

Physics 398DLP
Design Like a Physicist
Fall 2018

George Gollin
University of Illinois at Urbana-Champaign

Unit 1: Organizations, Distributions, and Installations

Unit 1: Organizations, Distributions, and Installations

Goals this unit	3
Introduction: project physics.....	4
Prerequisites.....	5
We are not building robots.....	5
Some possible projects.....	6
The style in which we will work.....	6
Groups, projects, electronic diaries.....	7
Distribution of stuff!	8
Install the Arduino programming IDE.....	9
Register a user account with Autodesk and download EAGLE	11
Visit the TinkerCad web site.....	11
Download and install the most recent version of Cura.....	13
It's time to build Version Zero of your data logging device.....	14
Accelerometer online!.....	15
Temperature, relative humidity, etc.....	15
Volunteers needed.....	15
Post-class assignment.....	16

Goals this unit

- 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 and 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.

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 in the elderly
- (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 2980w1 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 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 des 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.

Groups, projects, electronic diaries

Please form up into research/project groups, discuss which project you might prefer to undertake, and find a time that all members of your group can meet with me for 20 minutes. My preference is sometime on Wednesday (10 am – 4 pm) or Thursday (10 am – 1 pm; 2 pm – 4 pm).

Open up your diary file, and add to your account of the afternoon’s activities as you work.

Distribution of stuff!

Here's what I have for you. Once you have everything loaded into your plastic storage box, please have one of your partners check your inventory.

part	description	quantity	status
Sterilite plastic box	your toy box!	1	on loan
22 gauge solid wire, multiple colors	for breadboarding	several feet	yours to keep
wire strippers	tool	1	on loan
tweezers (#422)	tool	1	on loan
eye protection	safety	1	on loan
breadboard	prototyping	1	on loan
digital multimeter	tool	1	on loan
SD/MicroSD 8 GB Memory Card	memory	2	one is on loan
10 k Ω 1/4 W resistor (#2784), 25 per package		2	one is on loan
3 x 4 keypad (#1824)	4 x 3 keypad	2	one is on loan
470 Ω 1/4 W resistor (#2780), 25 per package		2	one is on loan
ADXL326 accelerometer (#1018)	3 axis	2	one is on loan
AA battery case (#3456)	5 x AA	2	one is on loan
BME 680 T/RH/P/VOC (#3660)	temp, etc.	2	one is on loan
breadboard trim potentiometer 10 k Ω (#356)	set LCD contrast	2	one is on loan
CR1220 backup batteries (#380)	for RTC & GPS	4	two are on loan
DS3231 real time clock (#3013)	clock	2	one is on loan
electret microphone with amplifier (#1063)	rock n roll	2	one is on loan
INA219 battery voltage/current sensor (#904)	battery monitor	2	one is on loan
MCP4725 DAC, 12-bit, I2C (#935)	everyone needs a DAC	2	one is on loan
MicroSD card breakout (#254)	memory breakout	2	one is on loan
Mini Metal Speaker w/ Wires - 8 ohm	0.5 W	2	one is on loan
MLX90614 IR sensor (#1748)	MLX90614	2	one is on loan
On-Off-On-Off Alternating Pushbutton (#1684)	battery/USB power	1	yours to keep
PAM8302 Audio Amplifier (#2130)	audio	2	one is on loan
ultimate GPS breakout board (#746)	NMEA 86	2	one is on loan
USB Cable - Standard A-B - 3 ft/1m (#62)	laptop - Arduino	1	yours to keep
AA batteries, 100-pack	5 needed per device	10	five are on loan
16 x 2 LCD CFAH1602B-NGG-JTV	display	2	one is on loan
0.1uF capacitors	for 5V bypass	20	
Arduino Mega 2560	microcontroller	2	one is on loan

Install the Arduino programming IDE

Go to the Arduino website <https://www.arduino.cc/> and navigate to <https://www.arduino.cc/en/Guide/ArduinoMega2560>. Download and install the Arduino Desktop IDE (Integrated Development Environment) on your laptop. Connect an Arduino to a USB port in your laptop. The Arduino probably comes with a blink-an-LED program preloaded, so a yellow LED near the USB connector might start blinking as soon as the board is powered.

You'll need to go to the Tools → Port menu to select which communication channel your laptop will use to talk to the Arduino. While you're at it, open a serial monitor window by following Tools → Serial Monitor. The Arduino will write information to this window as instructed by the program it is running.

Please create a folder in which you will store your various Arduino programs. (For some reason people call an Arduino program a sketch. I think that sounds silly.)

You can find Arduino tutorials and sample programs at <https://www.arduino.cc/en/Tutorial/BuiltInExamples>. See also File → Examples → 01.Basics → Blink for a ready-to-run program, which (after a few modifications) looks like this:

```

/*
  Blink: turns an LED on and off repeatedly.
  http://www.arduino.cc/en/Tutorial/Blink

  An Arduino Mega 2560 has an LED attached to digital pin 13.
  Technical Specs of your board Arduino can be found at:
  https://www.arduino.cc/en/Main/Products

  Authors: Scott Fitzgerald, Arturo Guadalupi, Colby Newman
*/

// global variables and constants go here. I'll explain the use of const
// in class.

// length of delay before changing LED state, in milliseconds
const int the_delay = 1000;

// the setup function runs once when you press reset or power the board

void setup() {
  // initialize digital pin LED_BUILTIN as an output.
  pinMode(LED_BUILTIN, OUTPUT);
}

// the loop function runs over and over again forever

void loop() {
  // turn the LED on (HIGH is the voltage level, predefined by the compiler)
  digitalWrite(LED_BUILTIN, HIGH);

  // wait for one second (1,000 milliseconds)
  delay(the_delay);

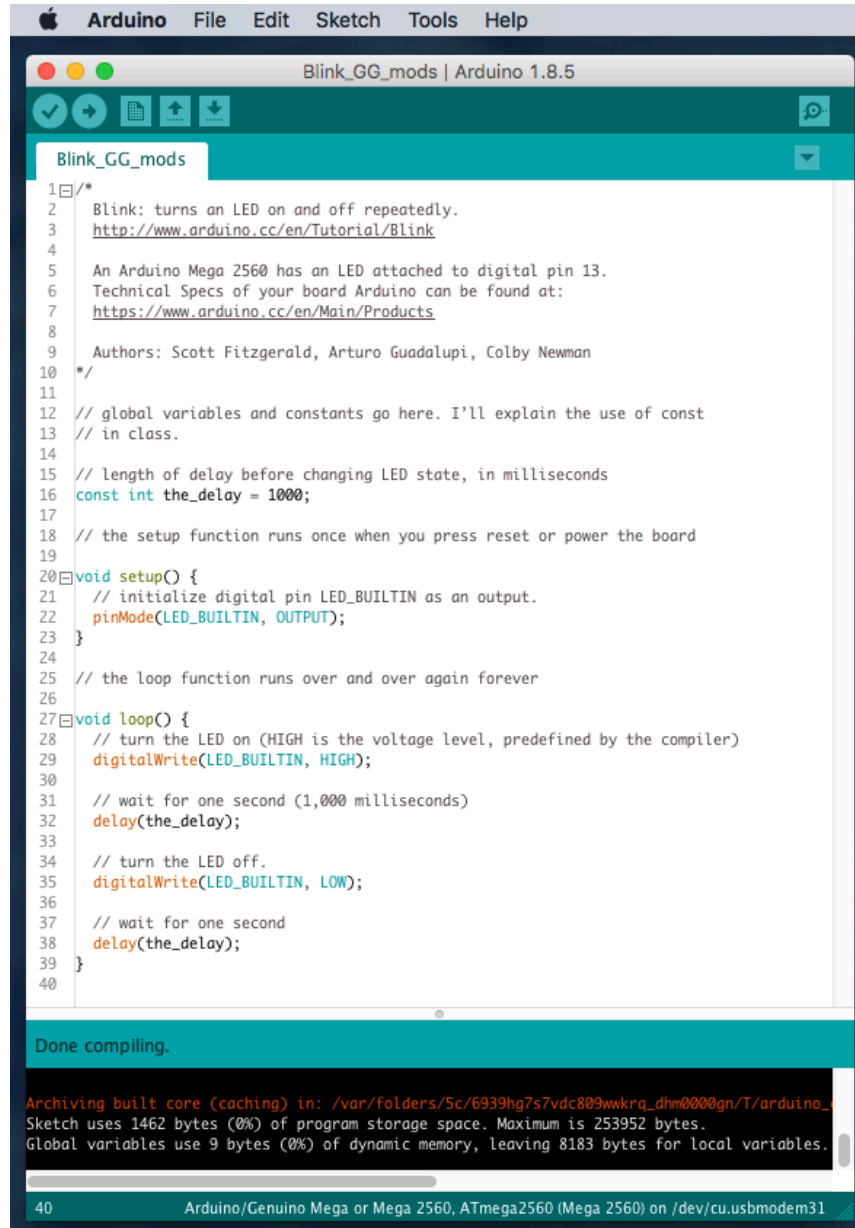
  // turn the LED off.
  digitalWrite(LED_BUILTIN, LOW);

  // wait for one second

```

```
    delay(the_delay);  
}
```

Here's how it looks in the IDE's editor:



The screenshot shows the Arduino IDE interface. The title bar reads "Blink_GG_mods | Arduino 1.8.5". The menu bar includes "Arduino", "File", "Edit", "Sketch", "Tools", and "Help". The toolbar contains icons for checkmark, left arrow, right arrow, upload, and download. The main editor window shows the following code:

```
1 1/*  
2  Blink: turns an LED on and off repeatedly.  
3  http://www.arduino.cc/en/Tutorial/Blink  
4  
5  An Arduino Mega 2560 has an LED attached to digital pin 13.  
6  Technical Specs of your board Arduino can be found at:  
7  https://www.arduino.cc/en/Main/Products  
8  
9  Authors: Scott Fitzgerald, Arturo Guadalupi, Colby Newman  
10 */  
11  
12 // global variables and constants go here. I'll explain the use of const  
13 // in class.  
14  
15 // length of delay before changing LED state, in milliseconds  
16 const int the_delay = 1000;  
17  
18 // the setup function runs once when you press reset or power the board  
19  
20 void setup() {  
21   // initialize digital pin LED_BUILTIN as an output.  
22   pinMode(LED_BUILTIN, OUTPUT);  
23 }  
24  
25 // the loop function runs over and over again forever  
26  
27 void loop() {  
28   // turn the LED on (HIGH is the voltage level, predefined by the compiler)  
29   digitalWrite(LED_BUILTIN, HIGH);  
30  
31   // wait for one second (1,000 milliseconds)  
32   delay(the_delay);  
33  
34   // turn the LED off.  
35   digitalWrite(LED_BUILTIN, LOW);  
36  
37   // wait for one second  
38   delay(the_delay);  
39 }  
40
```

Below the code editor, a status bar indicates "Done compiling." and provides memory usage information: "Archiving built core (caching) in: /var/folders/5c/6939hg7s7vdc809wvkra_dhm0000gn/T/arduino_... Sketch uses 1462 bytes (0%) of program storage space. Maximum is 253952 bytes. Global variables use 9 bytes (0%) of dynamic memory, leaving 8183 bytes for local variables." The bottom status bar shows "40 Arduino/Genuino Mega or Mega 2560, ATmega2560 (Mega 2560) on /dev/cu.usbmodem31".

Click on the check mark to compile the program; click on the right arrow button to compile it and download the executable to the Arduino.

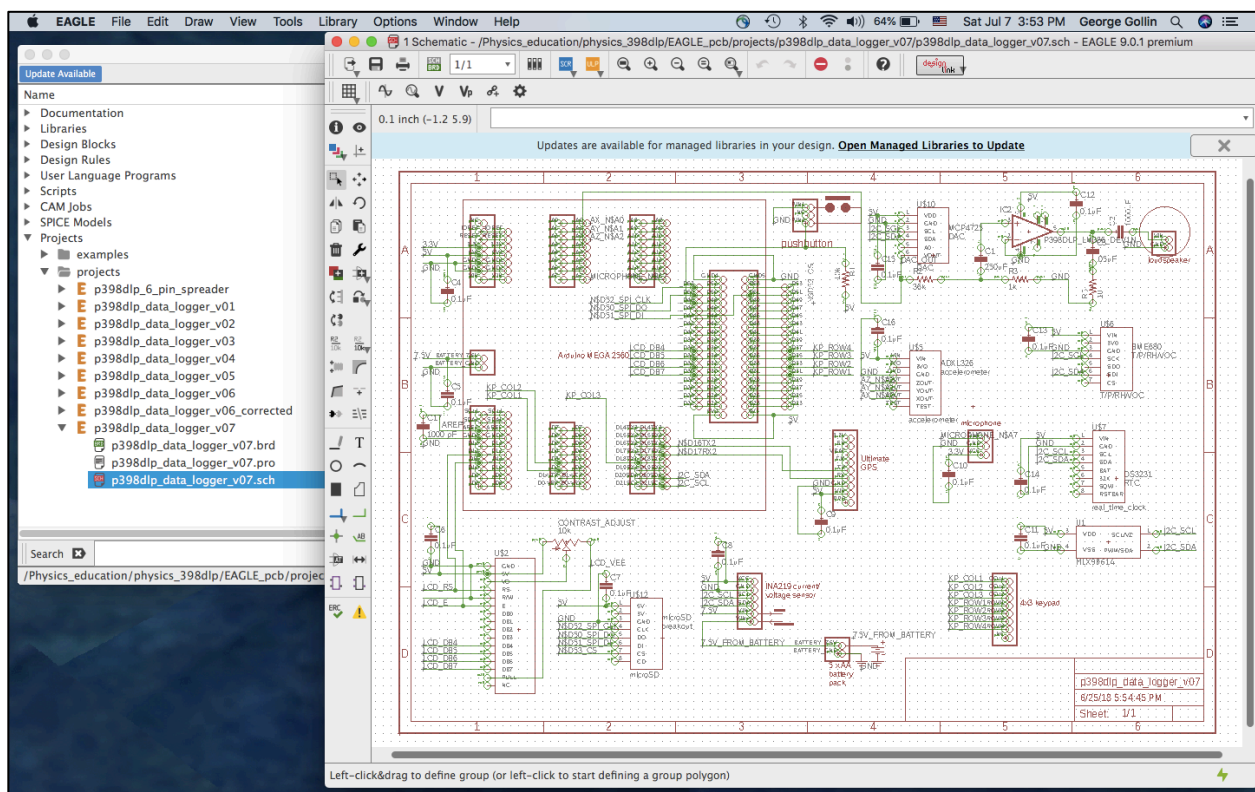
Please open the Blink example, modify it as you are inclined, then save it to the folder you've created to hold your programs. Compile and download it to confirm that it works.

Register a user account with Autodesk and download EAGLE

Go to

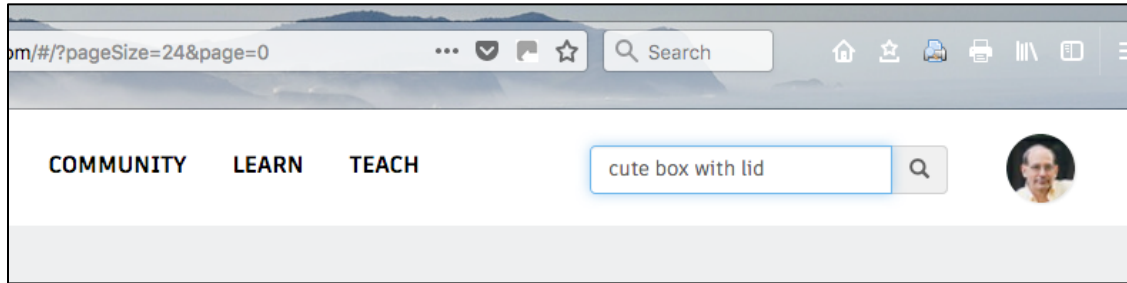
<https://knowledge.autodesk.com/customer-service/account-management/education-program/create-education-account/create-account-students-educators> to register an account with Autodesk as a student. You'll have much better (free!) access to Autodesk's tools this way than you would if you registered a non-student account, even if it were free.

After you've registered, visit <https://www.autodesk.com/education/free-software/eagle> and sign in. Follow the steps to download EAGLE, Autodesk's schematic capture and printed circuit board layout tool. We won't actually do anything with EAGLE until next week. But here's how my schematic for a prototype data logger looks in EAGLE.



Visit the TinkerCad web site

Go to TinkerCad.com and log in using your Autodesk username and password. Search for something interesting but small: “cute box with lid,” for example.

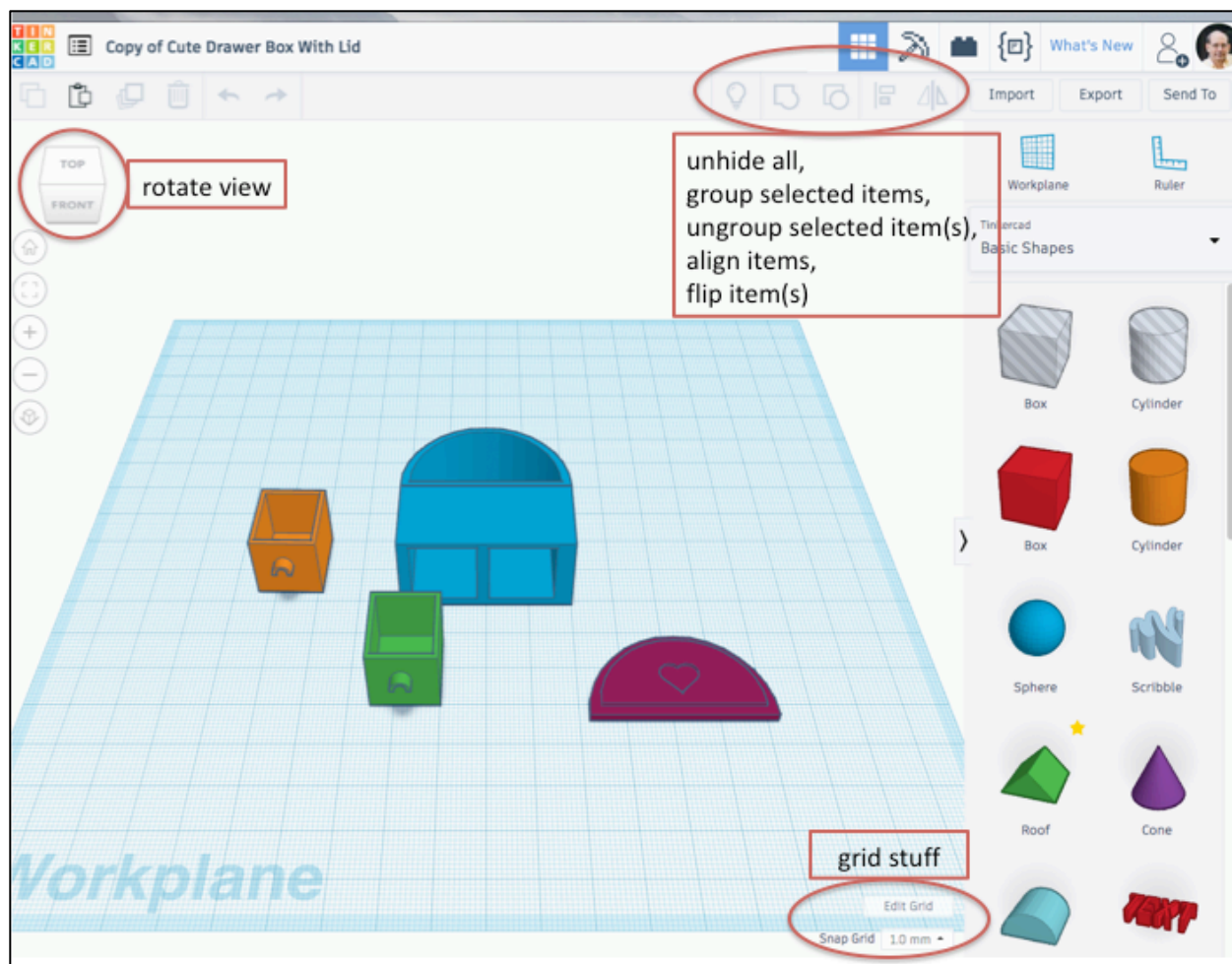


Click on the image, then select “Copy and Tinker.” Once the editor opens I’ll talk you through some of the things you can do. Make very brief notes on any tricks that you find useful, for later recall and reference.

We’ll try some of these actions:

- rotating point of view; zooming in and out
- grouping and ungrouping objects
- moving objects using the mouse and arrow keys
- rotating objects
- hiding/unhiding objects
- changing sizes while preserving (or not) aspect ratios
- making holes
- using (and changing) the grid
- aligning items
- quick example: how to make a lidless box
- adding text

After you’ve done as much nastiness to the thing you’ve been editing, make an STL (stereo lithography) file by going to the export tab near the upper right side of the screen. We’ll probably spend about 30 minutes on this today. Later, you’ll use TinkerCad to design the case for your device.



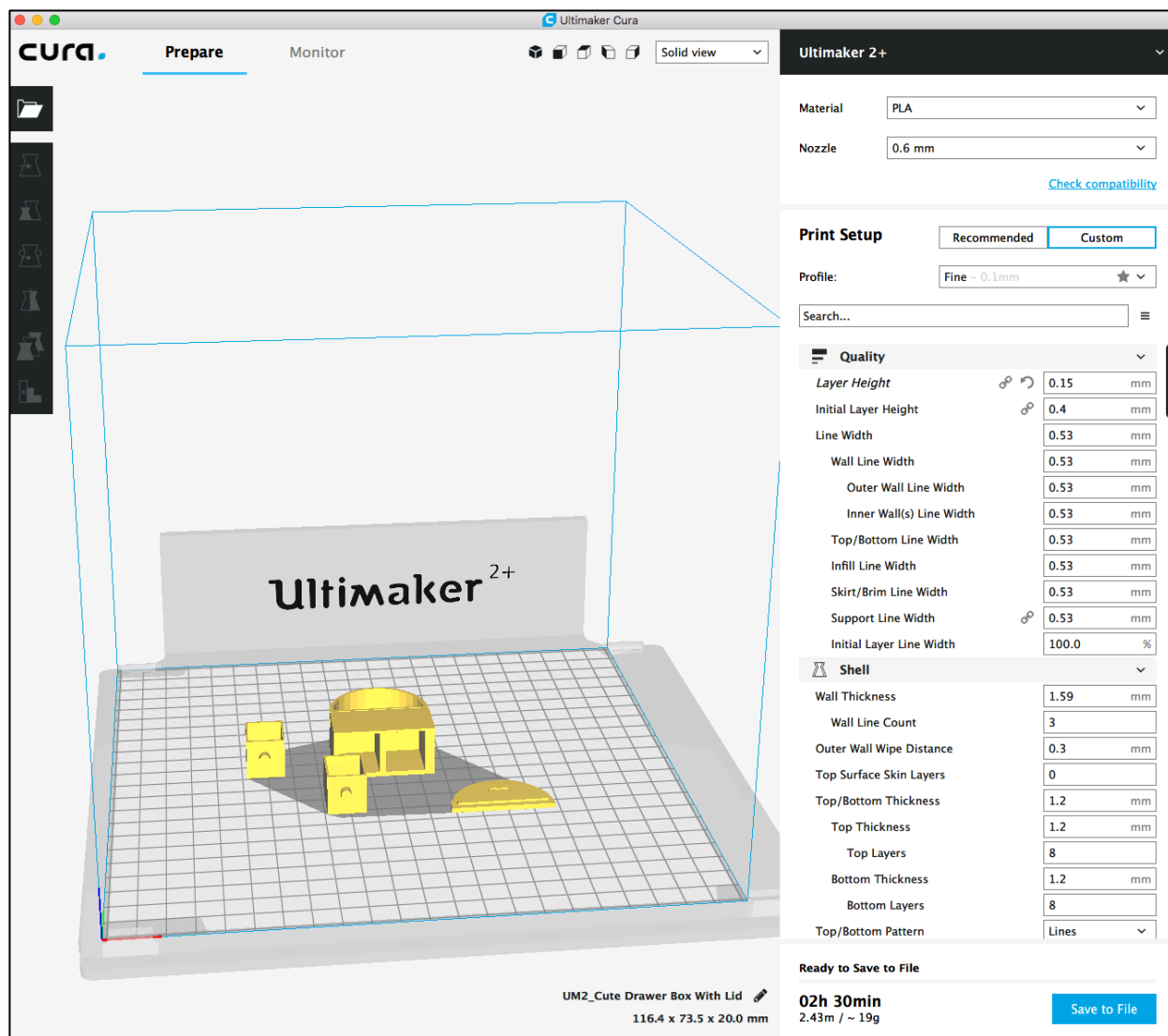
Download and install the most recent version of Cura

The 3D printer in my office, and the army of printers in the Business School's MakerLab are built by Ultimaker B.V., a Dutch company. Cura is their software product that generates gcode files (the inputs to a 3D printer) from STL files.

Download and install the latest version of Cura from Ultimaker's web site: go to <https://ultimaker.com/en/products/ultimaker-cura-software>. Open Cura and drop onto it the STL file you've created with TinkerCad.

I'll talk you through the process, along with a couple of settings you'll want to impose. Note that the program gives an estimate of how long your object will take to print.

A couple of technical issues to consider: (1) adhesion to the build surface; (2) support for bridged features. I'll explain what I mean in class.



It's time to build Version Zero of your data logging device.

Let's prep our breadboards. Please attach the connectors and rubber feet to your breadboard. Please fasten an Arduino, still in its plastic carrier, to your breadboard. I recommend duct tape (the baby sitter's friend!); position the device so that it doesn't cover any of the breadboard's plastic structures that are used to hold components.

Todd Moore, an electrical engineer who used to build complex devices for the High Energy Physics Group but now staffs the undergraduate physics program, will run a soldering clinic in class today. Please have Todd teach you how to solder! You'll attach pin headers to a few of the sensor breakout boards in your collection of goodies. I'd suggest you start with one of your ADXL326 accelerometer and BME680 T/P/RH//VOC boards.

Accelerometer online!

Most of our breakout boards were built by Adafruit Industries, a wonderful provider of small electronic packages intended largely for the hobbyist market. Go to the Adafruit site <https://www.adafruit.com/> and find the ADXL326 page that mentions some of the supporting infrastructure available for you. (The URL is <https://www.adafruit.com/product/1018>.)

Install your ADXL326 breakout board onto the breadboard. Using sensible colors (red for +5V, black for ground, other colors for signal lines), connect GND to one of the Arduino's GND lines and VIN to one of the Arduino's 5V lines. Also connect the Xout, Yout, and Zout lines to the Arduino's analog inputs A0, A1, and A2. (See <https://learn.adafruit.com/adafruit-analog-accelerometer-breakouts/arduino-wiring>.)

Download from Adafruit a sample program that communicates with the accelerometer. (See <https://learn.adafruit.com/adafruit-analog-accelerometer-breakouts/downloads> or <https://learn.adafruit.com/pages/747/elements/1791783/download>.) Compile the code, download it to the Arduino, and run it!

Temperature, relative humidity, etc.

Install the BME680 onto your breadboard. See <https://www.adafruit.com/product/3660> and links therein. You should power it using the Arduino's +5V and GND lines. We'll let the device and the Arduino communicate using an I2C (I Two C) interface; set this up by connecting the BME680's SCK (serial clock) pin to the Arduino's SCL output (pin 21). Also connect the BME680's SDI (serial data) to the Arduino's SDA input (pin 20). You should leave unconnected the BME680's 3Vo, SDO, and CS pins.

You'll need to install one of Adafruit's libraries to drive the BME680. See <https://learn.adafruit.com/adafruit-bme680-humidity-temperature-barometric-pressure-voc-gas/arduino-wiring-test> and scroll down to the section titled "Install Adafruit_BME680 library." Follow the directions to install the library and upload to the Arduino the demonstration software.

Get it to run! You should find that the pressure transducer is so sensitive that it can tell that you've lifted the board up from your worktable by a couple of feet just from the change in atmospheric pressure.

Volunteers needed

Here are the topics for presentation next week:

- An introduction to the Arduino Mega 2560
- How the BME680 T/P/H/VOC sensor works
- How a successive approximation ADC works

I'd like a report to 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.) Keep in mind that your audience is other students in the class, rather than a group of professional engineers.

Post-class assignment

1. 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. You will discuss this with me when your group and I meet next week.

2. Finish whatever installations didn't go smoothly during today's class.

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

