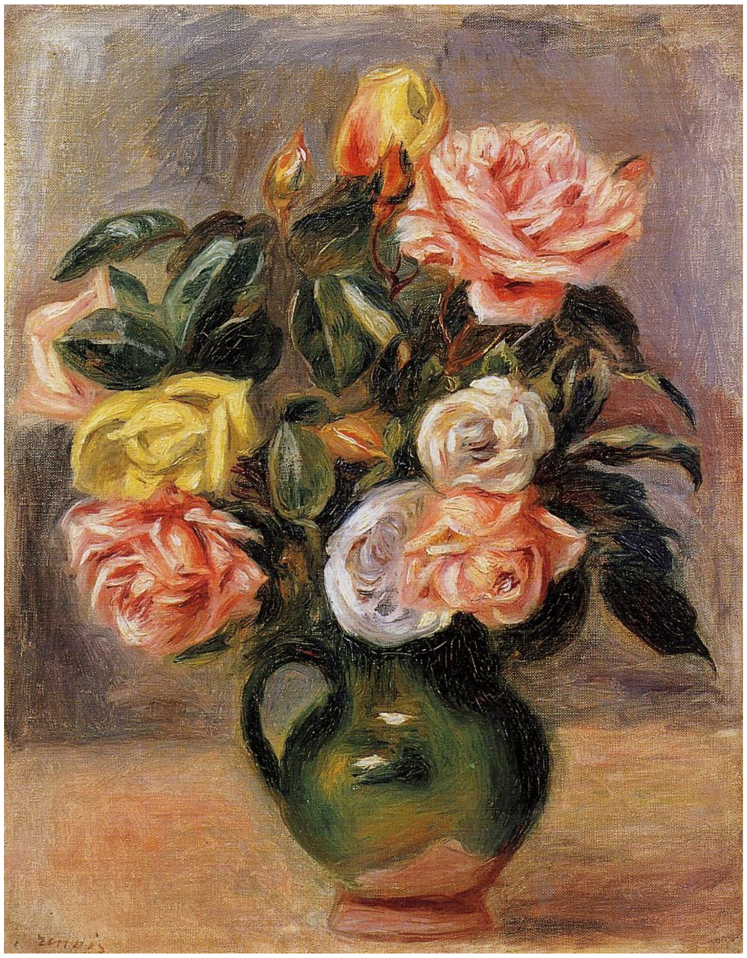


https://1.bp.blogspot.com/-XhQVLp4bA34/TzAnlNdwZHI/AAAAAAAABCbI/NCj49D2uP8w/s1600/Pierre+Auguste+Renoir+Flowers+-+Tutt%2527Art%2540+%252817%2529.jpg

Physics 524

Survey of Instrumentation and Laboratory Techniques

2023

George Gollin

University of Illinois at Urbana-Champaign

Unit 1a: Arduino C++ programming environment

Week 1: Organizations, Distributions, and Installations

Goal for this week

- You will assemble a starter circuit on a breadboard and use the Arduino Integrated Developers Environment (IDE) to program and test it.

Notebooks

Please open up your notebook, and add to your account of the afternoon's activities as you work. You'll want to keep track of your work, and your revelations, so you can return to them at a later date. We won't be looking at your notes, but if I ask you something like "where did you go to find that cool piece of demo software?" you should be able to give me an answer based on your notes.

Distribution of stuff!

I have an alarmingly large amount of stuff available for you. Some of the following I've already packed in the parts/tools kits I'll distribute in class. (You'll also use these in Physics 523.)

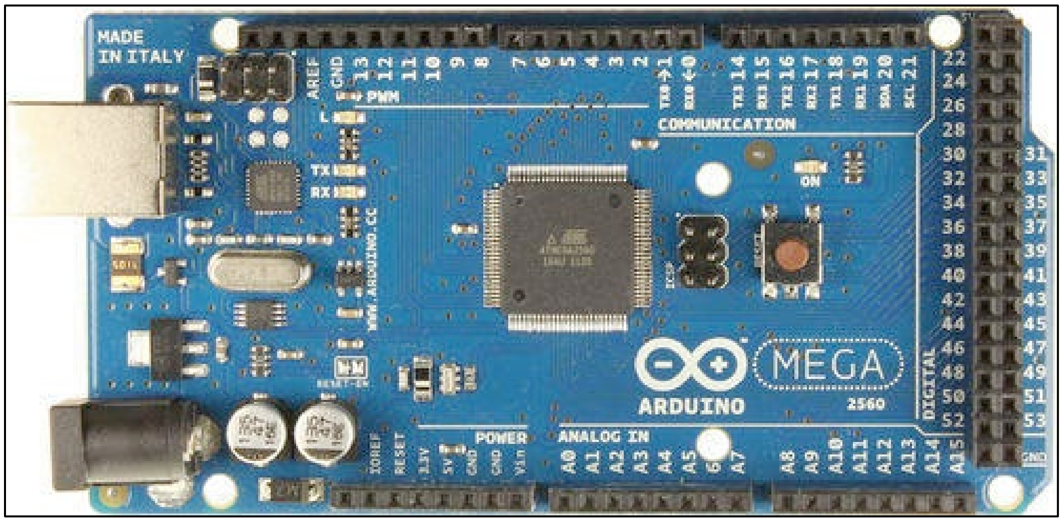
part
16 x 2 LCD CFAH1602B-NGG-JTV
2 x AAA & 1 x AAA battery holders
3 x 4 keypad (#3845)
5 x AA battery holder, Adafruit 3456
74HC137 3-8 decoder/demultiplexer (TTL)
74HC157N quad 2 input (TTL) multiplexer
Adalogger Feather M0, Adafruit 2796, and microUSB – USB A cable
ADXL326 accelerometer (#1018)
ALLPOWERS 2.5W, 5V photovoltaic (solar) cell

Arduino Mega 2560 and USB cable
BME 680 T/RH/P/VOC (#3660)
Diodes: 1N5817 Schottky
DS3231 real time clock (#3013)
electret microphone with amplifier (#1063)
RadioFeather M0 with 900 MHz LoRa radio (#3178)
GPS Antenna - External Active Antenna (#960)
INA219 battery voltage/current sensor (#904)
LSM9DS1 accelerometer/magnetometer/gyroscope 0x1E (#3387)
MCP23008 I2C port expander, Adafruit 593
MCP4725 DAC, 12-bit, I2C (#935)
MicroSD card breakout (#254)
MicroSD memory card 8 GB SDHC (#1294)
Mini Metal Speaker w/ Wires - 8 ohm 0.5W (#1890)
MLX90614 IR sensor. Note that there are distinct 3V and 5V versions!
PAM8302 Audio Amplifier (#2130)
red LEDs (#297)
TCA9548A mux I2C multiplexer
TMP36 analog medium temperature thermometers (#165)
TSL2561 or TSL2591 Digital Luminosity/Lux/Light sensor
ultimate GPS breakout board (#746)
USB DIY connectors

Installing the Arduino programming IDE; running code

Go to the Arduino webpage https://www.arduino.cc/en/software/. Download and install the Arduino Desktop IDE (Integrated Development Environment) on your laptop.

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


Arduino Mega 2560

It’s a remarkable little gizmo, featuring an Atmel Atmega2560 microcontroller (built by Microchip Technology, Inc.) 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 10-bit successive approximation analog to digital converter (ADC). See https://store.arduino.cc/arduino-mega-2560-rev3 for more details.

Some of the projects might involve a different processor: I have been developing systems using several Adafruit devices, two of which employ Microchip ATSAMD21G18 microcontrollers. The SAMD21 runs at 48 MHz, features a 12-bit ADC, and has 32 kB of SRAM, four times more than the Atmega2560. It also has a 10-bit digital to analog converter (DAC), which the 2560 does not.

The Adalogger M0 includes a built-in microSD memory holder, while the RadioFeather M0 with RFM95 LoRa radio has a built-in 915 MHz long range radio transceiver. Both are very cool.

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.

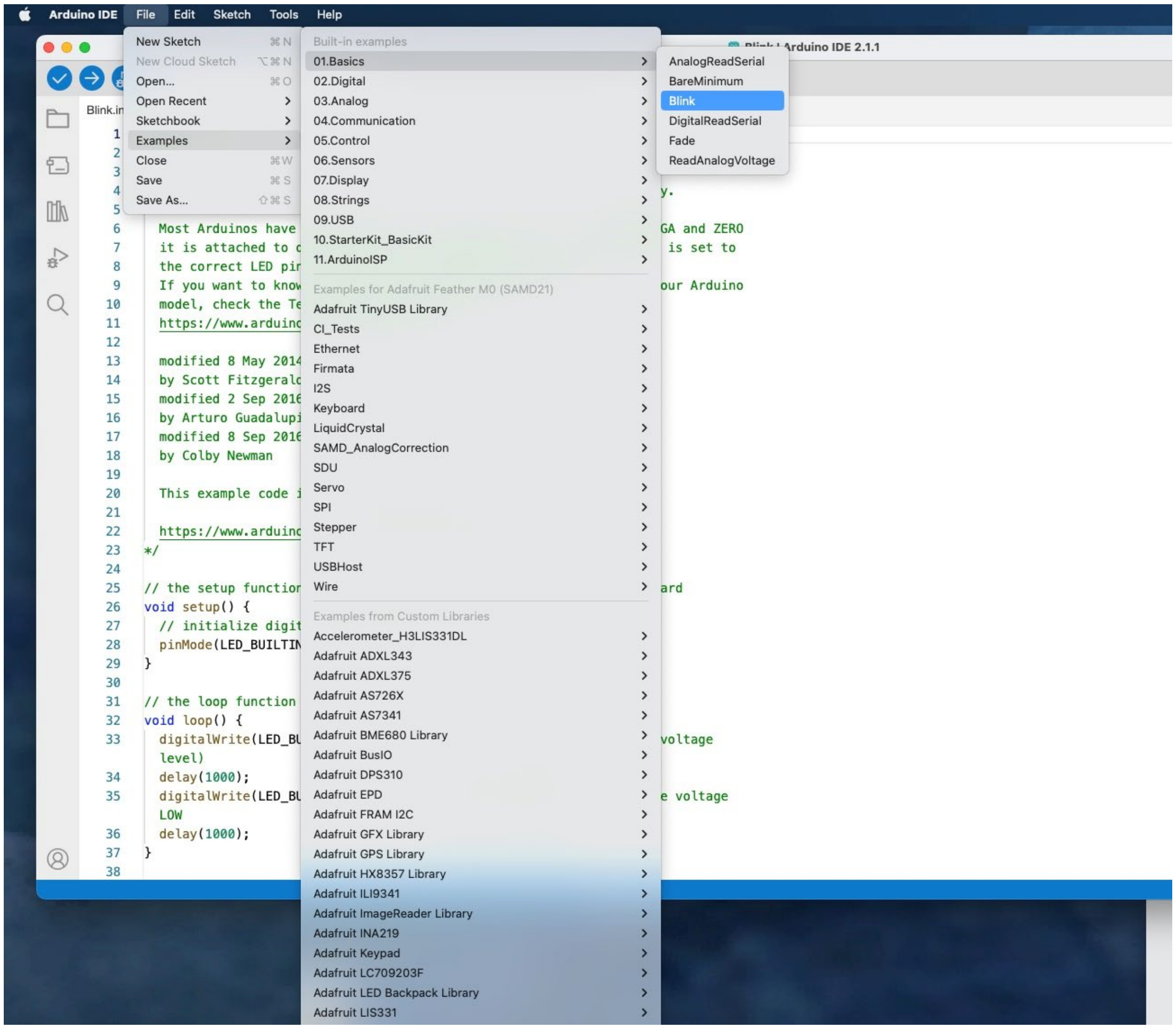
There isn’t a debugger, so you’ll be forced to print things to a “serial monitor” screen to keep track of what’s going on (and going wrong) in the code executed by the Arduino.

Something to keep in mind: the Arduino runs code that is very much like C++, with some minor differences. But the structure of a program is constrained: there are always two specific functions that must be included in a program. The first is called “setup”; it takes no arguments and does not return a value. It is the first routine in a program that executes. The next is called “loop.” It too takes no arguments and does not return a value. Upon completion of setup, the

Arduino goes straight into the loop function. It executes loop over and over, jumping back into it each time the function completes.

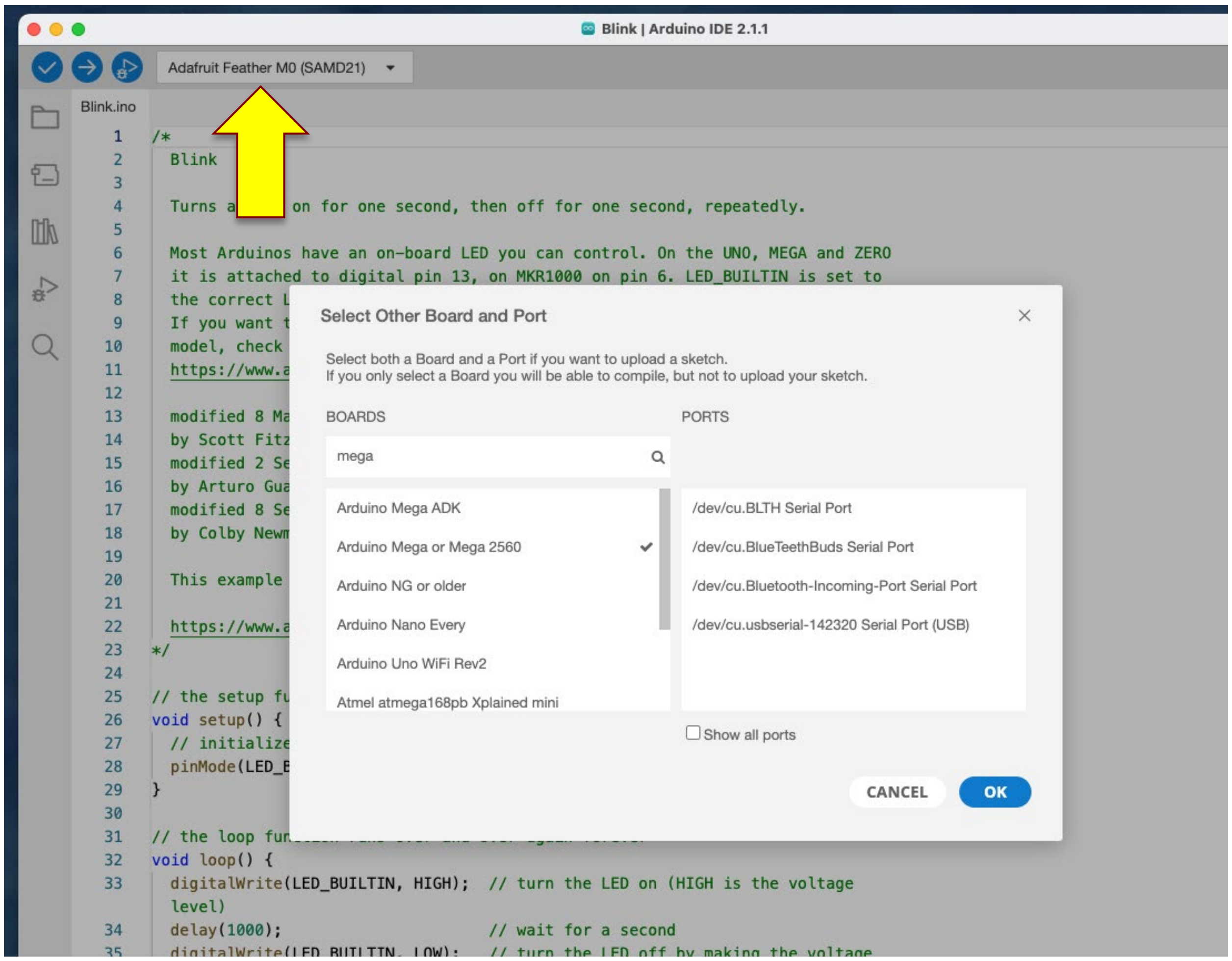
Do this: plug the Arduino into a USB port on your laptop, then fire up the IDE. Open the Blink example, compile and upload it, and see if your processor will talk to you. You'll need to make sure the IDE knows what kind of processor you are using: a Mega 2560. (Select this from the pull-down menu which, in my screen shot, below, initially says "Adafruit Feather M0 (SAMD21)."

Here are some screen shots:



```
                                                       y.

  Most Arduinos have                                   GA and ZERO
  it is attached to c                                   is set to
  the correct LED pir
  If you want to know                                  our Arduino
  model, check the Te
  https://www.arduinc

  modified 8 May 2014
  by Scott Fitzgeralc
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code i

  https://www.arduinc
*/

// the setup functior                                  ard
void setup() {
  // initialize digit
  pinMode(LED_BUILTIN
}

// the loop function
void loop() {
  digitalWrite(LED_BU                                  voltage
  level)
  delay(1000);
  digitalWrite(LED_BU                                  e voltage
  LOW
  delay(1000);
}
```

Opening the "blink" example.



```
/*
  Blink

  Turns a      on for one second, then off for one second, repeatedly.

  Most Arduinos have an on-board LED you can control. On the UNO, MEGA and ZERO
  it is attached to digital pin 13, on MKR1000 on pin 6. LED_BUILTIN is set to
  the correct L
  If you want t
  model, check
  https://www.a

  modified 8 Ma
  by Scott Fitz
  modified 2 Se
  by Arturo Gua
  modified 8 Se
  by Colby Newm

  This example

  https://www.a
*/

// the setup fu
void setup() {
  // initialize
  pinMode(LED_B
}

// the loop fun
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);  // turn the LED on (HIGH is the voltage
  level)
  delay(1000);                      // wait for a second
  digitalWrite(LED_BUILTIN, LOW);   // turn the LED off by making the voltage
```

Select Other Board and Port

Select both a Board and a Port if you want to upload a sketch.
If you only select a Board you will be able to compile, but not to upload your sketch.

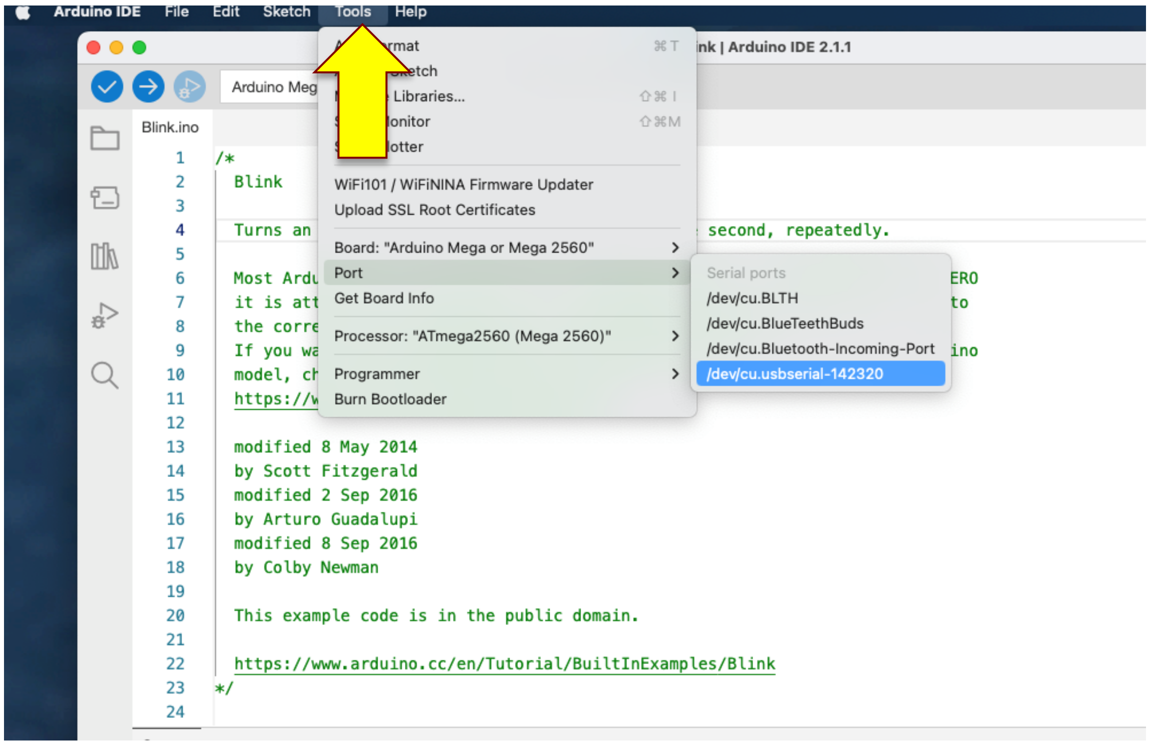
BOARDS	PORTS
mega	
Arduino Mega ADK	/dev/cu.BLTH Serial Port
Arduino Mega or Mega 2560 ✓	/dev/cu.BlueTeethBuds Serial Port
Arduino NG or older	/dev/cu.Bluetooth-Incoming-Port Serial Port
Arduino Nano Every	/dev/cu.usbserial-142320 Serial Port (USB)
Arduino Uno WiFi Rev2	
Atmel atmega168pb Xplained mini	

☐ Show all ports

CANCEL OK

Identifying the processor to be used.

You'll need to tell the IDE which port to use:

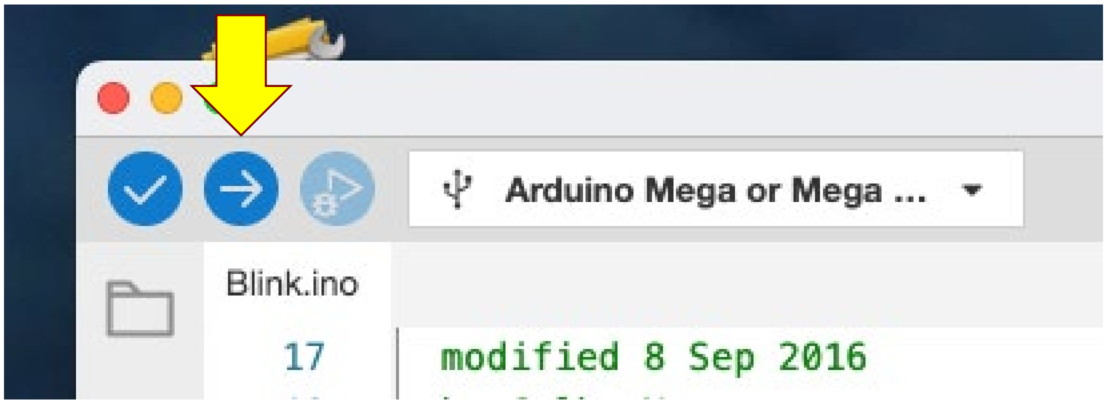


```
/*
  Blink

  Turns an                                          second, repeatedly.

  Most Ardu                                                           ERO
  it is att                                                           to
  the corre
  If you wa                                                           ino
  model, ch
  https://w

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is in the public domain.

  https://www.arduino.cc/en/Tutorial/BuiltInExamples/Blink
*/
```

Specifying the USB port to use.

What the program does is pretty obvious. Good coding references are on the Arduino site: see https://www.arduino.cc/reference/en/ and https://www.arduino.cc/en/reference/libraries.

You can compile the program upload it to your Arduino by clicking the right-arrow button near the top of the window:

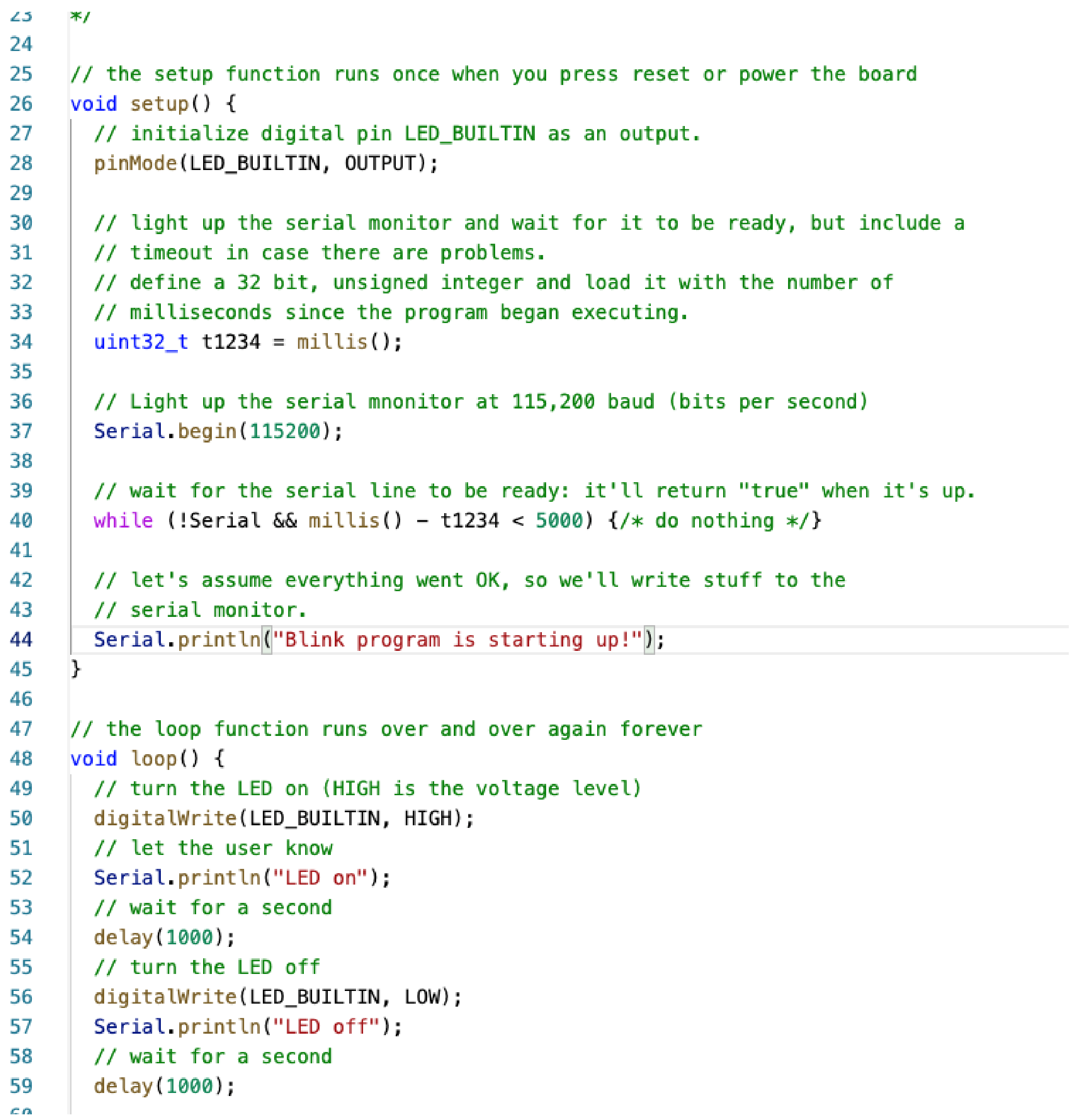


```
modified 8 Sep 2016
```

Note the presence of the setup and loop functions.

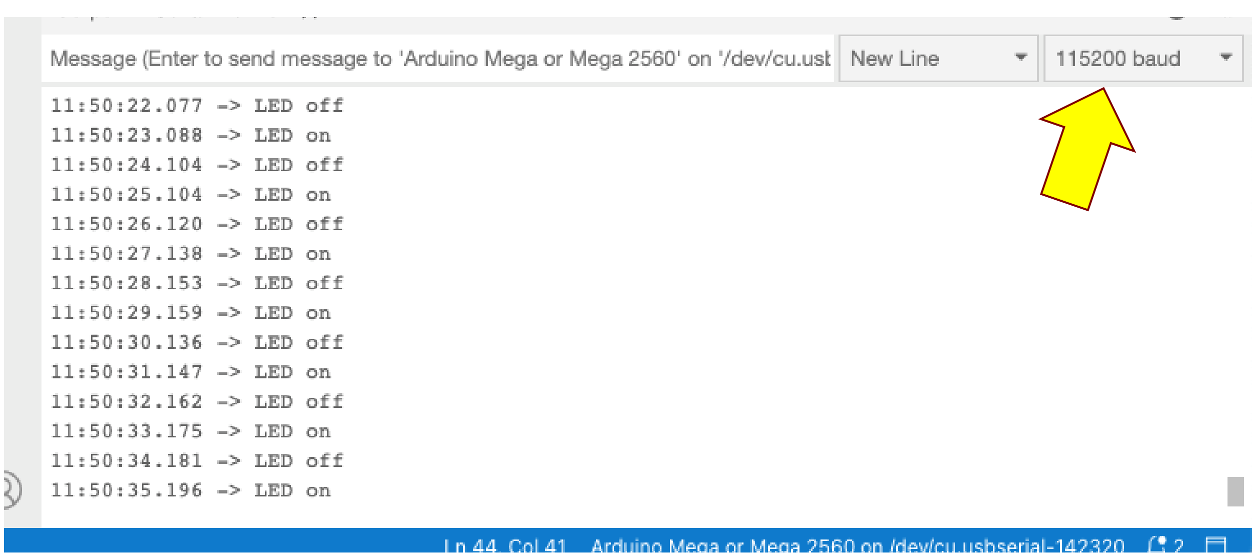
Some more technical commentary: many of the Arduino's "pins" are configurable: they can be defined to be digital inputs, or outputs, or analog inputs. The pinMode instruction defines the pin driving the red LED to be a (digital) output.

Here’s what you can do to view program output in a “serial monitor” window. First add some code to write to it:



```
*/

// 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);

  // light up the serial monitor and wait for it to be ready, but include a
  // timeout in case there are problems.
  // define a 32 bit, unsigned integer and load it with the number of
  // milliseconds since the program began executing.
  uint32_t t1234 = millis();

  // Light up the serial mnonitor at 115,200 baud (bits per second)
  Serial.begin(115200);

  // wait for the serial line to be ready: it'll return "true" when it's up.
  while (!Serial && millis() - t1234 < 5000) {/* do nothing */}

  // let's assume everything went OK, so we'll write stuff to the
  // serial monitor.
  Serial.println("Blink program is starting up!");
}

// the loop function runs over and over again forever
void loop() {
  // turn the LED on (HIGH is the voltage level)
  digitalWrite(LED_BUILTIN, HIGH);
  // let the user know
  Serial.println("LED on");
  // wait for a second
  delay(1000);
  // turn the LED off
  digitalWrite(LED_BUILTIN, LOW);
  Serial.println("LED off");
  // wait for a second
  delay(1000);
```

Then set the baud rate:


Message (Enter to send message to 'Arduino Mega or Mega 2560' on '/dev/cu.usb
New Line
115200 baud
Ln 44, Col 41 Arduino Mega or Mega 2560 on /dev/cu.usbserial-142320

```
11:50:22.077 -> LED off
11:50:23.088 -> LED on
11:50:24.104 -> LED off
11:50:25.104 -> LED on
11:50:26.120 -> LED off
11:50:27.138 -> LED on
11:50:28.153 -> LED off
11:50:29.159 -> LED on
11:50:30.136 -> LED off
11:50:31.147 -> LED on
11:50:32.162 -> LED off
11:50:33.175 -> LED on
11:50:34.181 -> LED off
11:50:35.196 -> LED on
```

In-class exercise: blink code

Modify the blink program so that it blinks the initials (of your English/American name) in Morse code.

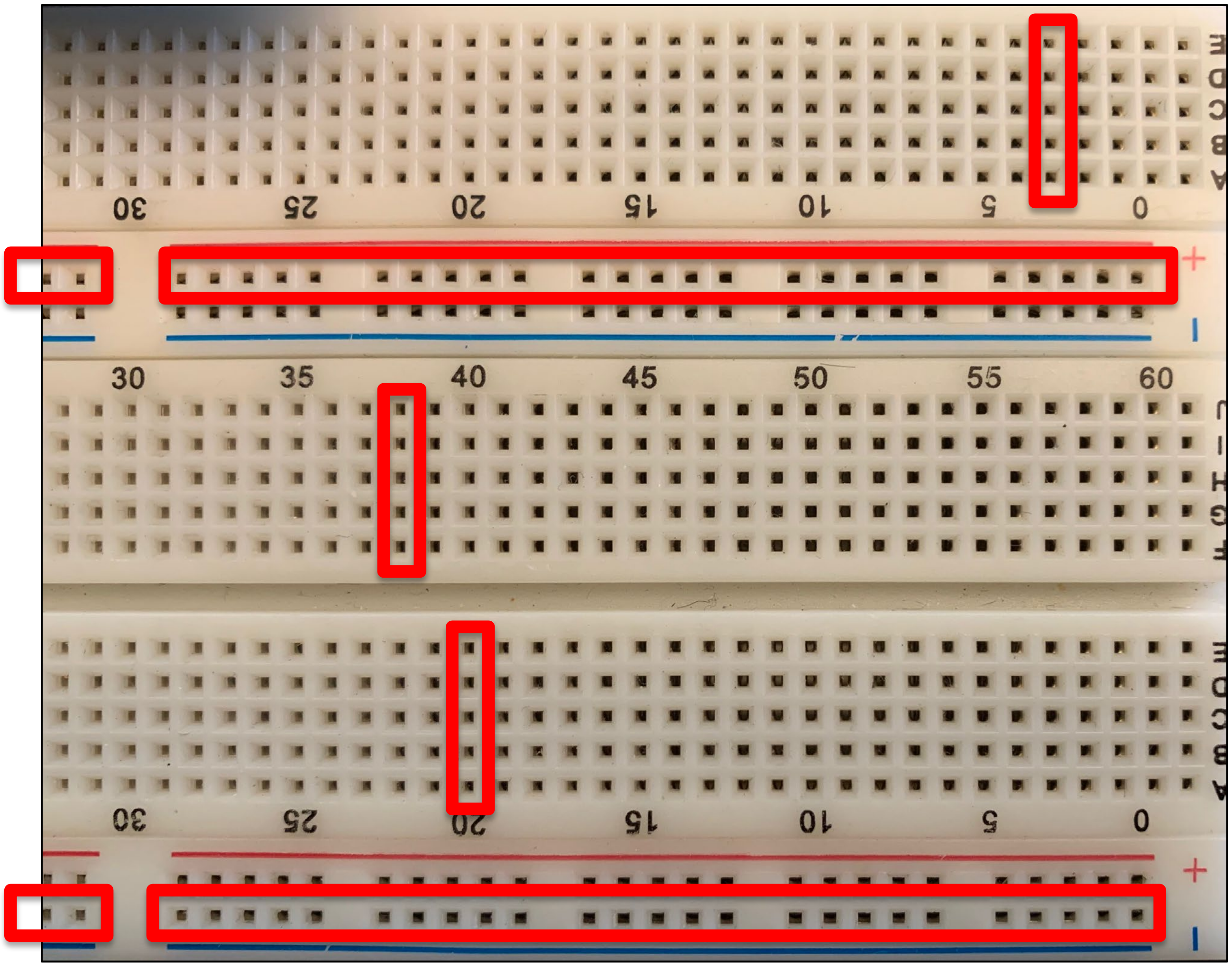
Breadboarding!

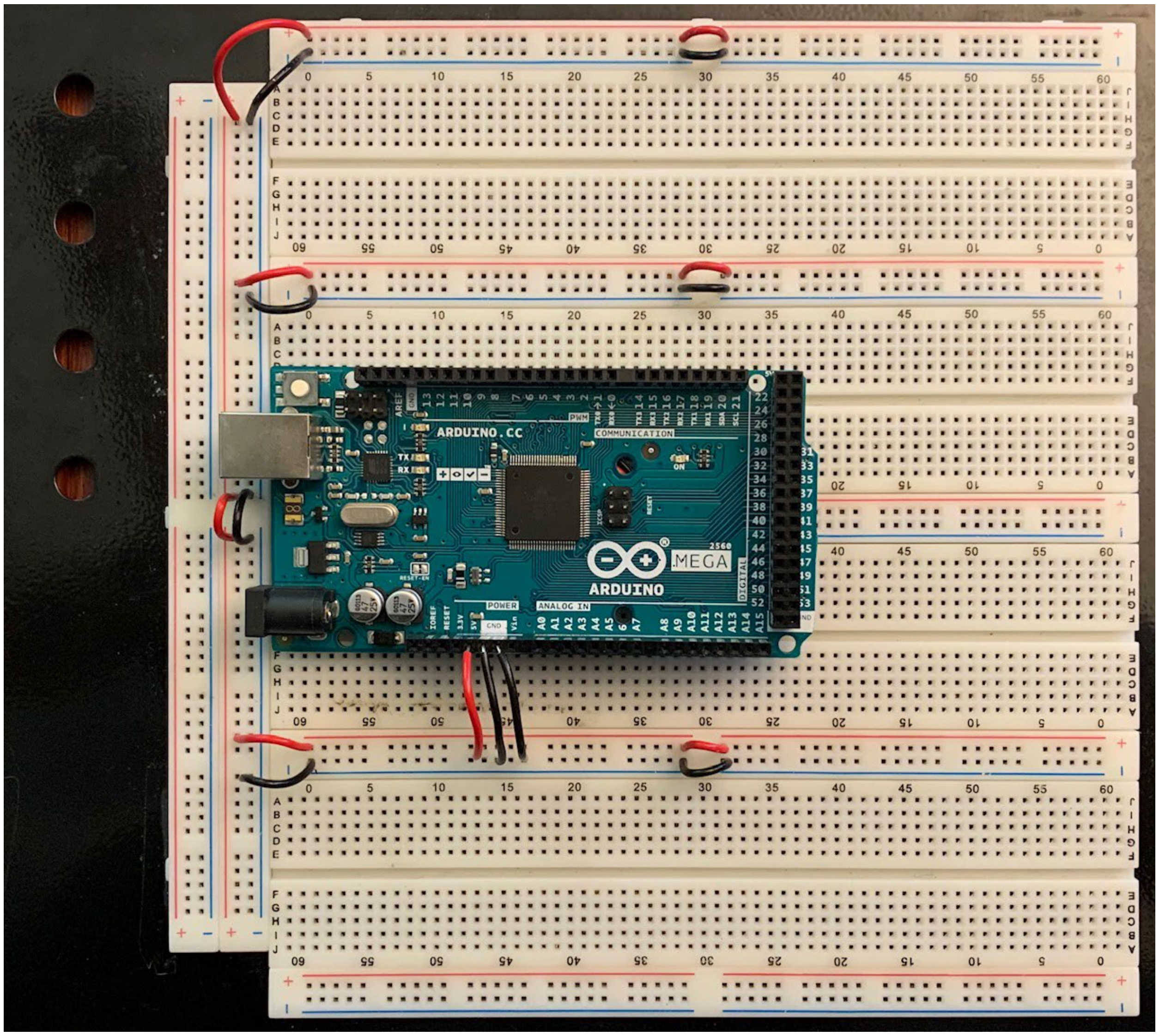
Let's prep our breadboards. Please fasten an Arduino to your breadboard. I recommend duct tape (the baby sitter's friend!); position the device to that it doesn't cover any of the breadboard's plastic structures that are used to hold components.

Here's how the holes are interconnected, underneath the plastic surface. The five holes in a column are connected, as shown in a few spots in the picture below. The 25 holes in a horizontal row are also connected.

In the photo after the close up I show a breadboard with an Arduino and connections between the Arduino's 5V and grounds to the breadboard. The doubled connection of the Arduino grounds is not electrically necessary, but it does provide redundancy in case one of the ground wires falls out.

Be sensible in your choices of wire colors: always use red for 5V and black for ground. You'll want to strip about 5 mm of insulation from each end of a wire when establishing your connections.

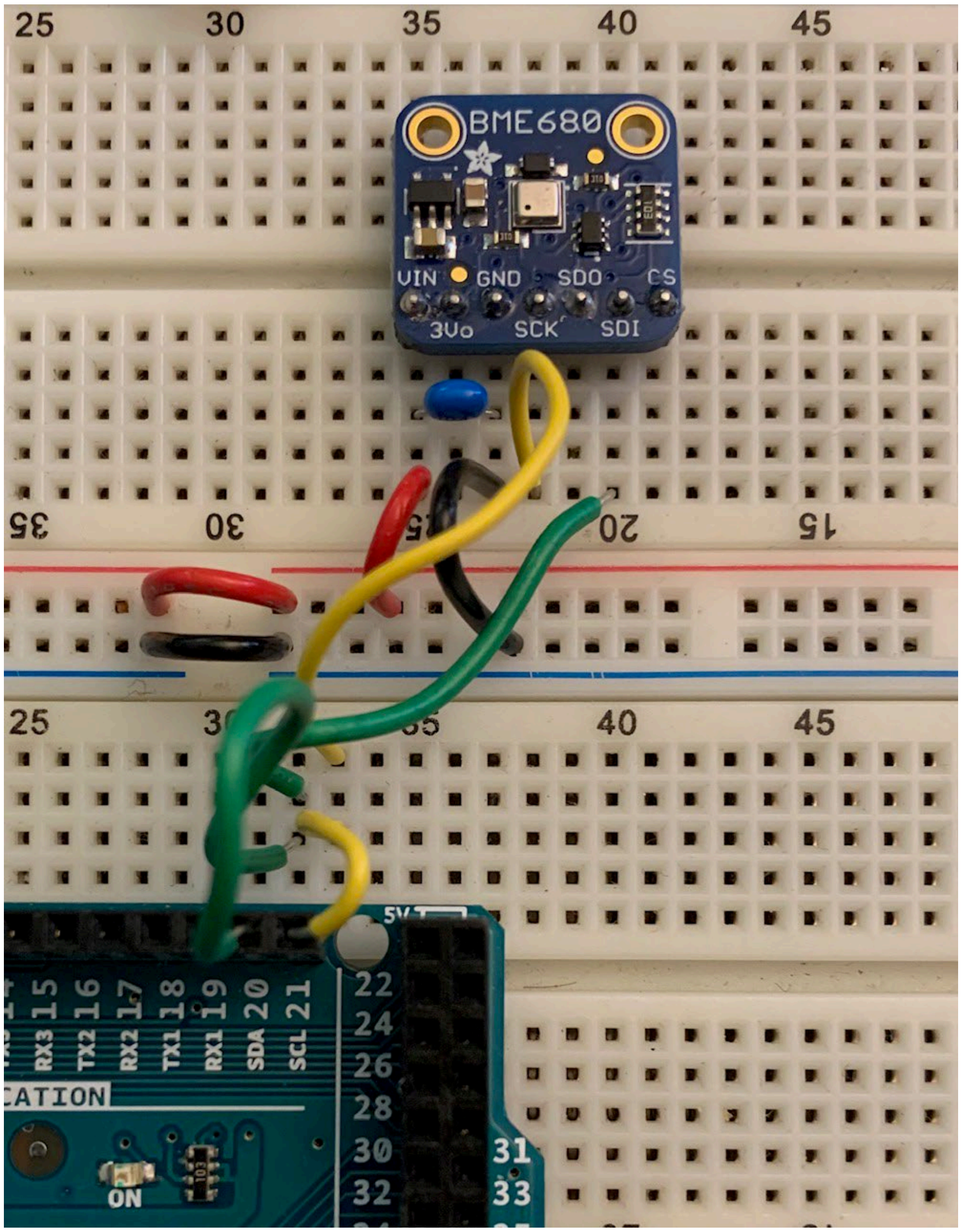

E
D
C
B
A
30
25
20
15
10
5
0
+
-
30
35
40
45
50
55
60
J
I
H
G
F



In-class exercise/milestone 1b: BME680

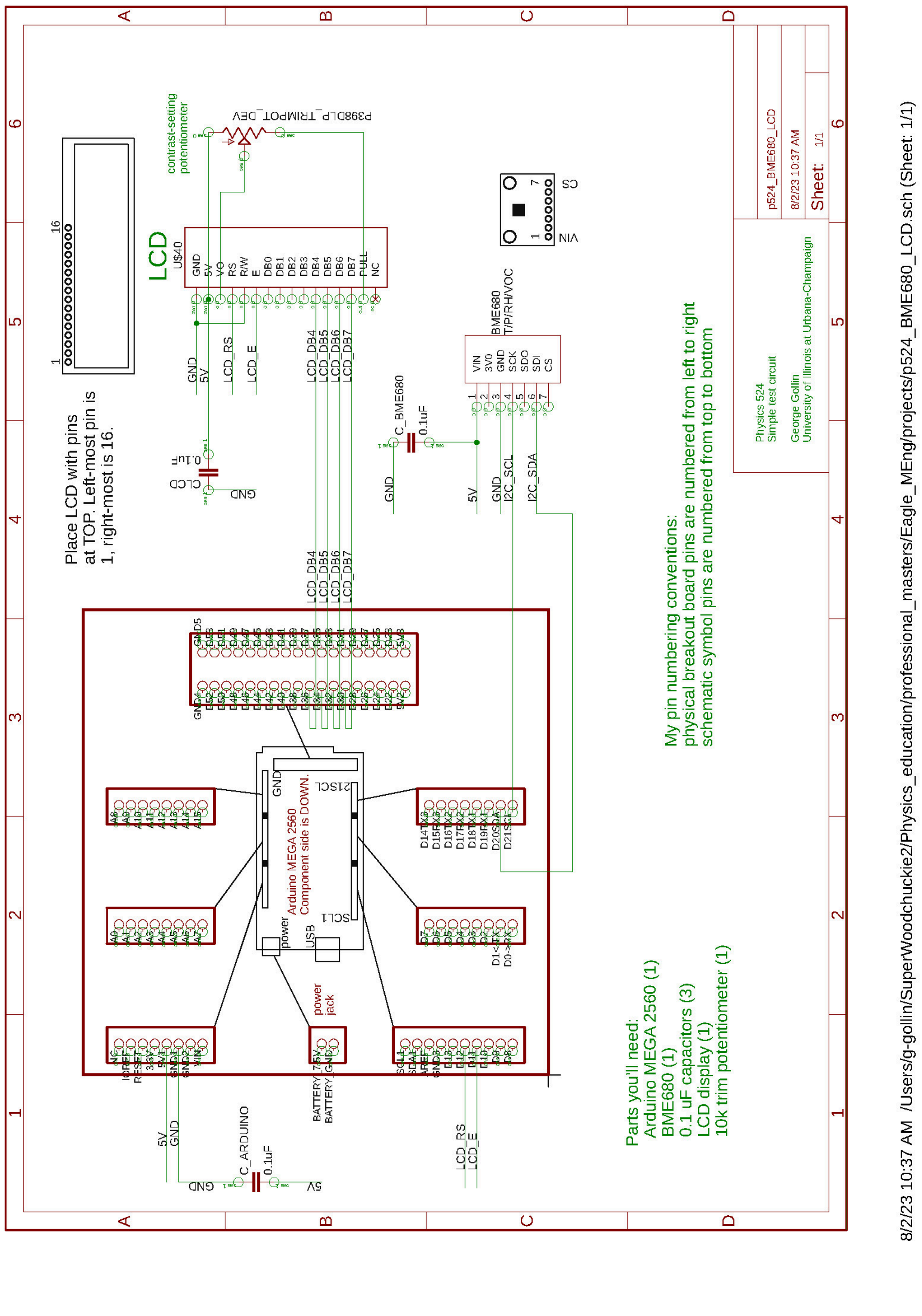
Most of our breakout boards were built by Adafruit Industries, a wonderful provider of small electronic packages intended largely for the hobbyist market. Go to the Adafruit site https://www.adafruit.com/ and find the BME680 page that mentions some of the supporting infrastructure available for you.

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. 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 leads of a 0.1μF capacitor to the BME680's power and ground inputs.

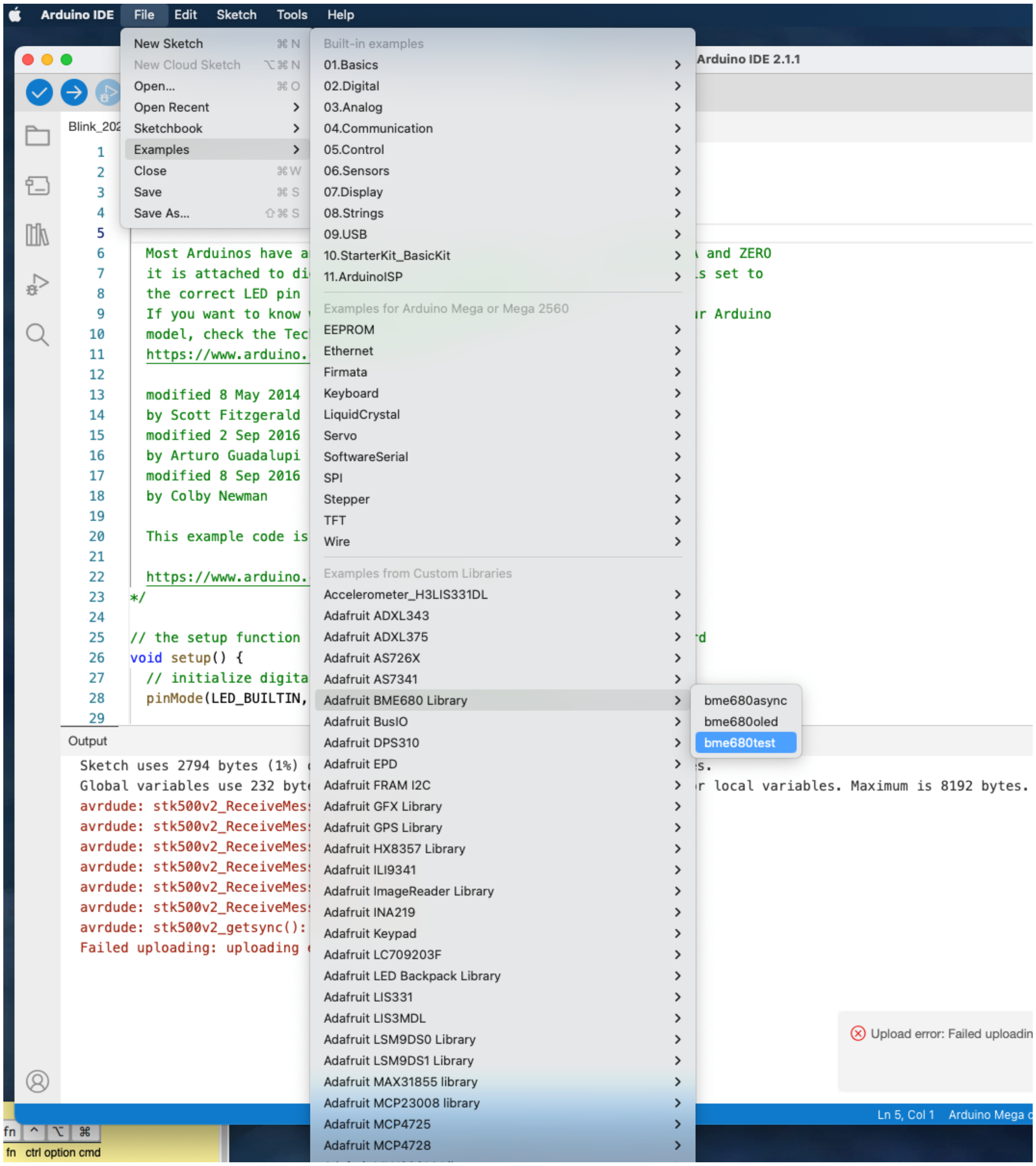


We’ll let the device and the Arduino communicate using an I2C (“I Two C”: Inter Integrated Circuit) interface; set this up by connecting the BME680’s SCK (serial clock) pin to the Arduino’s SCL output (pin 21). Also connect the BME680’s SDI (serial data) to the Arduino’s SDA input (pin 20). You should leave unconnected the BME680’s 3Vo, SDO, and CS pins.

Here's a schematic for the circuit that also includes a liquid crystal display. We’ll discuss briefly how to read the schematic. (Don’t bother with the LCD for the moment, though you are free to wire it up, if you’d like.)


LCD
U$40
GND
5V
VO
RS
R/W
E
DB0
DB1
DB2
DB3
DB4
DB5
DB6
DB7
PULL
NC
contrast-setting potentiometer
P398DLP_TRIMPOT_DEV
Place LCD with pins at TOP. Left-most pin is 1, right-most pin is 16.
CLCD
0.1uF
LCD_RS
LCD_E
LCD_DB4
LCD_DB5
LCD_DB6
LCD_DB7
BME680 T/P/RH/VOC
VIN
3V0
GND
SCK
SDO
SDI
CS
C_BME680
I2C_SCL
I2C_SDA
Arduino MEGA 2560 Component side is DOWN.
power jack
C_ARDUINO
My pin numbering conventions: physical breakout board pins are numbered from left to right schematic symbol pins are numbered from top to bottom
Parts you'll need: Arduino MEGA 2560 (1) BME680 (1) 0.1 uF capacitors (3) LCD display (1) 10k trim potentiometer (1)
Physics 524 Simple test circuit
George Gollin University of Illinois at Urbana-Champaign
p524_BME680_LCD
8/2/23 10:37 AM
Sheet: 1/1
8/2/23 10:37 AM /Users/g-gollin/SuperWoodchuckie2/Physics_education/professional_masters/Eagle_MEng/projects/p524_BME680_LCD.sch (Sheet: 1/1)

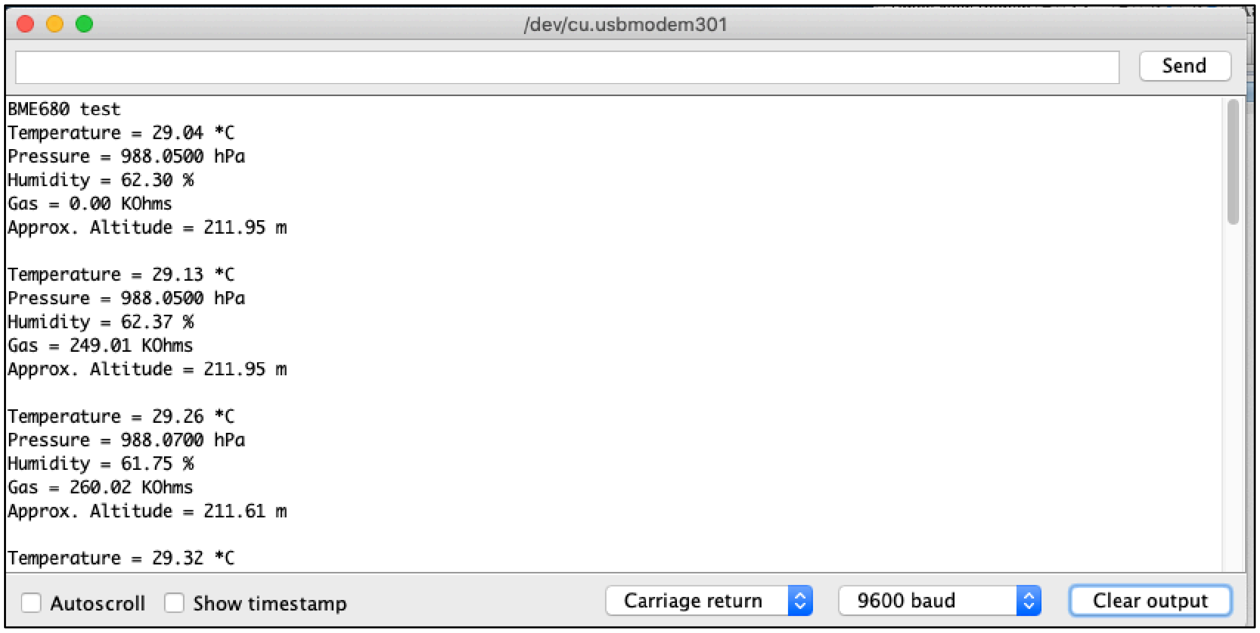
You'll need to install one of Adafruit's libraries to drive the BME680. See https://learn.adafruit.com/adafruit-bme680-humidity-temperature-barometic-pressure-voc-gas/arduino-wiring-test and scroll down to the section titled "Install Adafruit_BME680 library." Follow the directions to install the library and upload to the Arduino the demonstration software. Then open, compile, and run the example program BME680test.



```
  Most Arduinos have a                                    and ZERO
  it is attached to di                                   s set to
  the correct LED pin
  If you want to know                                    r Arduino
  model, check the Tec
  https://www.arduino.

  modified 8 May 2014
  by Scott Fitzgerald
  modified 2 Sep 2016
  by Arturo Guadalupi
  modified 8 Sep 2016
  by Colby Newman

  This example code is

  https://www.arduino.
*/

// the setup function                                    d
void setup() {
  // initialize digita
  pinMode(LED_BUILTIN,
```

```
Sketch uses 2794 bytes (1%)                               s.
Global variables use 232 byt                              r local variables. Maximum is 8192 bytes.
avrdude: stk500v2_ReceiveMes
avrdude: stk500v2_ReceiveMes
avrdude: stk500v2_ReceiveMes
avrdude: stk500v2_ReceiveMes
avrdude: stk500v2_ReceiveMes
avrdude: stk500v2_ReceiveMes
avrdude: stk500v2_getsync():
Failed uploading: uploading
```

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.



```
BME680 test
Temperature = 29.04 *C
Pressure = 988.0500 hPa
Humidity = 62.30 %
Gas = 0.00 KOhms
Approx. Altitude = 211.95 m

Temperature = 29.13 *C
Pressure = 988.0500 hPa
Humidity = 62.37 %
Gas = 249.01 KOhms
Approx. Altitude = 211.95 m

Temperature = 29.26 *C
Pressure = 988.0700 hPa
Humidity = 61.75 %
Gas = 260.02 KOhms
Approx. Altitude = 211.61 m

Temperature = 29.32 *C
```

Autoscroll Show timestamp Carriage return 9600 baud

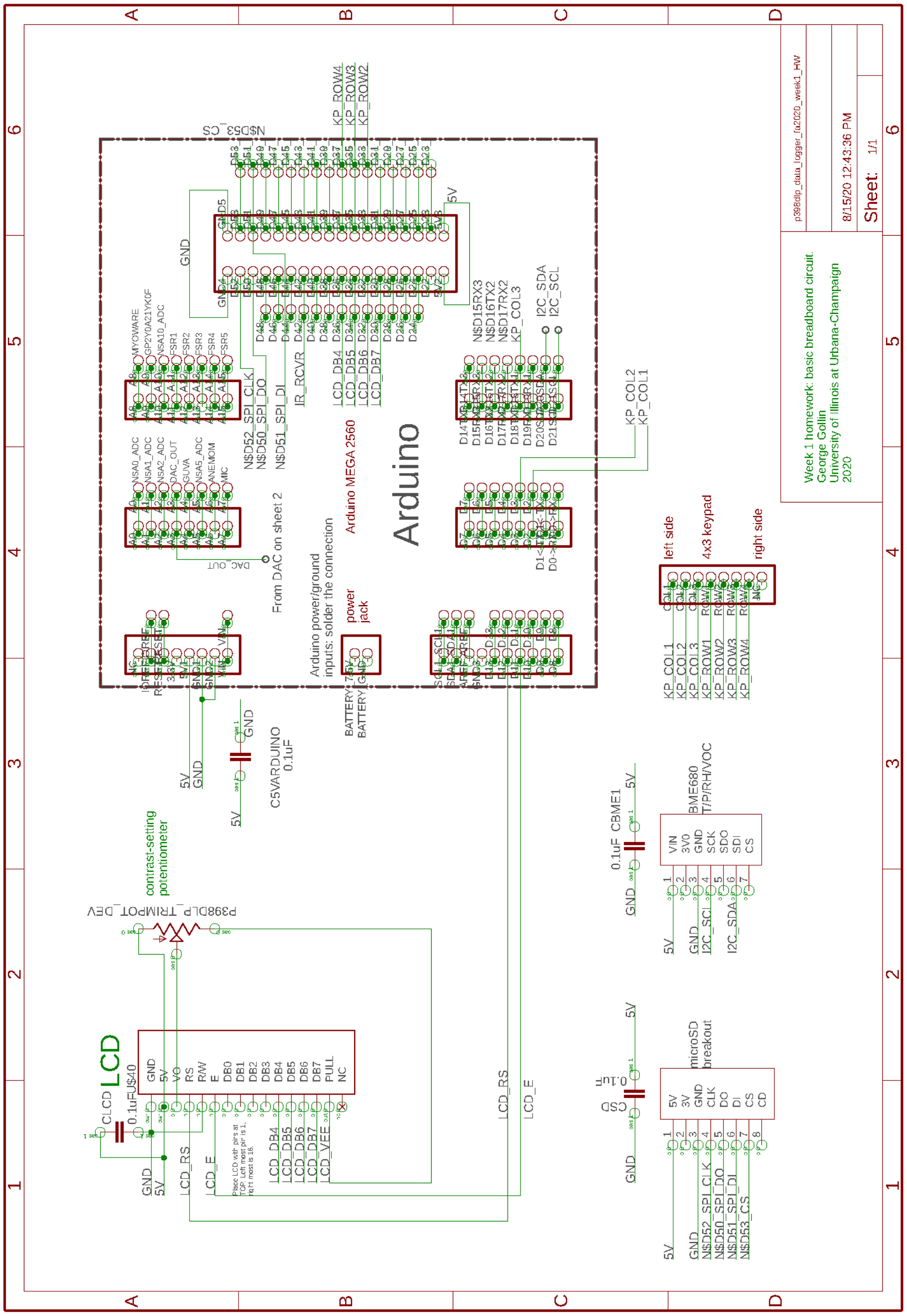
In-class (due at the first class meeting next week)

On your breadboard, install the following devices (in addition to the BME680 and Arduino): LCD (including 10kΩ trimpot), keypad, and microSD breakout. See the schematic, below.

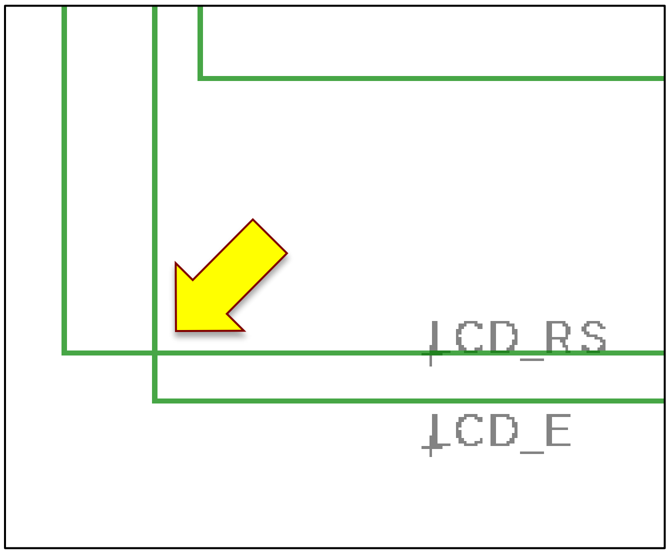
For each device find a demo program (perhaps on the Adafruit site, or from a library that you might install, or from the course's "Code & design resources repository") and confirm that it functions properly. You'll need to fool around with the 10k trimpot to adjust the LCD contrast properly.

How to read a schematic

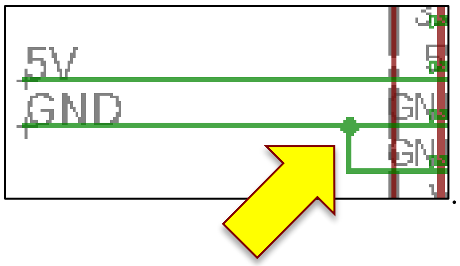
A schematic holds a topological representation of an electronic circuit. The two most important things on it are symbols for the various components—resistors, capacitors, integrated circuits and so forth—and (named) nets, which define the electrical connections between components. For example, the BME680 symbol on the schematic shows seven pins, with pins 1, 3, 4, and 6 connected to the nets named 5V, GND, I2C_SCL, and I2C_SDA, respectively. (Pins 2, 5, and 7 are not connected to anything.) Anything tied to the 5V net is electrically connected to everything else on the 5V net.


Arduino
Arduino MEGA 2560
LCD
contrast-setting potentiometer
P398DLP_TRIMPOT_DEV
C5VARDUINO 0.1uF
From DAC on sheet 2
Arduino power/ground inputs: solder the connection
power jack
BATTERY+
BATTERY GND
left side
4x3 keypad
right side
BME680 T/P/RH/VOC
0.1uF CBME1
microSD breakout
CSD 0.1uF
CLCD 0.1uF
Place LCD with pin's at TOP. Left most pin is 1, right most is 16.
LCD_RS
LCD_E
LCD_DB4
LCD_DB5
LCD_DB6
LCD_DB7
LCD_VEE
KP_COL1
KP_COL2
KP_COL3
KP_ROW1
KP_ROW2
KP_ROW3
KP_ROW4
N$D52_SPI_CLK
N$D50_SPI_DO
N$D51_SPI_DI
N$D53_CS
I2C_SDA
I2C_SCL
IR_RCVR
N$A0_ADC
N$A1_ADC
N$A2_ADC
DAC_OUT
GUVA
N$A5_ADC
ANEMOM
MIC
MYOWARE
GP2Y0A21YK0F
N$A10_ADC
FSR1
FSR2
FSR3
FSR4
FSR5
N$D15RX3
N$D16TX2
N$D17RX2
5V
GND
Week 1 homework: basic breadboard circuit.
George Gollin
University of Illinois at Urbana-Champaign
2020
p398dlp_data_logger_fa2020_week1_HW
8/15/20 12:43:36 PM
Sheet: 1/1

When lines representing nets cross each other the point of crossing does ***not*** represent a connection between the nets! For example, the LCD_RS and LCD_E nets are not connected to each other here:



We indicate a point at which two lines (which are on the ***same*** net) are electrically connected with a dot as follows:



This week's homework assignment (due at the first class meeting next week)

Finish all of the in-class exercises in this unit, and be prepared to show your results to the course staff during class.