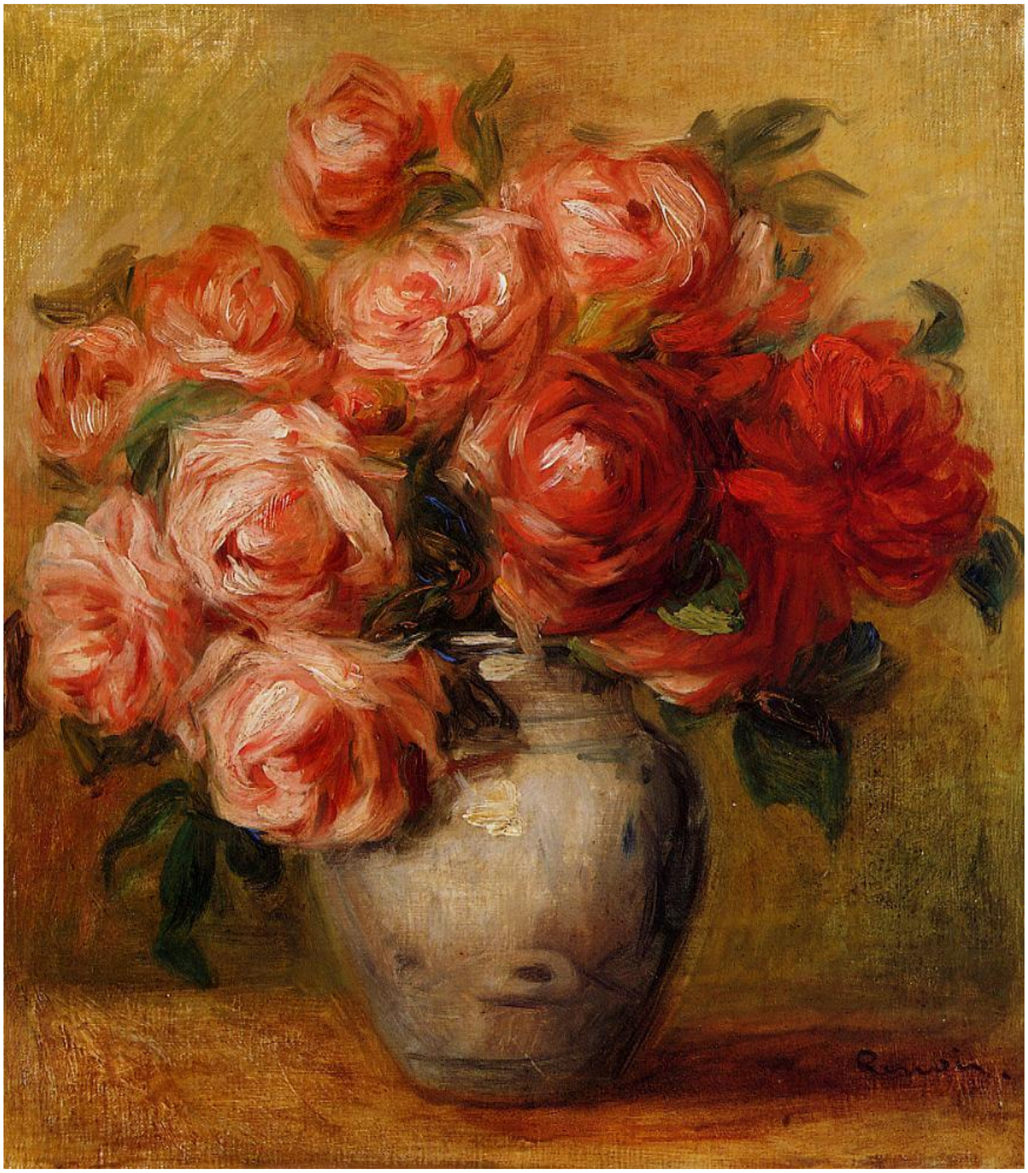


Physics 398DLP
Design Like a Physicist

Fall 2018

George Gollin
University of Illinois at Urbana-Champaign

Physics 398DLP
George Gollin
University of Illinois at Urbana-Champaign
Fall 2018
Fridays, 1 pm – 5 pm; 3 credit hours.

Introduction and Syllabus

©George Gollin, 2018

Introduction: project physics

Some years ago Carl Wieman won the Nobel Prize for creating a Bose-Einstein condensate in a dilute cloud of 2,000 atoms. At the time he was a professor at the University of Colorado, and had noticed that his physics students appeared to undergo a dramatic transition during the first year of graduate school. As undergraduates they would attend lecture-based classes and master course content by listening to their professors and slogging through weekly problem sets. (You all know what this is like!) By the end of the semester, most of the class would understand most of the material, but would find it difficult to integrate it into a coherent picture of, say, classical electrodynamics. And a semester after a course had ended, most students would not have retained their mastery of the topic. They would find it difficult to apply the material in, say, a lab course. But after a year of graduate school—during which students would work on difficult material without the distracting edge effects of 50-minute class periods—their competence at navigating confusing subjects and difficult problems would increase enormously.

Wieman thought that teaching physics to undergraduates in a manner that more closely resembled graduate education might be beneficial. He began to explore project-based courses, in which students would learn physics by mastering what they needed to complete tasks that were more like research projects than was usually true in undergraduate instruction. The results were dramatic.

You've already had some experience with this instructional mode if you've taken Physics 298owl from me. It's different from fighting to stay awake for an hour in lecture, then sifting through the wreckage to extract what you need to do the homework assignment!

You will be performing the one-semester analog of a PhD research thesis: defining a measurement to be performed, designing and building an instrument that might be capable of recording data necessary for the measurement, testing your device, doing the field work to record valid data, then analyzing the data to form supportable, reproducible conclusions. If all goes well, you'll find this so captivating that it will be hard to put your work aside to attend to your other academic obligations. I suspect it is this strong engagement with a project that drives the transition from an undergraduate level of skill to the expert mastery typical of graduate researchers.

Your device will comprise an embedded processor—a Microchip Technology Inc. ATmega2560 microcontroller on an Arduino Mega 2560 board—interfaced to a suite of sensors built onto small "breakout" printed circuit boards. The Arduino's USB interface will allow you to download your (compiled) programs to the microcontroller and communicate with processor through a serial interface.

After selecting your research partners and choosing a project, you'll begin assembling your instrument on a breadboard. You will develop the programs necessary to drive the various sensors, integrate them into a data acquisition system of your own design, build a more robust version of your device on a printed circuit board, 3D-print a case for it, and venture out into the world to do field work and record data. You will analyze your data, draw (and justify) conclusions, and document (and present) your findings. In the interest of efficiency, I encourage you to use in your design as much public-domain material as you are able to find. There is, in this course, no reason to reinvent the wheel.

Prerequisites

You must already know how to program. If you've learned to code in python or C/C++, or Java, or some other language, you'll do fine. CS 101, CS 125, and Physics 298owl are suitable prerequisites. It's also fine if you've learned on your own. If you've never programmed before, consider delaying enrollment in Physics 398DLP until after you've done some coding.

You must have a basic working knowledge of introductory physics at the level of Physics 211 and Physics 212. More is better, though not necessary.

We are not building robots

Physics 398DLP is not a course in robot building. That would be an engineer thing, and we are physicists, not engineers. We are going to tackle measurements that—if they prove feasible—might make our corner of the world a little bit better. If we ***did*** build a robot, it would be to accomplish a significant end, for example recognizing the onset of a potentially catastrophic fall by an elderly person.

In Physics 398DLP you'll construct a hand-held device loaded with inexpensive sensors that are interrogated by a microcontroller—a small computer larded with additional features such as timers and analog-to-digital converters—and write the data acquisition software necessary to perform the measurements associated with your project. You'll assemble a prototype on a breadboard, construct a final (electrically equivalent) version on a printed circuit board, use a 3D printer to build a case for it, do field work, then write analysis code to understand what conclusions can be drawn from your data. You'll write a report presenting your results and justifying your conclusions, publish it to the web, and send it to the appropriate recipient—the Illinois Department of Transportation, for example—and request a meeting to discuss your findings.

We will loan you the parts and tools necessary to construct the prototype, and will expect you to return these at the end of the course. But—at least in this initial offering of the course—we will give you what you need to build the PCB version, and let you keep it at the end of the term. (If you withdraw from the course we'll want you to return everything we've given you.)

The intellectual tradition in physics is for researchers to build their own instruments (buying off-the-shelf parts when available), ultimately creating sophisticated devices to perform the measurements that will tell us about the physics we are researching. It is not like this in all fields; my wife's background is in bio-inorganic chemistry, and she would assemble reactors from stock components, then run reaction products through spectrometers built by vendors like Varian and Hitachi.

So you'll be following the physics tradition, and you will be working in close collaboration with one or two other students.

Some possible projects

Some of the projects are probably best imagined as feasibility studies that might inform the design of a more definitive future measurement. We will see how it goes!

Here are some that I have in mind. You are free to suggest other possibilities, though I reserve the right to veto anything that I feel is too difficult or too expensive.

- Noise mitigation in public spaces (e.g. the *Radio Maria* restaurant)
- Mapping track irregularities and anomalous accelerations on Amtrak trains
- Green house gas production by livestock
- The acoustic properties of the Krannert Center's Great Hall as a function of position, temperature, humidity, and barometric pressure
- Fall recognition and mitigation

- (Potentially injury-producing) accelerations experienced by athletes and dancers
- Inexpensive range and force sensors for smart prosthetic limbs
- Mitigation of heat leakage through exterior walls by installation of spot insulation
- Vertical temperature profiles in large lecture halls
- Quantitative studies of how technicians "voice" instruments like pianos
- Mapping the temperature and humidity above a corn field (this might require a drone to carry the data logger);
- Pressure gradients inside a residence as an indication of wind-induced air leakage;
- Noise and pressure profiles in the vicinity of wind turbines.

The style in which we will work

"DLP" stands for "Design Like a Physicist." That's a reasonably descriptive term for how we will go about things, though it wasn't my first choice for the three-character course identifier. Here is what I mean. If you took Physics 298owl from me you'll remember that I had you hand-code a lot of algorithms—integrators, Fourier transforms—that could also be found in professionally produced libraries. For pedagogical purposes, I had you reinventing a lot of wheels.

That's not how I've gone about my own research. If there's a pre-coded numerical algorithm that I can use, I'll appropriate it, generally putting proper attribution to its source in comments in my own code. If there's a circuit I need that's described in an engineering web site, I'll use it. Proper attribution can be placed on my schematic diagram. Sometimes it might be difficult to publish the source attribution—the 3D STL files you'll create for TinkerCad projects—but do keep in your own notes information about where you have found useful material.

You will keep track of your efforts in an electronic diary in which you describe your work, useful revelations, and calculations. Put into it screen shots of useful stuff. I do not want this to be anything fancy, but the diary should be cumulative, rather than something you close off at the end of each class meeting. When your file becomes unwieldy, close it and start the next "volume." You should have your diary open while working on your project.

My preference is for you to use Microsoft Word, though if there's another word processing system you'd prefer to use, that's fine. You will be uploading a PDF version of the file to the course directory right before the beginning of each class. You should put notes about techniques you find (or invent) into your diary so you can find them later.

You may be tempted to use LaTeX for your diary. But unless you are exceptionally facile with LaTeX (and can already upload screen shots, for example), this will be a mistake. I am going to be fairly hardnosed about this: making a lovely version of your diary at the cost of ten minutes of extra time in class is unacceptable

We will be using several different IDEs—Integrated Development Environments—during the semester. I expect you to install and use these in your work. If there are other tools that you'd prefer to use, keep in mind that it will be hard for you to share material with other members of your group. If you insist on staying with these, I am not going to be happy to find you wasting time translating your work into a mutually acceptable format.

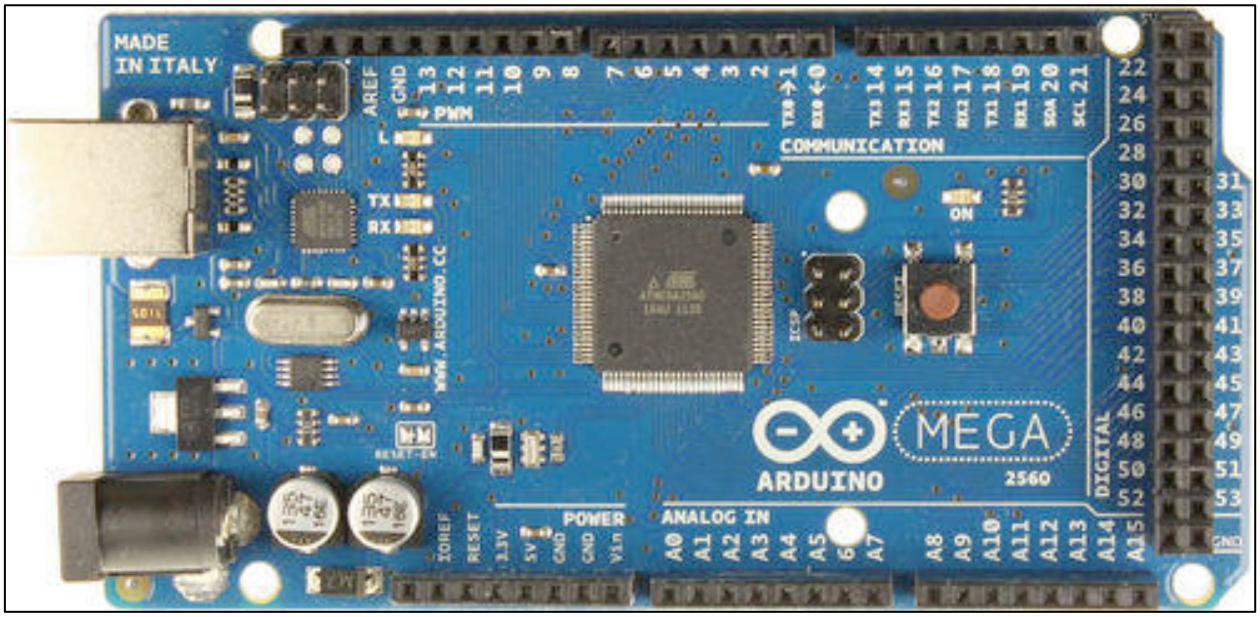
You will be working with things for which your understanding will often be a little blurry. That's OK, and in fact that's the usual state of things in research. Taking the time to understand every last detail about an IDE is a waste of your time: it is better to focus your efforts on getting by, on muddling through. You will get more done per week this way than you would if you spent the time to understand everything completely. There is too much to do, and far more interesting things to consider than the arcane details of SPI and I2C interfaces. You want to understand them well enough to work with them, but not to write—without reference to external sources—the definitive *Handbuch der Was Auch Immer* document.

I will expect you to have these windows open on your laptop in class at all times: (1) the IDE for whatever you're doing; (2) a browser window with which you can search for (and download) useful things; (3) a word processor window in which you are updating your diary.

The tools we will use

Arduino coding IDE

The heart of your data logger will be an Arduino Mega 2560 microcontroller board, shown here.

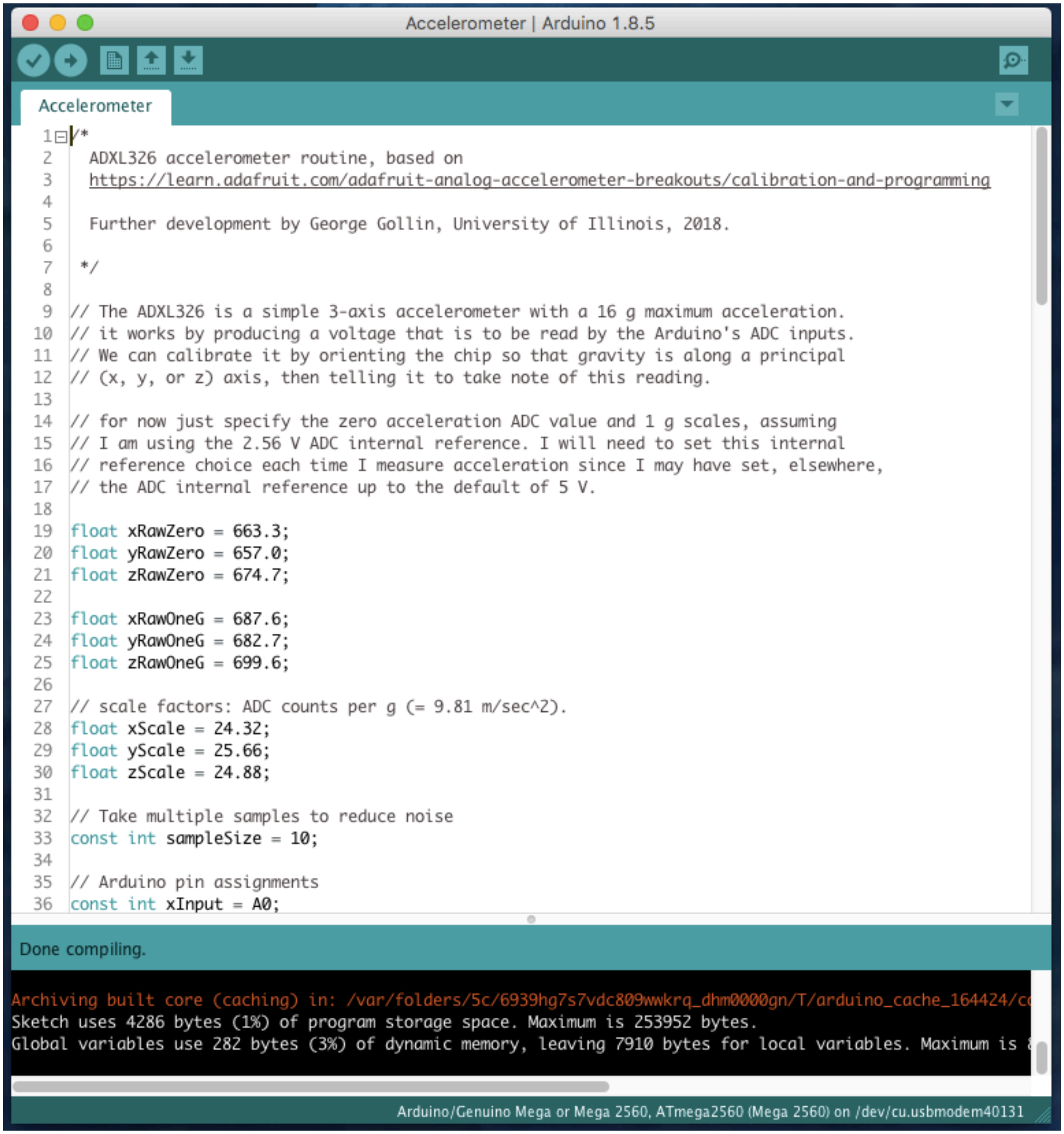


Arduino Mega 2560

It’s a remarkable little gizmo, featuring an Atmel Atmega2560 microcontroller running at 16MHz. The Atmega2560 has 256kB of flash memory in which your program will reside, along with 8kB of SRAM (static random access memory) in which will live the variables your program modifies as it executes. There are 16 analog inputs that feed an internal multiplexer whose output drives a successive approximation analog to digital converter. See https://store.arduino.cc/arduino-mega-2560-rev3 for more details.

The Arduino IDE is quite a bit simpler than Anaconda’s iPython IDE. Most of what you will see on your screen is an editor window in which you will create/modify C++ programs that you will compile and upload to the Arduino. See the screen shot, below. You’ll write and compile programs using the IDE, then upload them to the Arduino through a USB cable.

There isn’t a debugger, so you’ll be forced to print things to a “serial monitor” screen to keep track of what’s going on (and going wrong) in the code executed by the Arduino.


Accelerometer | Arduino 1.8.5
Accelerometer
Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on /dev/cu.usbmodem40131

```
/*
  ADXL326 accelerometer routine, based on
  https://learn.adafruit.com/adafruit-analog-accelerometer-breakouts/calibration-and-programming

  Further development by George Gollin, University of Illinois, 2018.

 */

// The ADXL326 is a simple 3-axis accelerometer with a 16 g maximum acceleration.
// it works by producing a voltage that is to be read by the Arduino's ADC inputs.
// We can calibrate it by orienting the chip so that gravity is along a principal
// (x, y, or z) axis, then telling it to take note of this reading.

// for now just specify the zero acceleration ADC value and 1 g scales, assuming
// I am using the 2.56 V ADC internal reference. I will need to set this internal
// reference choice each time I measure acceleration since I may have set, elsewhere,
// the ADC internal reference up to the default of 5 V.

float xRawZero = 663.3;
float yRawZero = 657.0;
float zRawZero = 674.7;

float xRawOneG = 687.6;
float yRawOneG = 682.7;
float zRawOneG = 699.6;

// scale factors: ADC counts per g (= 9.81 m/sec^2).
float xScale = 24.32;
float yScale = 25.66;
float zScale = 24.88;

// Take multiple samples to reduce noise
const int sampleSize = 10;

// Arduino pin assignments
const int xInput = A0;
```

Done compiling.

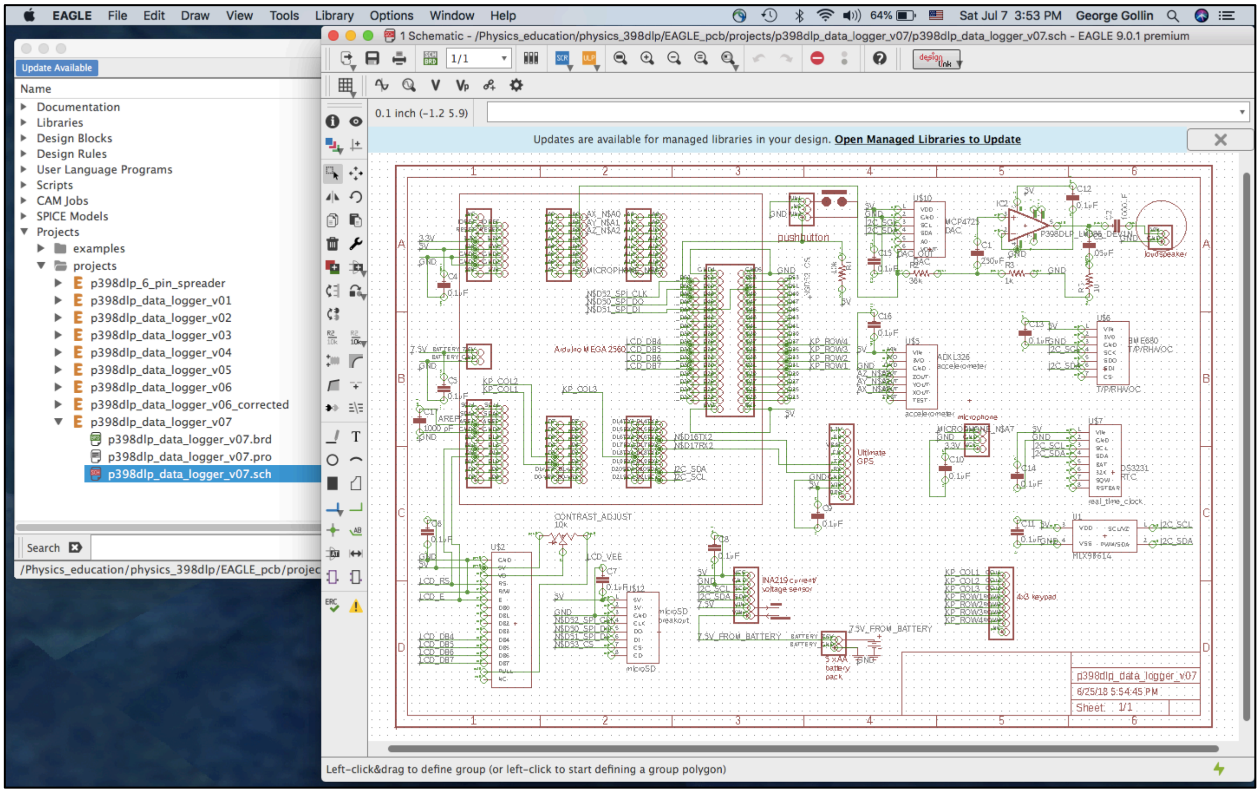
```
Archiving built core (caching) in: /var/folders/5c/6939hg7s7vdc809wwkrq_dhm0000gn/T/arduino_cache_164424/c
Sketch uses 4286 bytes (1%) of program storage space. Maximum is 253952 bytes.
Global variables use 282 bytes (3%) of dynamic memory, leaving 7910 bytes for local variables. Maximum is
```

Arduino Integrated Development Environment editor window

Schematic capture and PCB

As you assemble your prototype on a breadboard, you'll want to keep track of the wiring in your ever-more complex circuit. To do this, you'll register an account with Autodesk and use their free-for-three-years EAGLE schematic capture and printed circuit board tools.

Here's a screen shot of the the schematic capture tool. It'll take you a couple of hours to become proficient with it.



1 Schematic - /Physics_education/physics_398dlp/EAGLE_pcb/projects/p398dlp_data_logger_v07/p398dlp_data_logger_v07.sch - EAGLE 9.0.1 premium

Updates are available for managed libraries in your design. Open Managed Libraries to Update

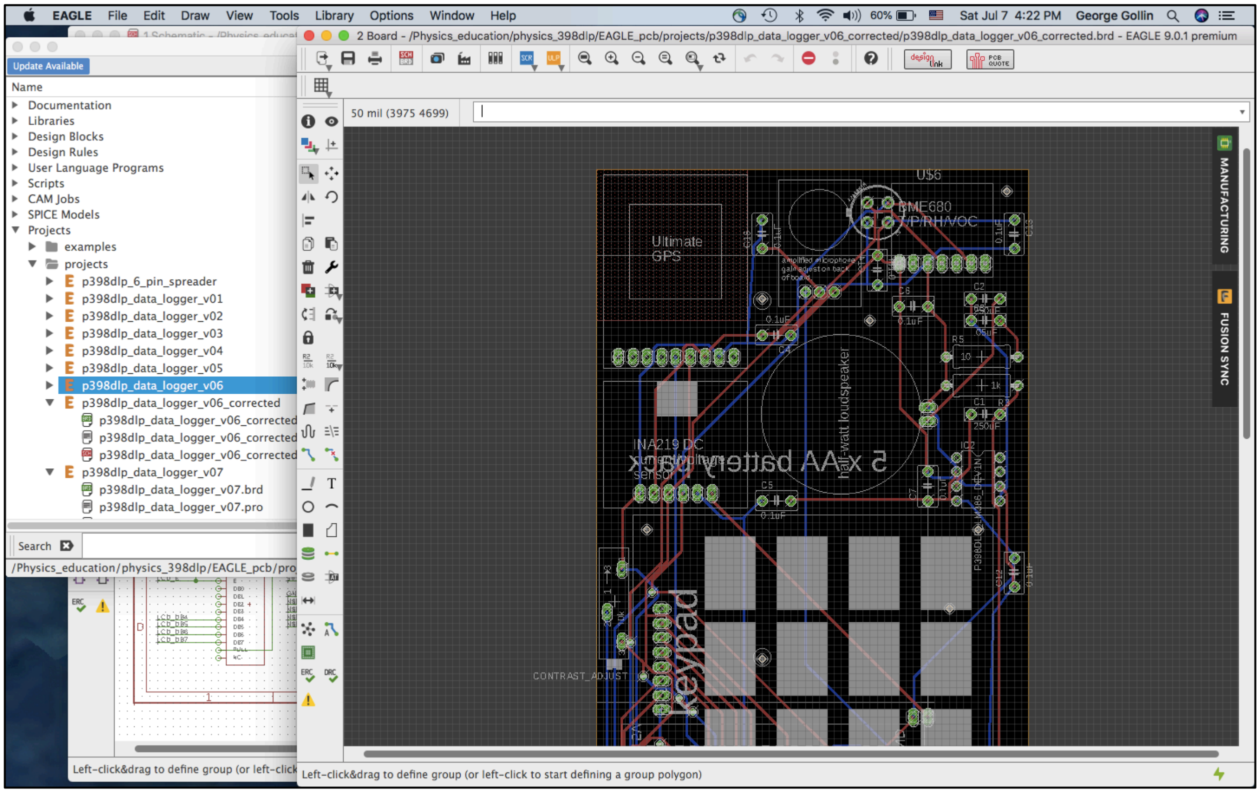
Left-click&drag to define group (or left-click to start defining a group polygon)

EAGLE schematic capture tool

If you're interested, you could try using the EAGLE printed circuit layout tool to design a circuit board that could be fabricated in the ECE shop. Unfortunately, the ECE facility isn't able to produce "plated-through holes" that connect signal traces on the top and bottom layers. That makes a board produced this way considerably less reliable as it ages.

I plan to lay out a PCB that is sufficiently general that everyone can use it, installing the sensors that you plan to use, and omitting those that you don't. I'll have a few dozen copies of this fabricated by a commercial vendor so that we'll have plated-through holes and "silkscreen layers" at our disposal.

Here is a screen shot of the EAGLE PCB layout tool.



2 Board - /Physics_education/physics_398dlp/EAGLE_pcb/projects/p398dlp_data_logger_v06_corrected/p398dlp_data_logger_v06_corrected.brd - EAGLE 9.0.1 premium

Left-click&drag to define group (or left-click

Left-click&drag to define group (or left-click to start defining a group polygon)

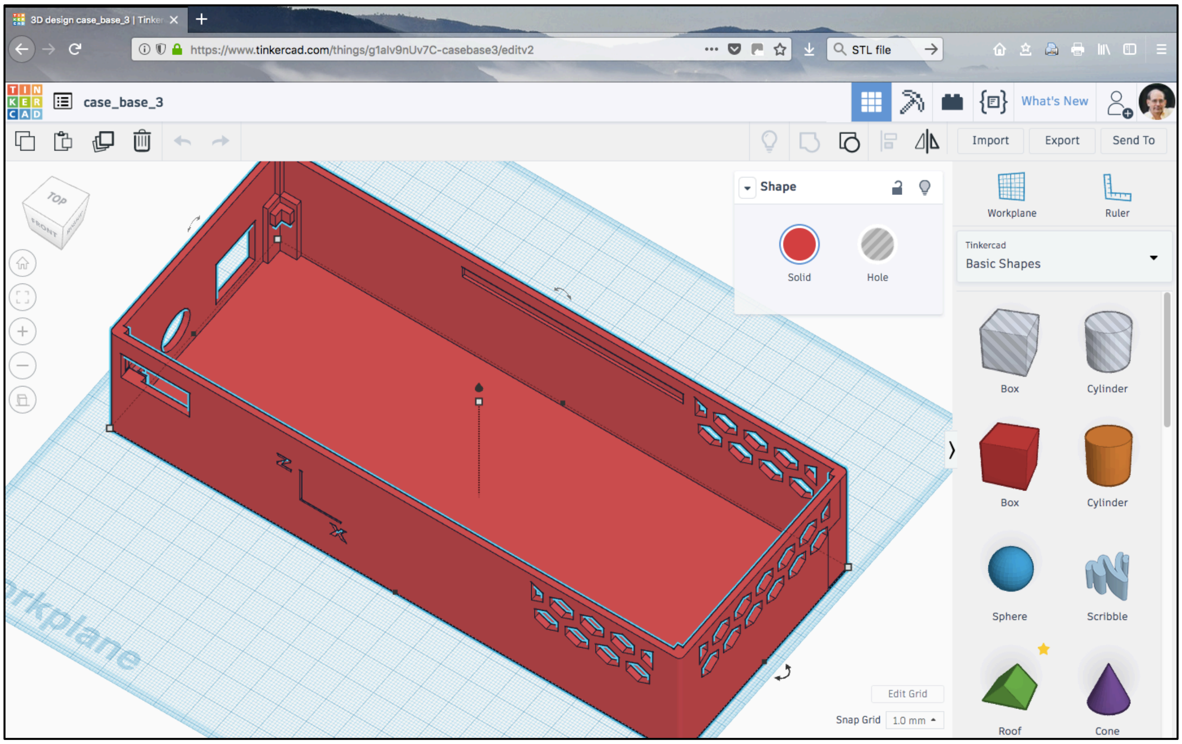
EAGLE PCB layout tool

3D printing

You will use TinkerCad—another Autodesk product—to design a case for your device. TinkerCad will produce an STL (stereolithography) file that we will feed to Cura, also an Autodesk application, to convert the STL file into a gcode file of instructions to be executed by a 3D printer.

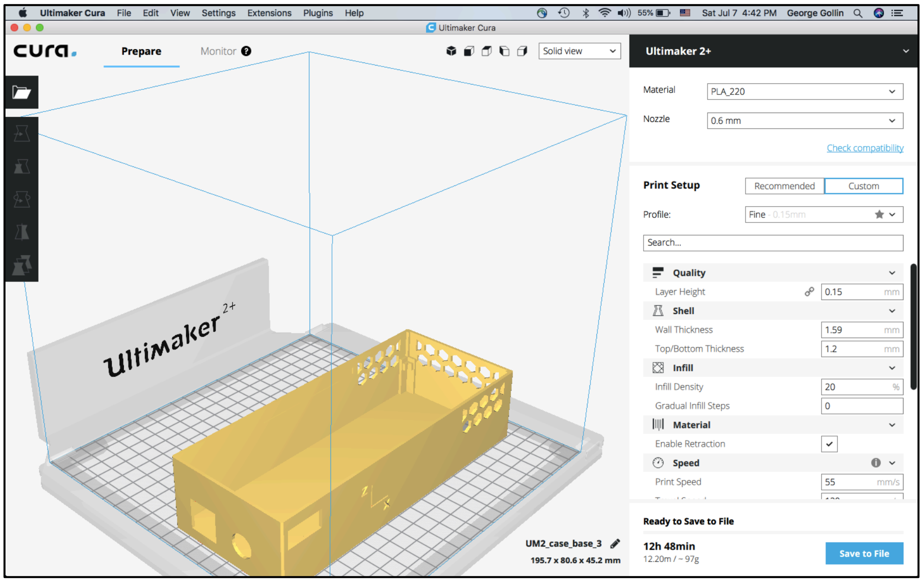
For practice I will have you design a small box to hold the sensor "breakout boards" and Arduino; I can print these for you on an Ultimaker 2+ printer in my office. When it comes time to fabricate the case for your data logger, though, we'll negotiate with the Business School's MakerLab management to take over half of their machines, since your case will probably take 24 hours to print, and will require dedicated use of a printer.

Here's a TinkerCad screen shot of my case design. TinkerCad is a web-based tool, easy to learn, and great fun to use.



TinkerCad screen shot

Here's the Cura window. The latest version is 3.4.1; you'll need to make sure you have the correct nozzle diameter (0.6 mm for the machine in my office, but probably 0.4 mm for the printers in MakerLab), as well as the "Preferences → Printers → Machine Settings → Printer → Origin at center" box unchecked..



Material: PLA_220

Nozzle: 0.6 mm

Print Setup: Custom

Profile: Fine - 0.15mm

Quality

Layer Height: 0.15 mm

Shell

Wall Thickness: 1.59 mm

Top/Bottom Thickness: 1.2 mm

Infill

Infill Density: 20 %

Gradual Infill Steps: 0

Material

Enable Retraction: ☑

Speed

Print Speed: 55 mm/s

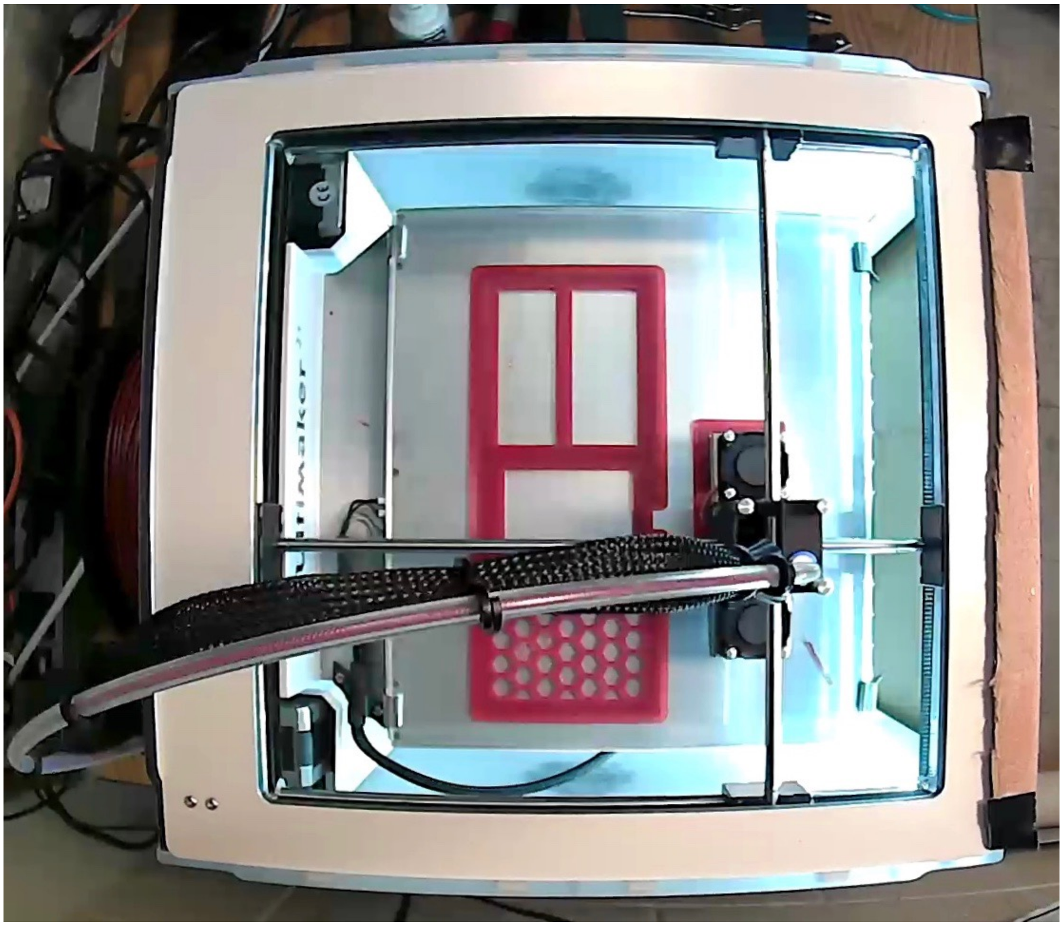
Ready to Save to File

12h 48min

12.20m / ~ 97g

Cura screen shot

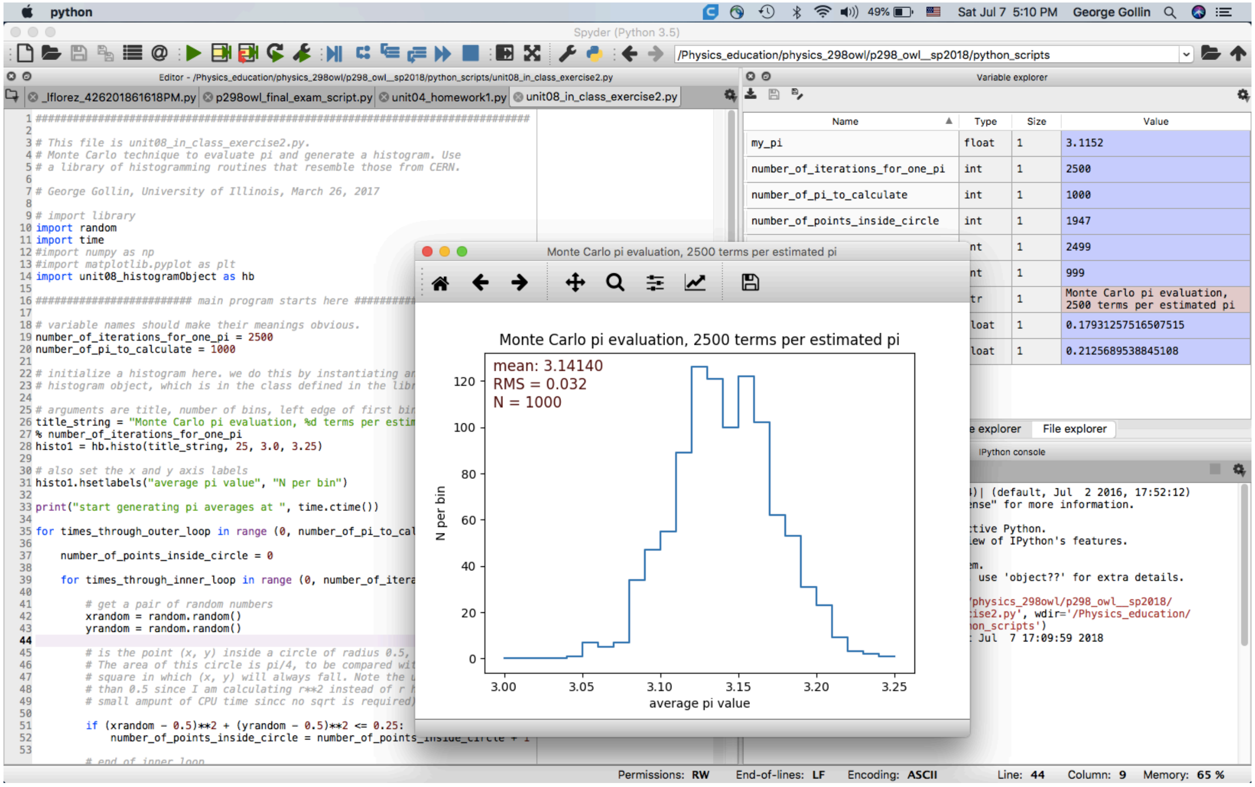
I printed the case for my prototype on an Ultimaker 2+ in my office, shown below. The camera uses a fish eye lens, which is responsible for the distortions in the image.



Ultimaker 2+ 3D printer

Data analysis

I strongly encourage you to install Anaconda's iPython spyder IDE for your data analysis work. Many of you are already familiar with python, and with spyder. Here's a screenshot.


python
Sat Jul 7 5:10 PM George Gollin
Spyder (Python 3.5)
/Physics_education/physics_298owl/p298_owl__sp2018/python_scripts
Editor - /Physics_education/physics_298owl/p298_owl__sp2018/python_scripts/unit08_in_class_exercise2.py
_lflorez_426201861618PM.py
p298owl_final_exam_script.py
unit04_homework1.py
unit08_in_class_exercise2.py
Variable explorer
Monte Carlo pi evaluation, 2500 terms per estimated pi
Monte Carlo pi evaluation, 2500 terms per estimated pi
mean: 3.14140
RMS = 0.032
N = 1000
N per bin
average pi value
0
20
40
60
80
100
120
3.00
3.05
3.10
3.15
3.20
3.25
e explorer
File explorer
IPython console
Permissions: RW
End-of-lines: LF
Encoding: ASCII
Line: 44
Column: 9
Memory: 65 %

```
################################################################################

# This file is unit08_in_class_exercise2.py.
# Monte Carlo technique to evaluate pi and generate a histogram. Use
# a library of histogramming routines that resemble those from CERN.

# George Gollin, University of Illinois, March 26, 2017

# import library
import random
import time
#import numpy as np
#import matplotlib.pyplot as plt
import unit08_histogramObject as hb

######################### main program starts here #########

# variable names should make their meanings obvious.
number_of_iterations_for_one_pi = 2500
number_of_pi_to_calculate = 1000

# initialize a histogram here. we do this by instantiating ar
# histogram object, which is in the class defined in the libr

# arguments are title, number of bins, left edge of first bir
title_string = "Monte Carlo pi evaluation, %d terms per estim
% number_of_iterations_for_one_pi
histo1 = hb.histo(title_string, 25, 3.0, 3.25)

# also set the x and y axis labels
histo1.hsetlabels("average pi value", "N per bin")

print("start generating pi averages at ", time.ctime())

for times_through_outer_loop in range (0, number_of_pi_to_cal

    number_of_points_inside_circle = 0

    for times_through_inner_loop in range (0, number_of_itera

        # get a pair of random numbers
        xrandom = random.random()
        yrandom = random.random()

        # is the point (x, y) inside a circle of radius 0.5,
        # The area of this circle is pi/4, to be compared wit
        # square in which (x, y) will always fall. Note the u
        # than 0.5 since I am calculating r**2 instead of r h
        # small ampunt of CPU time sincc no sqrt is required)

        if (xrandom - 0.5)**2 + (yrandom - 0.5)**2 <= 0.25:
            number_of_points_inside_circle = number_of_points_inside_circle + 1

        # end of inner loop
```

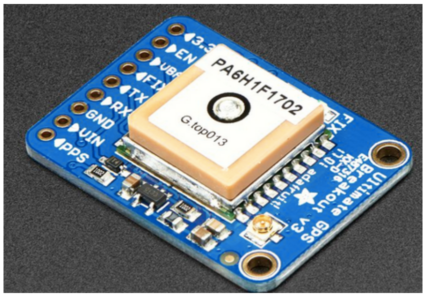
Name	Type	Size	Value
my_pi	float	1	3.1152
number_of_iterations_for_one_pi	int	1	2500
number_of_pi_to_calculate	int	1	1000
number_of_points_inside_circle	int	1	1947
	nt	1	2499
	nt	1	999
	tr	1	Monte Carlo pi evaluation, 2500 terms per estimated pi
	loat	1	0.17931257516507515
	loat	1	0.2125689538845108

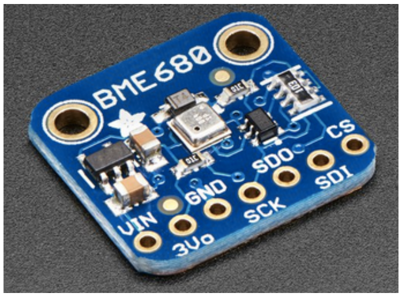
```
4)| (default, Jul  2 2016, 17:52:12)
ense" for more information.

ctive Python.
iew of IPython's features.

em.
 use 'object??' for extra details.

'physics_298owl/p298_owl__sp2018/
cise2.py', wdir='/Physics_education/
hon_scripts')
: Jul  7 17:09:59 2018
```

Sensors and other hardware

Most of the sensors we'll use have been assembled onto small "breakout boards" by Adafruit Industries. We'll solder "pin headers" onto them and plug these into the breadboard you'll use when developing your prototype.

Here are photos of a few of the sensors, taken from the Adafruit web site. In order, left to right: GPS, amplified microphone, temperature-pressure-humidity-atmospheric volatile organic compound level. They are tiny, typically less than a square inch in size.





Here's a full list of sensors and other goodies that we have on hand.

I have enough of the following for all of your breadboard and PCB devices:

- Arduino Mega 2560
- BME680 temperature, humidity, pressure, atmospheric volatile organic compounds
- ADXL326 accelerometer
- electret microphone with amplifier
- MLX90614 IR sensor
- MCP4725 DAC, 12-bit, I2C
- ultimate GPS breakout board
- PAM8302 Audio Amplifier
- DS3231 real time clock
- INA219 battery voltage/current sensor
- 3 x 4 keypad
- AA battery case
- AAA battery case
- MicroSD card breakout
- SD/MicroSD Memory Card (8 GB SDHC)
- Mini Metal Speaker w/ Wires - 8 ohm 0.5W
- 16 x 2 LCD
- AA, AAA, and CR1220 batteries

I only have ten of the following but can order more if they're popular:

- L3GD20H Triple-Axis Gyro Breakout
- Analog UV sensor
- SMA to uFL/u.FL/IPX/IPEX RF Adapter Cable for GPS antenna
- GPS Antenna - External Active Antenna
- TSL2561 Digital Luminosity/Lux/Light sensor
- VL53L0X Time of Flight Distance Sensor

Tools we'll loan to each of you:

- wire stripper
- tweezers
- eye protection
- breadboard
- digital multimeter

Tools/devices/material we'll share:

- soldering stations, solder, solder wick, "solder suckers"
- spools of brightly colored 22 gauge wire
- basic toolset (pliers, screwdrivers, etc.)

- Ultimaker 2+ printer
- PLA plastic filament, in various colors

So it's a pretty good collection of stuff.

Some of what you'll learn

You will learn to identify a measurement that you could make, in hopes of understanding more about, and perhaps improving, the State of Things.

You will learn to construct and build a device that might allow you to make those measurements.

You will learn to test and calibrate your device so that your data bear an understandable relationship to the physical parameters you are studying.

You will learn to do fieldwork, so that you come back from the world with valid data that can be analyzed and interpreted.

You will learn to report your results and speak of them to an audience of skeptics, supporting your conclusions with irrefutable facts. You will be prepared to argue convincingly when a dishonest charlatan claims to possess "alternative facts" without evidentiary backing, and insists that these are as valid as your observations.

You will also have a blast building your gizmo.

The rhythm of things

Time-on-task is an important part of mastering the tools you will use this semester. Rather than staging the various tasks for completing your project sequentially, you'll work with many of them in parallel. This will give you more time to digest the fine points of working with the tools we'll employ.

Speaking loosely, these are the things you'll do:

- write (or adapt) code to talk to sensors
- create a schematic diagram representing your device
- design a printed circuit board (this is optional)
- load and test a PCB
- 3D-print a case
- write a data acquisition program
- figure out how to take data and verify its validity

- do your field work
- write code to analyze your data
- report your results.

The computer-based tool set comprises the following:

- Arduino programing IDE
- EAGLE schematic capture
- EAGLE PCB layout
- TinkerCad 3D printing design tool
- Cura 3D printing renderer
- Anaconda iPython programming IDE.

Each week you'll advance the design of your data logger, write Arduino code to communicate with your sensors, further develop your plans for the field work necessary for your project (including the structure of data acquisition code), and address physical infrastructure matters like case construction and PCB fabrication.

After the first week we'll have brief reports to the class from a few of the teams. Topics (which I will assign) might include how a particular sensor works, what an interrupt does, how the I2C data transfer protocol works, and so forth. A report should last at most ten minutes, be carried in at most ten PowerPoint slides, be presented to the class by all the team members, and be suitable for upload to the course web site. (That means proper attribution of sources, and so forth.)

Every week (preferably Wednesday or Thursday), each team will meet with me for 20 minutes to discuss progress, problems, clever ideas, and any other issues that might arise.

From time to time we'll have experts speak to us about subjects relevant to your efforts. For example, Professor Eric Benson (Chair of Graphic Design in UIUC's School of Art and Design) will discuss design issues before you begin to sketch a case for your device.

Homework, exams, grading, and your obligations

The homework will consist of moving your design forward as far (and as fast) as you can. I expect you will spend about six hours at this outside of class every week. You should work with the other members of your team as much as possible, sharing code and design tips as convenient. You should document your progress, your plans, your brilliant realizations, your frustrations, and your concerns in your electronic diary.

You are to upload a PDF version of your entire cumulative electronic diary to the course directory every week. Please put this into a single file; when you close one volume and open another, please upload a file that includes ***all previous volumes*** as well as the present one. Uploads are due at noon on Fridays, beginning with the second week. You should indicate the time/date for the material you enter, as well as with whom you've been collaborating.

You and your team will meet with me once per week for 20 minutes. All members of your team must attend, and must arrive promptly at the scheduled time, without exception.

You and your team members will give several reports to the class over the course of the semester. The reports are to be clearly written PowerPoint presentations (with proper attribution of sources) aimed at your audience of fellow students, who will not necessarily know what you mean by, for example, "I2C interface."

You must come to class on time, arriving with your laptop and power adapter, goodie box of parts and tools, and breadboard. When you have a working, finished, PCB version of your device you should bring both the breadboarded and PCB versions of your device.

There will be no midterm exams.

In place of a final exam, I will require you and members of your team to generate three documents, with all team members as coauthors:

- A 20 minute PowerPoint presentation describing your project: the measurement(s) you've made, and the reasons for so doing; the hardware and software you've built to perform these measurements; your fieldwork and calibrations; your analysis and conclusions. Your intended audience comprises your Physics 398DLP classmates. Presentations are due at the start of class, week 13; you will present during the 13th or 14th week of the term.
- A written report of approximately five pages, single spaced, describing your measurements and conclusions; this is essentially the same information that you will write into your PowerPoint presentation. However, your report should be aimed at an audience that is outside the university community. For example, if you've measured the anomalous transverse accelerations of Amtrak trains, your audience might be the Illinois Department of Transportation. Your report is due in class on week 14.
- A cover letter suitable for sending with your written report to, for example, the Illinois Department of Transportation. You will request a face-to-face meeting to describe your results. Your letter is due in class on week 14.

I would hope to mail your report and cover letter to the intended recipients at the end of the semester. We will also post these to the course web site.

The grading will be similar to Physics 298owl: if you work hard, are clever, come to all class activities, and do a good job on your reports you will receive at least an A. But if you miss class, miss conferences with me, are late uploading your diary files, or do not do a good job on your reports, I will hammer you.

If you miss class for a legitimate reason, you must submit documentation to Kate Shunk in the Undergraduate Physics Office or upload it through the excused absence portal linked to the course web site. I will expect you to make up the work you didn't accomplish.

Your obligations include working in a safe manner: always wearing eye protection when soldering, always washing your hands soon after handling metallic objects such as header pins or solder.

Course calendar

Note: the list of guest speakers is speculative. I have only spoken to a few of them.

what	day	date	comments
semester begins	Monday	August 27	
week 1	Friday	August 31	
week 2	Friday	September 7	Eric Benson: design considerations
week 3	Friday	September 14	Celia Eliot: communicating clearly
week 4	Friday	September 21	Bill O'Brien: sound and acoustics
week 5	Friday	September 28	Scott Willenbrock: sustainability
week 6	Friday	October 5	
week 7	Friday	October 12	
week 8	Friday	October 19	XYZ: Social innovation
week 9	Friday	October 26	
week 10	Friday	November 2	
week 11	Friday	November 9	
week 12	Friday	November 16	
Thanksgiving break	Friday	November 23	
week 13	Friday	November 30	PPT project report due
week 14	Friday	December 7	written report and cover letter due

end of term conferences	Monday - Wednesday	December 10 - 12	
reading day	Thursday	December 13	

Detailed syllabus

Note that we have about five weeks of slack time in the following schedule.

Week 1, in class

- Form up into research teams of two to four people and begin discussing which project you'll pursue. Schedule a time for the team's weekly meeting with GG.
- Sign out the tools and hardware needed to build your devices.
- Install the Arduino programming IDE.
- Register an account with Autodesk, then install the EAGLE schematic capture/PCB IDE
- Log in to TinkerCad and make sure it recognizes your Autodesk account. Find a simple design for something amusing on the TinkerCad site and export it to an STL file. (Hands-on demo by GG. You will take notes as you follow along.)
- Download Cura 3.4. Have Cura generate a gcode file from your STL file.
- Plug the Arduino into a USB port on your laptop, find a demonstration program that will blink its LED, upload the program and confirm that the LED really does blink. (Hands-on demo by GG if desired.)
- Modify the blinking LED program to flash your first and last initials in Morse code.
- Get a soldering lesson from Todd Moore, an electrical engineer who is staffing the undergraduate physics program. Solder pins onto the bottoms of some of your breakout boards, including the ADXL326 accelerometer and BME680 temperature, etc. sensor.
- Install the power terminals and plastic feet onto your breadboard and duct-tape your Arduino to the surface of your breadboard, but not on top of any of the interconnect holes.
- Install a BME680 on your breadboard, following the instructions on the Adafruit site, and download the demonstration software to communicate with it. Make it report what it sees in a serial monitor window. (Hands-on demo by GG if desired.)
- Do the same with the ADXL326 accelerometer.
- Select volunteers for the following reports: (1) introduction to the Arduino Mega 2560; (2) how the BME680 T/P/H/VOC sensor works; (3) how a successive approximation ADC works

Post-class assignment

- Formulate a plan of action with the members of your team. I want all of you to be involved with all flavors of activity: writing Arduino code, generating schematics to represent your devices, and so forth. But it is fine for one person to take the lead on, say, managing the code that interrogates the GPS package.
- Finish whatever installations didn't go smoothly during today's class.

- Install other sensors on your breadboard circuits, download demonstration software to communicate with them, and see that the new code runs properly.

Week 2, conference with the professor

- Discuss your project plan, including who you must contact for permission to, for example, install atmospheric methane detectors in the UIUC barns.

Week 2, in class

- Reports (1), (2), (3) that were assigned last week. (We'll scatter these throughout the four-hour class meeting, rather then hearing them sequentially.)
- Select volunteers for the following reports: (1) how the ADXL326 accelerometer works; (2) how an electret microphone works; (3) how the MLX90614 IR sensor works.
- Open a project in Eagle schematic capture/PCB and create a schematic that represents your breadboarded prototype as it currently exists, and reflects what you intend to do with it today. I will help with this, and show you how to work with the libraries of parts that you'll need. (Hands-on demo by GG, for sure: placing parts, connecting stuff, naming nets.)
- Install the microSD card reader and more sensors on your breadboard, making sure to leave room for the AA battery pack.
- Download software to communicate with your new sensors and test that they respond properly.
- Soldering clinic with Todd Moore.
- Class visitor: Professor Eric Benson (Art + Design). *Thinking about design.*

Post-class assignment

- Using TinkerCad, create a 3D-printable design for a smallish box with a lid in which you'll keep your various breakout boards. It's fine to borrow a design you find on the web, but figure out how to engrave your name onto the lid of the box. Please don't make your box larger than about 2.5 cm × 5 cm × 2.5 cm. I'll print it for you (you can pick the color!) once you've used Cura to create a gcode file.
- Finish adding sensors to your breadboard prototype, downloading code from the web. Keep your schematic diagram up to date.

Week 3, conference with the professor

- Keep me apprised of your progress, and how you have decided to share the responsibilities for the various tasks needed to advance your project.
- Tell me how well it worked to manipulate objects with TinkerCad.
- Pick a PLA filament color for your group's TinkerCad projects and leave with me copies of your gcode files.

Week 3, in class

- Reports (1), (2), (3) that were assigned last week.
- Select volunteers for the following reports: (1) I2C communication protocol; (2) interrupts; (3) how the MCP4725 DAC works.
- TinkerCad tutorial with GG: grouping and ungrouping objects, aligning objects, making holes, rotating your point of view, using build planes.
- Whole class group discussion: how is it going? What's too hard/too easy? What are your thoughts about the field work you'll be doing in a few weeks? What might your data acquisition code look like? What kind of technical support might I provide to make your work go more smoothly? Have you made contact yet with anyone to learn the rules concerning entering into their environment to make measurements?
- Block out a data acquisition program (DAQ) that you'll use for your field work and measurement; begin writing a skeleton version of it. Do this collaboratively.
- Install Anaconda's iPython IDE and begin building an offline data analysis package.
- Class visitor: Ms. Celia Elliott (Physics). *Communicating clearly.*

Post-class assignment

- Continue writing code for your device's DAQ, and for your offline analysis. Do this in collaboration with other members of your group. Your DAQ must be able to write data to your device's microSD card.
- Using TinkerCad, start designing the case for your device. (I hope to have a PCB design ready for you by this time.)
- Develop with your group a detailed plan for field testing your (breadboarded) devices.
- Run a quick field test. For example, if you plan to study Amtrak trains, ride an MTD bus for a half hour.

Week 4, conference with the professor

- Keep me apprised of your DAQ and offline analysis progress.
- Tell me how your TinkerCad case is progressing.
- Describe how your quick field trials went, and whether you were able to read/process the data on the microSD card.

Week 4, in class

- Reports (1), (2), (3) that were assigned last week.
- Select volunteers for the following reports: (1) Adafruit GPS breakout board; (2) Atmel 2560 microcontroller timer modules; (3) PWM (pulse width modulation) and the PAM8302 amplifier.
- TinkerCad assistance from GG and teaching assistant as needed.
- DAQ development.
- Offline analysis code development.

- Class visitor: I will ask Professor Bill O’Brien (ECE). *Sound and acoustics.*

Post-class assignment

- Continue your DAQ and offline analysis code development.
- Finish the TinkerCad case design.
- Run another quick field test if necessary.
- Analyze your quick field test data and draw tentative conclusions about your project’s feasibility. Consider adjusting your plans based on your findings.

Week 5, conference with the professor

- Discuss your quick field trial analyses, conclusions, and proposed adjustments.

Week 5, in class

- Reports (1), (2), (3) that were assigned last week.
- Select volunteers for the following reports: (1) L3GD20H Triple-Axis Gyro; (2) VL53L0X Time of Flight Distance Sensor; (3) high speed microSD data transfer issues
- Soldering Clinic II with Todd Moore: solder headers onto printed circuit boards, mount all necessary components onto the PCB.
- Comparison tests: breadboard vs. PCB device.
- Class visitor: I will ask Professor Scott Willenbrock: (Physics). *Sustainability.*

Post-class assignment

- Continue your DAQ and offline analysis code development.
- Finish the TinkerCad case design, if it is still in progress.
- Prepare a ten minute status report to present to the class about your field tests.

Week 6, conference with the professor

- Discuss your status report and online/offline code progress.

Week 6, in class

We will meet at MakerLab, Business Instructional Facility, 515 E. Gregory, room 3030.

- Clean, calibrate, and tune MakerLab’s Ultimaker printers, including installing windshields.
- Start printing data logger cases.
- Continue class meeting at BIF, with groups presenting their status reports.

Post-class assignment

- Pick up your case from MakerLab; contact me if the print failed or exhibited adhesion (or other) problems.
- Finish your DAQ and offline software.
- Finalize your plans for field work.

Week 7, conference with the professor

- Discuss your readiness to do field work (software is working, PCB device is working, etc.).
- Discuss how you will calibrate/monitor your device.
- Discuss your field work plans.

Week 7, in class

Field work, so we won't meet in Loomis.

- Disperse, make measurements, email me to let me know how it's going.

Post-class assignment

- Analyze your data, including accounting for calibration issues.

Week 8, conference with the professor

- How'd it go? What did you find? Were there surprises?
- Discuss any possible follow-up measurements that might be necessary.

Week 8, in class

- Full-class discussion of how it went, what you found, problems you encountered, and so forth.
- Further coding to refine offline analysis and tune DAQ.
- Class visitor: Professor XYZ (Academy for Entrepreneurial Leadership). *Social innovation.*

Post-class assignment

- Prepare your reports: 20 minute PPT to present in class; 5 page paper for external distribution; cover letter.

Week 9, conference with the professor

- Discuss your reports with me.

Week 9, in class

- All groups present their results, with full-class discussions after each group's presentation.

Post-class assignment

- Adjust/tune/repair your reports so that they can be released publicly.

Week 10, conference with the professor

- Wrap up discussions

Week 10, in class

Meet at GG's house for a…

- Serious pizza party