update read signals re1/re2 if inst reads rs or rt reg */
  int opcode = fetch_out >> 26;
  decode_out.opcode = opcode;
  decode_out.rs = (fetch_out & rs_mask) >> 21;
  decode_out.rt = (fetch_out & rt_mask) >> 16;
```

Computer Architecture: Riscy-uconn (CSE4302)

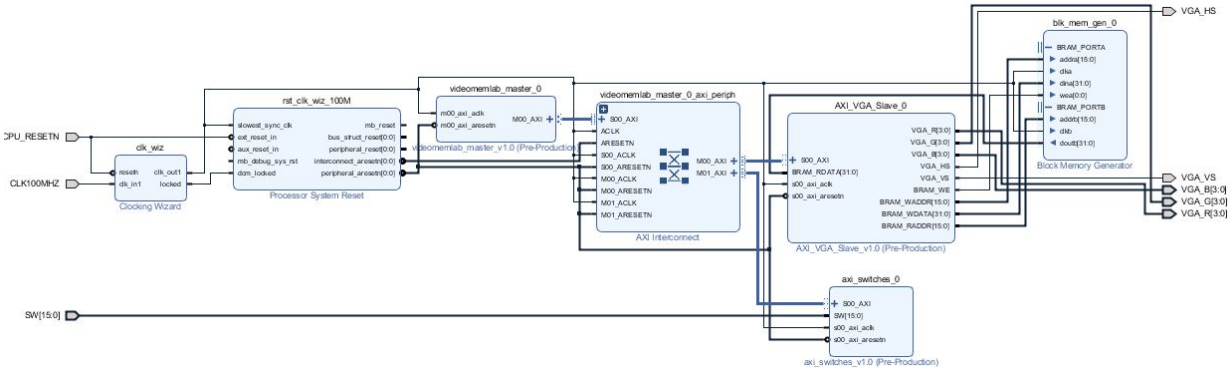
Implemented a virtual machine for Riscy-uconn architecture in C99.

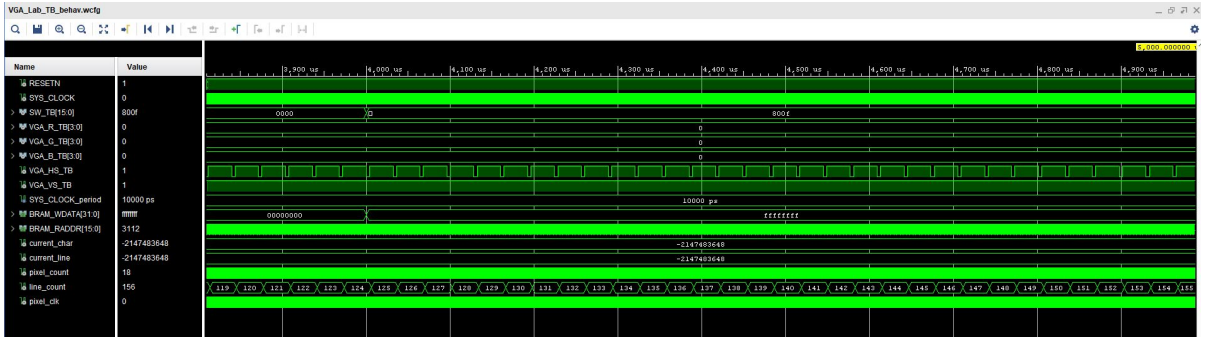
Stage 2) Decode

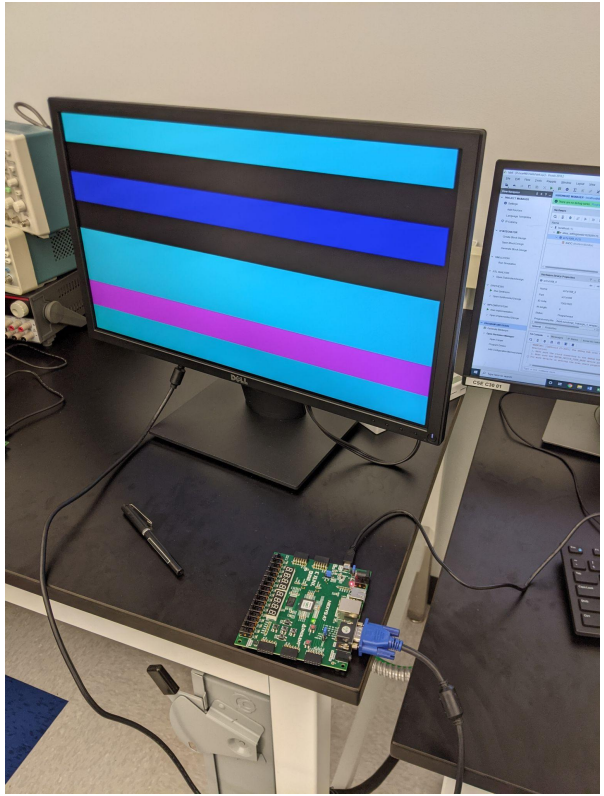
```
85    switch(decode_out.opcode) {
86    case 0x00: // R-Type
87      decode_out.rd = (fetch_out & rd_mask) >> 11;
88      decode_out.sa = (fetch_out & sa_mask) >> 6;
89      decode_out.func = (fetch_out & func_mask);
90      if ((decode_out.func==0x00||decode_out.func==0x02) && decode_out.rt!=0)
91        re2 = 1; // SLL/SRL only read rt register
92      else if (decode_out.func==0x08 && decode_out.rs!=0)
93        re1 = 1; // JR only reads rs register
94      // otherwise R type reads both, check if they are zero tho
95      if (decode_out.rs!=0)
96        re1 = 1;
97      if (decode_out.rt!=0)
98        re2 = 1;
99      break;
100   case BEQ:
101   case BNE:
102   case SW: // sw reads rt per bdag
103     if (decode_out.rt!=0)
104       re2 = 1; // BEQ/BNE read both rs/rt
105   case LW:
106   case ANDI:
107   case ADDI:
108   case ORI:
109   case SLTI:
110     if (decode_out.rs!=0)
111       re1 = 1;
112   case LUI:
113     decode_out.imm = (fetch_out & imm_mask);
114     break;
115   case JAL:
116     decode_out.jmp_out_31 = pc;
117   case J:
118     decode_out.rs = 0;
119     decode_out.rt = 0;
120     break;
121   }
```

FPGA Lab (ECE 4401)

Vivado, VGA output lab6!



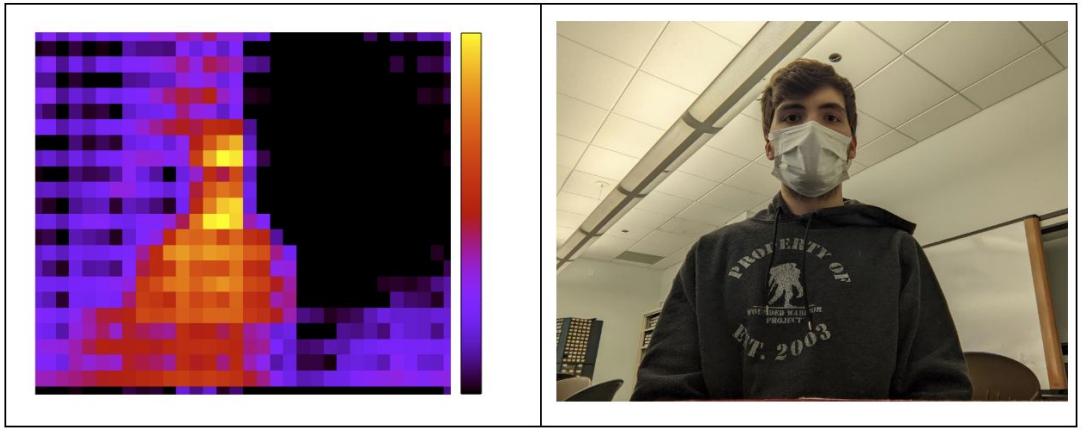


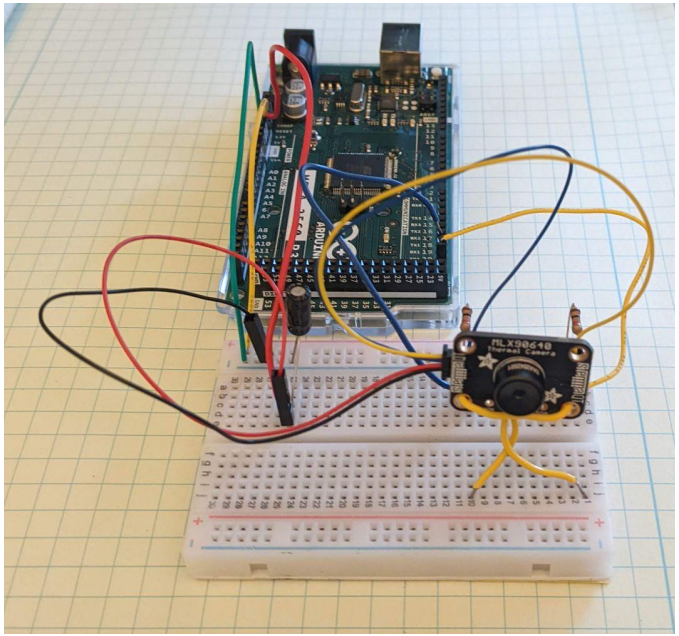


ECE 4901 / 4902, Senior Design

An IR sensor connected to microcontroller to detect falls in the household.

My role: interfacing the sensor and board, and getting data back to a host computer over a serial port.
In C.





Spring 2022

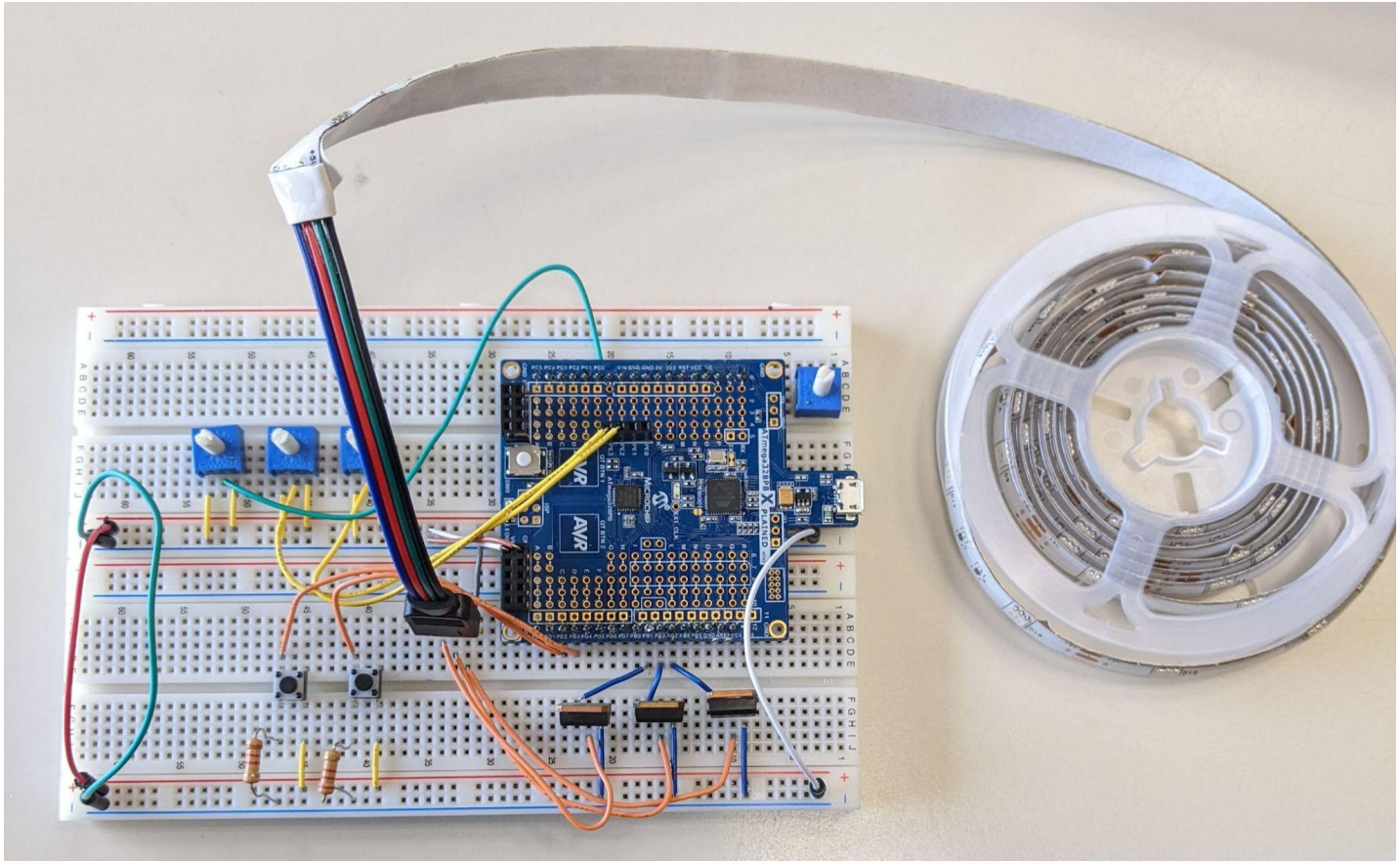
Final semester!

CSE	4300	Operating Systems	3.00	3.00	B+	9.900
ECE	3411	Microprocessor Application Lab	3.00	3.00	B+	9.900
ECE	3421	VLSI Design & Simulation	4.00	4.00	B	12.000
ECE	4131	Intr Digital Signal Processing	3.00	3.00	B-	8.100
ECE	4902	ECE Design II	3.00	3.00	A-	11.100

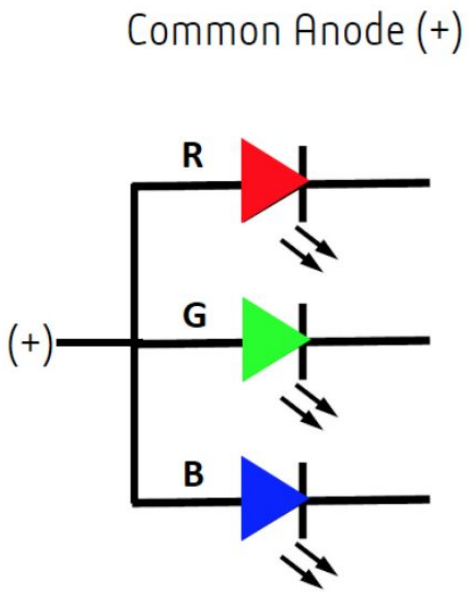
Microprocessor Applications: Project 5

Used Xplain Miniboard (pictured right) ATMega AVR 328PB.

I2C (TWI), SPI, PWM, UART, GPIO, ADC, Interrupts



Non-addressable led strips (common anode)



4 pins: +5V R G B: brightest when any light pin is grounded. Implication on software PWM:

```
 99  static inline void initTimers(void) {
100      // 8-bit fast PWM, TOP = OCR1A
101      TCCR1A |= (1 << WGM10);
102      TCCR1B |= (1 << WGM12);
103      // PWM Freq = F_CPU / 8 * 256
104      TCCR1B |= (1 << CS11);
105      // PWM output on OCR1A, non-inverting mode lets you do RGB colors 'directly'
106      TCCR1A |= (1 << COM1A1);
107      TCCR1A |= (1 << COM1B1);
108
109      TIMSK1 = (1 << TOIE1);
110      OCR1A = 100;
111      OCR1B = 100;
112
113      TIMSK2 = (1 << TOIE2);
114      TCCR2A |= (1 << WGM20);
115      TCCR2A |= (1 << WGM21);
116      TCCR2B |= (1 << CS21);
117      TCCR2A |= (1 << COM2A1);
118      OCR2A = 100;
119  }
```

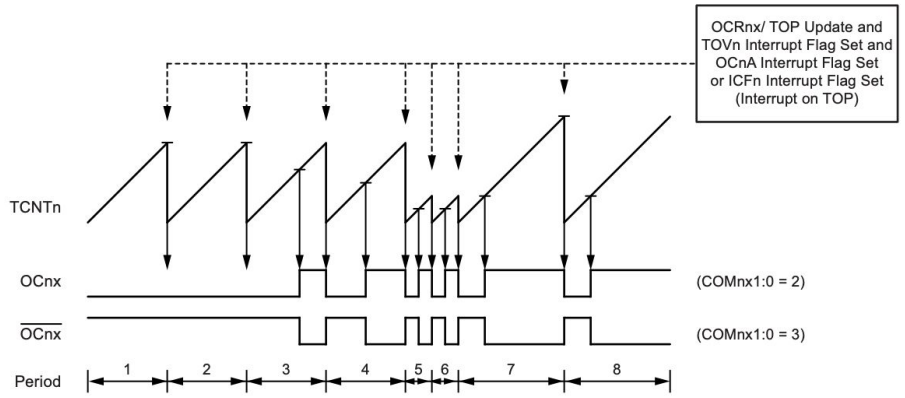
Non-addressable led strips (common anode)

implication on software PWM:
Non-inverting: toggle on at bottom, turn off at compare match. Higher voltage->higher OCRnx value -> more time on -> higher ‘analog value’ -> dimmer

```
// button not pushed, get from pots
else {
    red_duty = (int)(r_voltage * 50) % 256;
    green_duty = (int)(g_voltage * 50) % 256;
    blue_duty = (int)(b_voltage * 50) % 256;
}
```

```
 99  static inline void initTimers(void) {
100      // 8-bit fast PWM, TOP = OCR1A
101      TCCR1A |= (1 << WGM10);
102      TCCR1B |= (1 << WGM12);
103      // PWM Freq = F_CPU / 8 * 256
104      TCCR1B |= (1 << CS11);
105      // PWM output on OCR1A, non-inverting mode lets you do
106      TCCR1A |= (1 << COM1A1);
107      TCCR1A |= (1 << COM1B1);
108
109      TIMSK1 = (1 << TOIE1);
110      OCR1A = 100;
111      OCR1B = 100;
112
113      TIMSK2 = (1 << TOIE2);
114      TCCR2A |= (1 << WGM20);
115      TCCR2A |= (1 << WGM21);
116      TCCR2B |= (1 << CS21);
117      TCCR2A |= (1 << COM2A1);
118      OCR2A = 100;
119  }
```

Figure 15-7. Fast PWM Mode, Timing Diagram

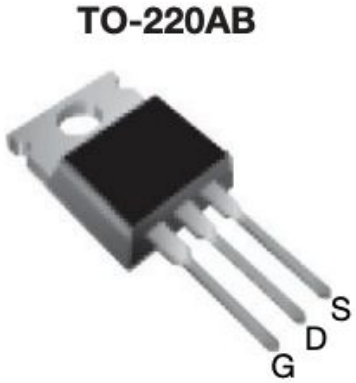


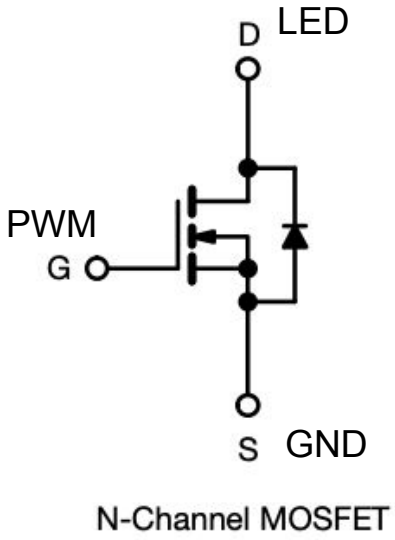
Why 8-bit PWM?

```
typedef struct color {
    char red;
    char green;
    char blue;
} Color;

// 12 colors which get switched between when you press the int1 button
Color white = {255,255,255};
Color red = {255, 0, 0};
Color green = {0, 255, 0};
Color blue = {0, 0, 255};
Color orange = {250, 104, 0};
Color amber = {240, 163, 10};
Color yellow = {227, 200, 0};
Color teal = {0, 171, 169};
Color indigo = {105, 0, 255};
Color emerald = {0, 138, 0};
Color magenta = {216, 0, 115};
Color violet = {170, 0, 255};
Color pink = {244, 144, 208};
```

Cool consequent of 8-bit pwm with these settings (and pin connections!): we can define colors exactly like we would on a website ;)

Not enough current..





Analog (non-addressable) LED strips seem to have greater demand for current.

Improve a little by stealing npn MOSFETs from my ECE 2001 component kit.

You may have also noticed an issue with my previous slide, higher OCRnx values should dim lights, not be RGB accurate. This is why: pwm signal is at gate of npn.

Measured and does help a little, would be better if I had higher voltage source or different FET I think.

Using interrupts instead of polling

```
68  ISR(ADC_vect) {
69      if (selected_pot == R_POT) {
70          r_voltage = ((float)ADC/1024*5);
71          selected_pot = G_POT;
72          ADMUX = (1<<MUX2) | (1 << MUX1) | (1 << MUX0) | (1 << REFS0);
73      }
74      else if (selected_pot == G_POT) {
75          g_voltage = ((float)ADC/1024*5);
76          selected_pot = B_POT;
77          ADMUX = (1 << REFS0);
78          }
79      else {
80          b_voltage = ((float)ADC/1024*5);
81          selected_pot = R_POT;
82          ADMUX = (1<<MUX2) | (1 << MUX1) | (1 << REFS0);
83      }
84      ADCSRA |= (1 << ADSC);
85  }
```

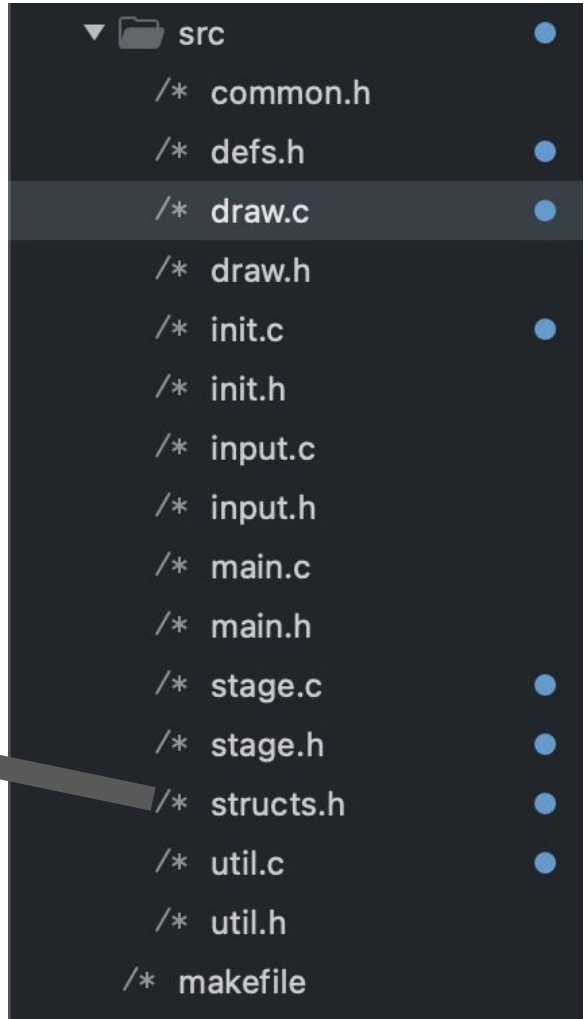
```
94  ISR(TIMER0_COMPA_vect) {
95      current_ms++;
96  }
```

Operating Systems: Fun final project

Making a game with C and SDL library (we also learned how to write system calls but this is more fun)

Source file tree & Lessons learned in organization

```c
typedef struct {
    void (*logic)(void);
    void (*draw)(void);
} Delegate;

typedef struct {
    SDL_Renderer* renderer;
    SDL_Window* window;
    Delegate delegate;
    int keyboard[MAX_KEYBOARD_KEYS];
} App;

typedef struct Entity Entity;

struct Entity {
    float x;
    float y;
    int w;
    int h;
    float angle;
    float dx;
    float dy;
    int health;
    int reload;
    int side;
    int e_type;
    SDL_Texture* texture[8];
    Entity* next;
};

typedef struct {
    Entity fighterHead, *fighterTail;
    Entity bulletHead, *bulletTail;
} Stage;
```



Main game loop:

Atexit is nice

Try to minimize the spaghetti
with a “delegate”

Easy frame rate with library timer!

```
int main(int argc, char* argv[]) {

    long then;
    float remainder;

    memset(&app, 0, sizeof(App));

    initSDL();
    atexit(cleanup);

    initStage();

    then = SDL_GetTicks();
    remainder = 0;

    while(1) {
        prepareScene();

        doInput();

        app.delegate.logic();
        app.delegate.draw();

        presentScene();

        capFrameRate(&then, &remainder);
    }
    return 0;
}
```

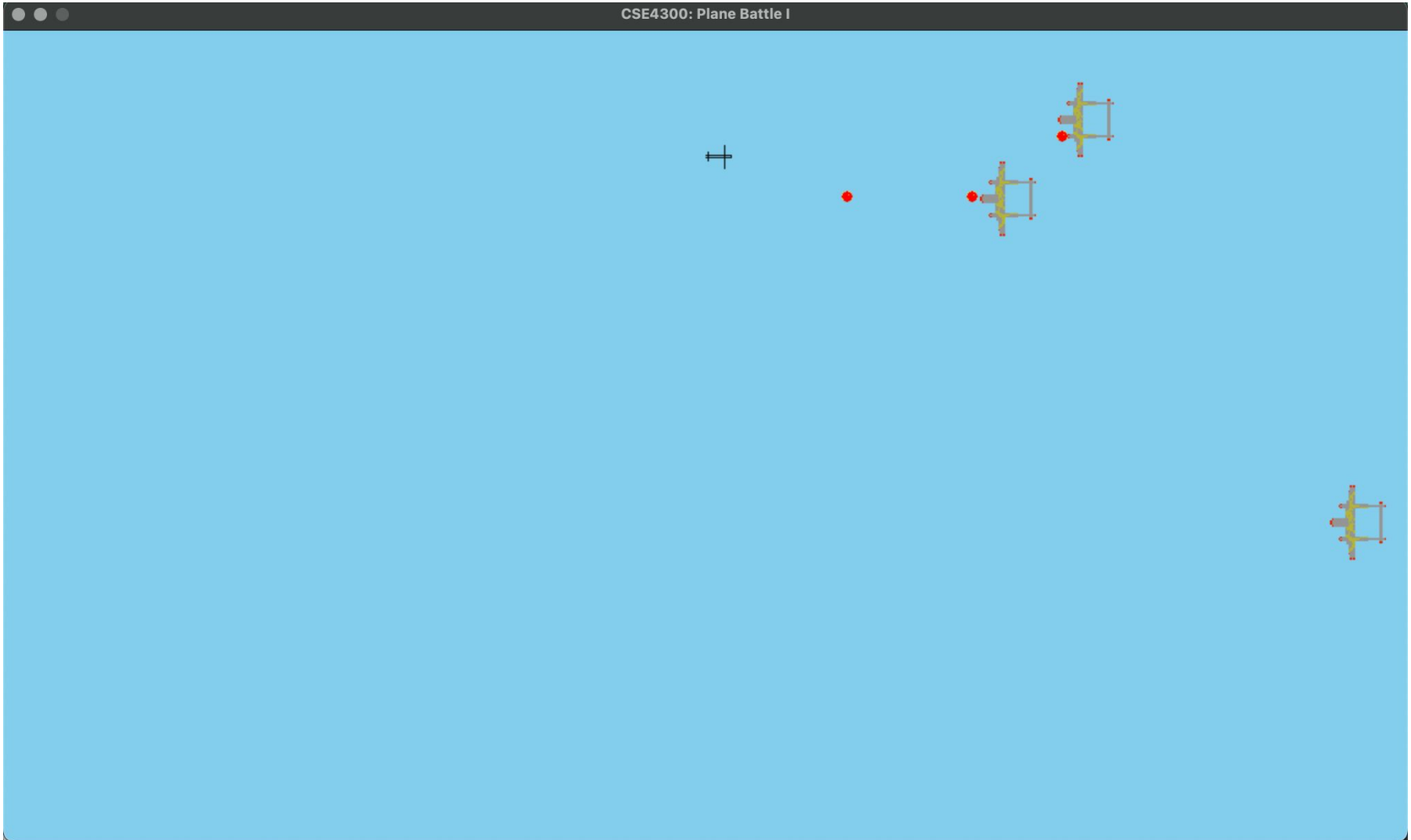
Logic and draw call other atomic functions

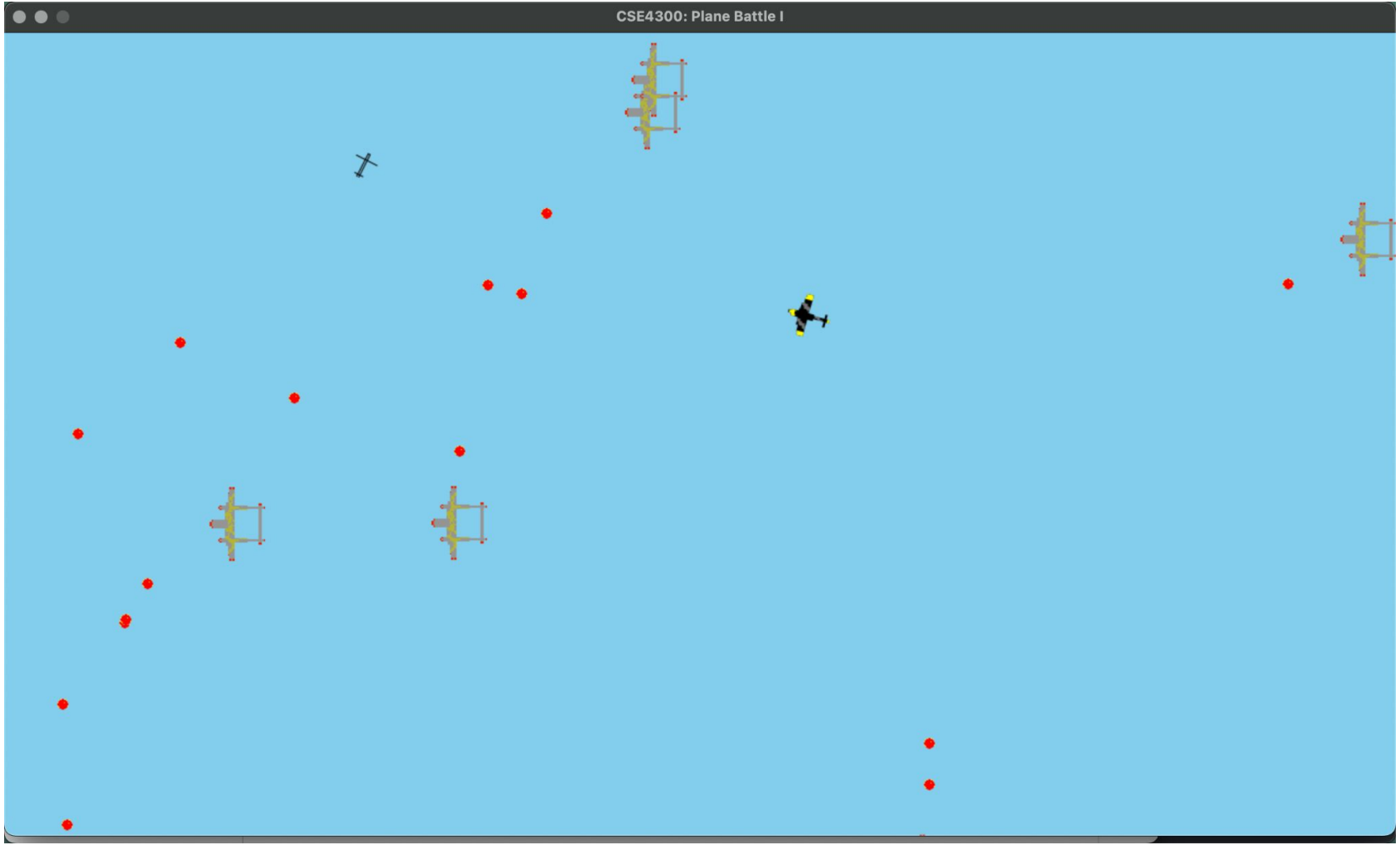
```
static void logic(void) {
    doPlayer();

    doEnemies();

    doFighters();

    doBullets();

    spawnEnemies();

    clipPlayer();

    if (player == NULL && --stageResetTimer <= 0) {
        resetStage();
    }
}
```

```
static void draw(void) {
    drawFighters();
    drawBullets();
}
```

doPlayer()
The first function called by logic
Function pointer
Can see throttle and turn “tank” controls

```
static void doPlayer(void) {
    if (player != NULL) {
        //player->dx = player->dy = 0;

        if (player->reload > 0) {
            player->reload--;
        }
        player->dy = sin(player->angle)*PLAYER_SPEED;
        player->dx = cos(player->angle)*PLAYER_SPEED;

        if (app.keyboard[SDL_SCANCODE_UP]) {
            player->dy += 0.5*sin(player->angle)*PLAYER_SPEED;
            player->dx += 0.5*cos(player->angle)*PLAYER_SPEED;
        }
        if (app.keyboard[SDL_SCANCODE_DOWN]) {
            player->dy -= 0.5*sin(player->angle)*PLAYER_SPEED;
            player->dx -= 0.5*cos(player->angle)*PLAYER_SPEED;
        }
        if (app.keyboard[SDL_SCANCODE_RIGHT]) {
            player->angle+=0.025;
            //player->dx = PLAYER_SPEED;
        }
        if (app.keyboard[SDL_SCANCODE_LEFT]) {
            player->angle-=0.025;
            //player->dx = -PLAYER_SPEED;
        }
        if (app.keyboard[SDL_SCANCODE_SPACE] && player->reload == 0) {
            fireBullet();
        }
        if (player->angle > 2*PI) {
            player->angle-=2*PI;
        }
        if (player->angle < 0) {
            player->angle+=2*PI;
        }
    }
}
```


CSE4300: Plane Battle I


CSE4300: Plane Battle I

Thank you!

Questions?