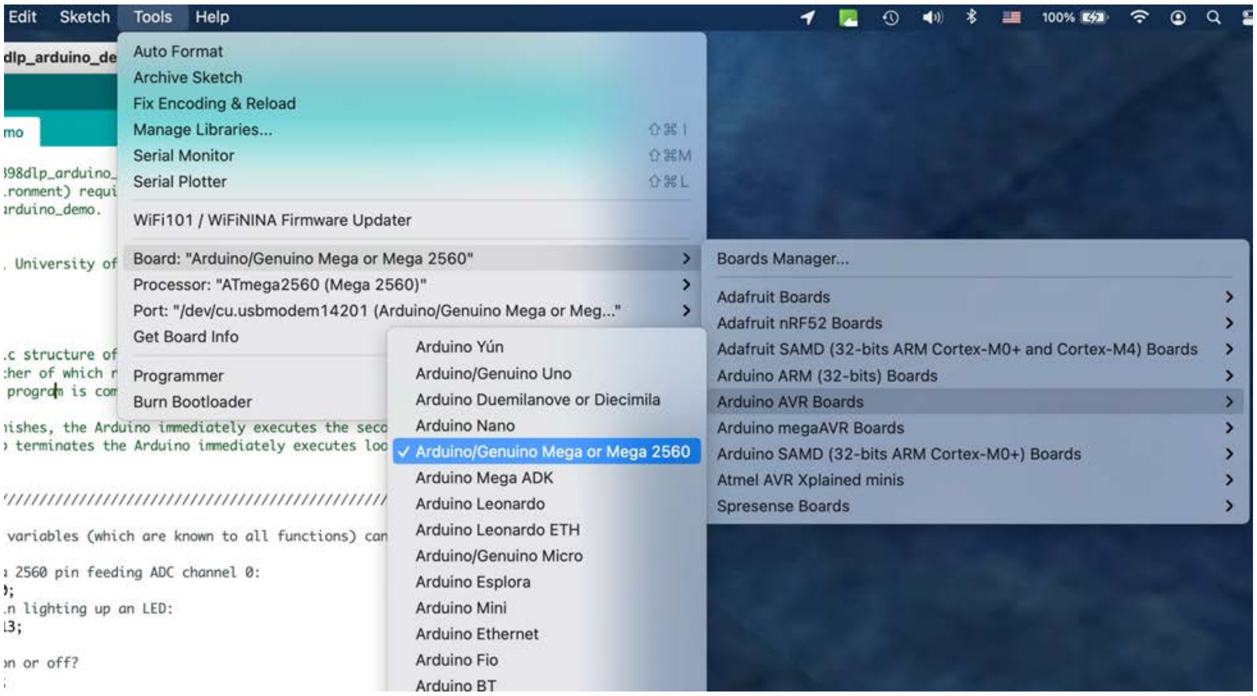
## *Arduino IDE*

You'll be using the Arduino IDE (integrated developer's environment) to generate, load, and execute software in your Arduino.

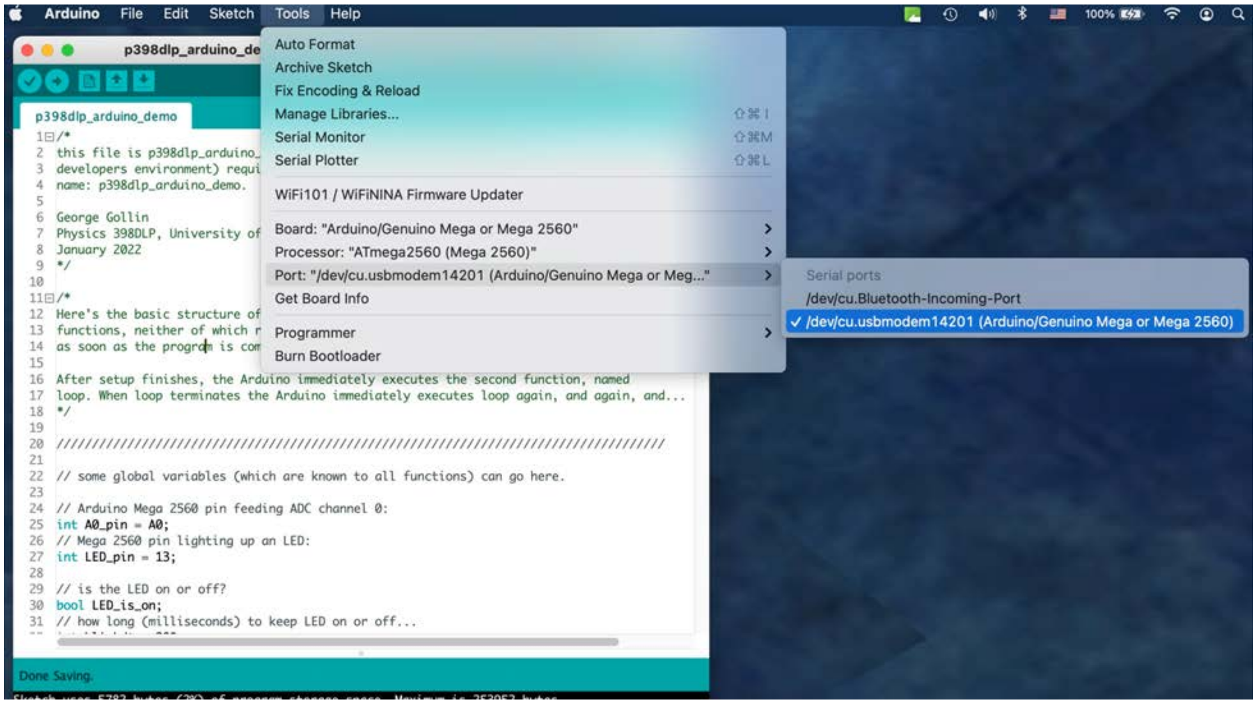
The basics:

- The IDE expects a program to reside inside a folder with the same name as the filename. We'll be working with the program arduino_demo.ino which lives inside the folder arduino_demo.
- An Arduino program must contain at least two functions, neither of which returns a value. The first, named setup, executes once, as soon as the program is compiled and begins to execute. After setup finishes, the Arduino immediately executes the second function, named loop. When loop terminates the Arduino immediately executes loop again, and again, and...
- The Arduino programming language looks very much like C++.
- You'll have the Arduino connect to your laptop through a USB cable.
- You need to tell the IDE what kind of processor it'll be dealing with, and what USB port will be used to talk to the Arduino.

Here's a screen shot of the IDE as I tell it I am using an Arduino Mega 2560:


Board: "Arduino/Genuino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port: "/dev/cu.usbmodem14201 (Arduino/Genuino Mega or Meg..."
✓ Arduino/Genuino Mega or Mega 2560

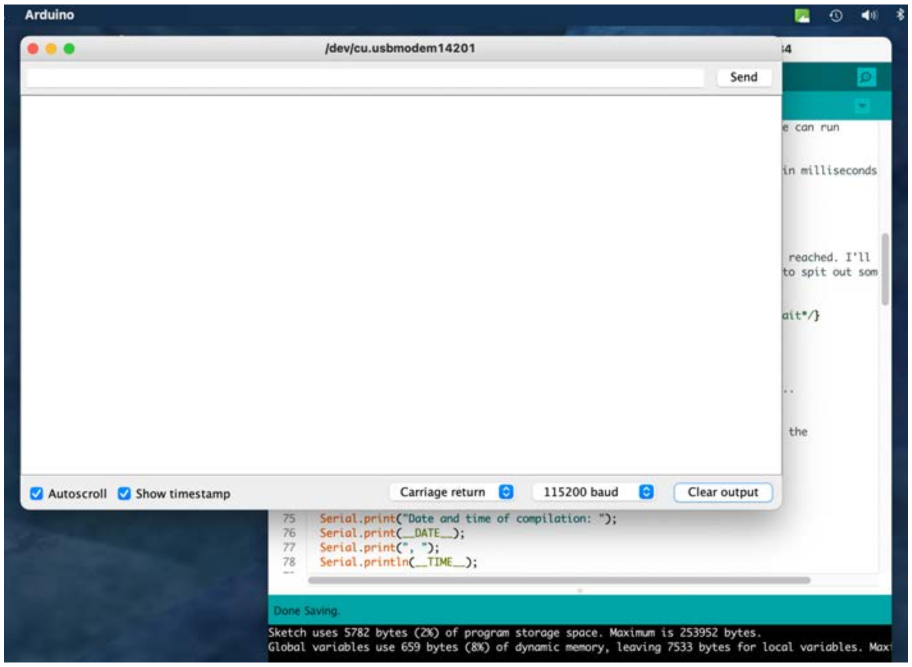
…and here's a screen shot in which I specify the USB port I'll be using.


Arduino File Edit Sketch Tools Help
p398dlp_arduino_de
p398dlp_arduino_demo
Auto Format
Archive Sketch
Fix Encoding & Reload
Manage Libraries... ⇧⌘I
Serial Monitor ⇧⌘M
Serial Plotter ⇧⌘L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino/Genuino Mega or Mega 2560"
Processor: "ATmega2560 (Mega 2560)"
Port: "/dev/cu.usbmodem14201 (Arduino/Genuino Mega or Meg..."
Get Board Info
Programmer
Burn Bootloader
Serial ports
/dev/cu.Bluetooth-Incoming-Port
✓ /dev/cu.usbmodem14201 (Arduino/Genuino Mega or Mega 2560)
Done Saving.

```
/*
this file is p398dlp_arduino_
developers environment) requi
name: p398dlp_arduino_demo.

George Gollin
Physics 398DLP, University of
January 2022
*/

/*
Here's the basic structure of
functions, neither of which r
as soon as the program is com

After setup finishes, the Arduino immediately executes the second function, named
loop. When loop terminates the Arduino immediately executes loop again, and again, and...
*/

////////////////////////////////////////////////////////////////////////////////

// some global variables (which are known to all functions) can go here.

// Arduino Mega 2560 pin feeding ADC channel 0:
int A0_pin = A0;
// Mega 2560 pin lighting up an LED:
int LED_pin = 13;

// is the LED on or off?
bool LED_is_on;
// how long (milliseconds) to keep LED on or off...
```

You can open up a serial monitor window from the Tools menu. Make sure you set the “baud rate” to agree with the Serial.begin() statement in your program:

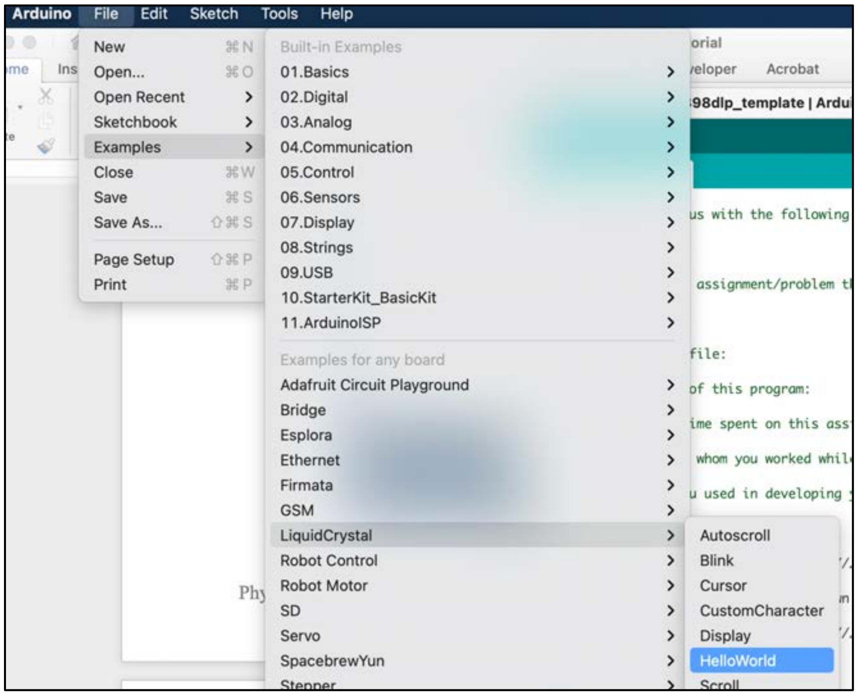
```

// open the serial communication line
Serial.begin(115200);

```



```
Serial.print("Date and time of compilation: ");
Serial.print(__DATE__);
Serial.print(", ");
Serial.println(__TIME__);
```

```
Sketch uses 5782 bytes (2%) of program storage space. Maximum is 253952 bytes.
Global variables use 659 bytes (8%) of dynamic memory, leaving 7533 bytes for local variables. Max
```

This is useful: in the IDE’s File menu you can find a large number of demonstration programs.



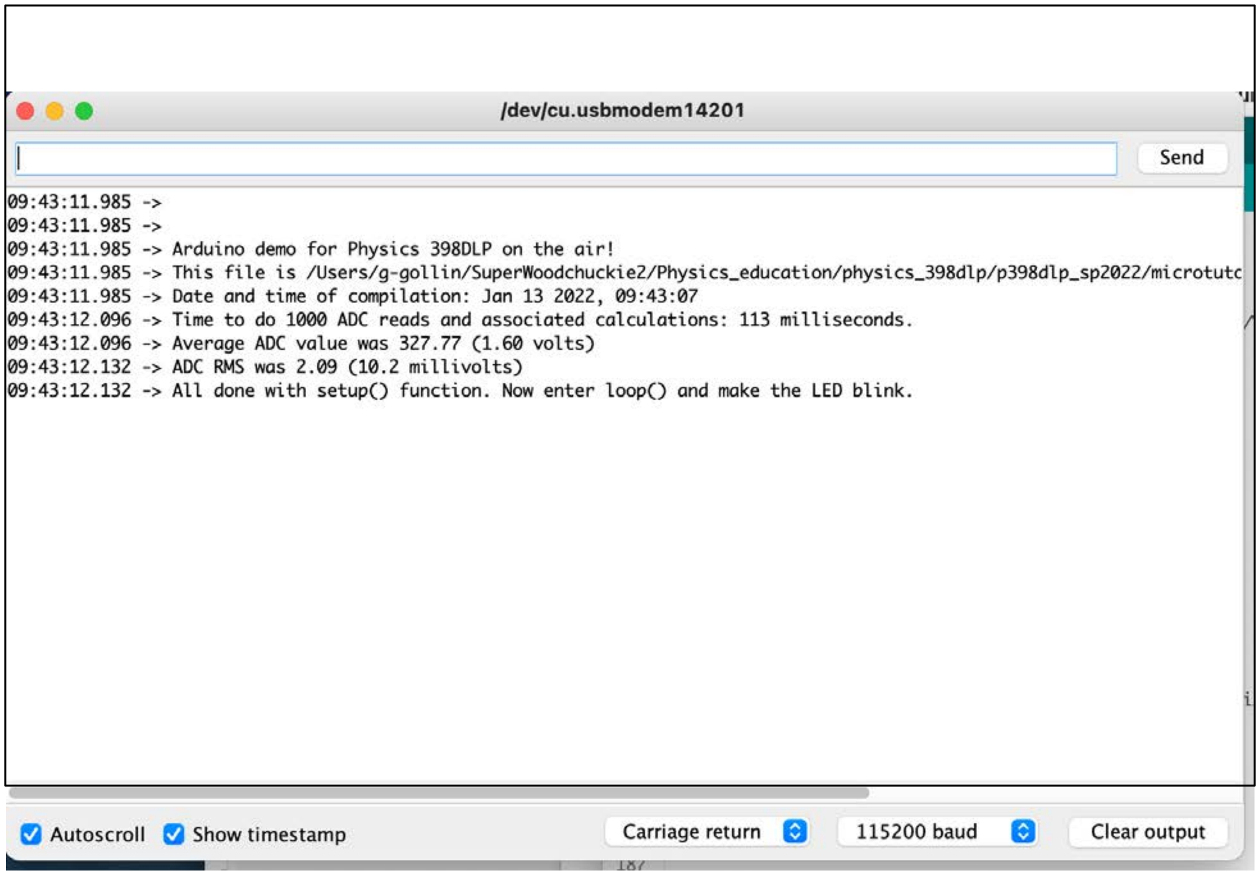
I’ve put a p398dlp demonstration program on the course web site, in the “Code and design resource repository” page.

Here’s a full list of this program, as of today: it reads the analog signal from an electret microphone 1,000 times and calculates the average and RMS signal values. Since the Arduino processor includes a 10-bit ADC (analog to digital converter), digitizations of the microphone signal will range from 0 to 1023 “counts,” corresponding to signals between 0 and 5 volts.

```
/*
this file is p398dlp_arduino_demo.ino. The Arduino IDE (integrated
developer’s environment) requires it to reside inside a folder with a similar
name: p398dlp_arduino_demo.

George Gollin
Physics 398DLP, University of Illinois
January 2022
*/

/*
Here's the basic structure of an Arduino program: there must be at least two
functions, neither of which returns a value. The first, named setup, executes once,
as soon as the program is compiled and begins to execute.

After setup finishes, the Arduino immediately executes the second function, named
loop. When loop terminates the Arduino immediately executes loop again, and again,
and...
```

```
*/

////////////////////////////////////////////////////////////////////////////////////

// some global variables (which are known to all functions) can go here.

// Arduino Mega 2560 pin feeding ADC channel 0:
int A0_pin = A0;
// Mega 2560 pin lighting up an LED:
int LED_pin = 13;

// is the LED on or off?
bool LED_is_on;
// how long (milliseconds) to keep LED on or off...
int blinkdt = 200;

// set the maximum waiting time for the serial monitor to open to be 5,000
milliseconds.
uint32_t t_give_up = 5000;

// ADC voltage reference
float ADC_Vref = 5.0;

////////////////////////////////////////////////////////////////////////////////////

void setup()
{
  // fire up the serial monitor and set its speed to be 115,200 baud.
  // make sure the baud rate setting (bottom of the serial monitor window,
  // towards the right side) is set to the proper speed...

  // wait for the serial line to finish opening, but include a timeout so we can run
  // this program without the serial monitor open.

  // declare a 32-bit unsigned integer to hold the time-since-compilation (in ms)
  uint32_t t_start = millis();

  // open the serial communication line
  Serial.begin(115200);

  // now wait for it to be finished opening up, or the time-out limit to be reached.
I'll
  // add a bit of additional delay on top of this, since the Arduino seems to spit out
some
  // gibberish if I don't let it finish waking up...

  while(!Serial && millis() - t_start < t_give_up) {/*do nothing while we wait*/}

  // here's the extra delay: 1,000 milliseconds.
  delay(1000);

  // if serial doesn't open, we're stuck. However, let's assume we're fine...
  Serial.println("\n\nArduino demo for Physics 398DLP on the air!");

  // this is handy... let's tell the user what file holds this program, and the
  // date and time it was most recently compiled.

  Serial.print("This file is " );
  Serial.println(__FILE__);

  Serial.print("Date and time of compilation: ");
  Serial.print(__DATE__);
  Serial.print(", ");
  Serial.println(__TIME__);

  // note the use of Serial.print and Serial.println, above. ...ln includes a line
break.

  // now let's read the analog voltage at pin A0, which is where I have a microphone
attached.
  // microphone voltage will vary between 0 and about 3.3 volts, since we're powering
it using the
  // Arduino's 3.3V regulated output. Note that the ADC digitization is 10 bits,
spanning the full
  // range from 0 to 5 volts. The microphone output in a quiet environment is half-way
between ground
  // and it "VCC" value, or about 1.65 volts. This'll map into an ADC value of 1024 *
1.65 / 5 = 338.

  // let's use a "for" loop, and read the microphone a lot of times. let's also
calculate an average,
  // and keep track of how long it takes to do the reads.

  // initialize some variables. uint32_t is an unsigned 32-bit integer.

  uint32_t number_of_reads = 1000;
  int A0_val;
  uint32_t A0_sq_val;
  uint32_t A0_sum = 0;
  uint32_t A0_sq_sum = 0;

  // record starting time
  t_start = millis();

  // here's the ADC-reading loop.
  for(int i = 0; i < number_of_reads; i++)
  {
    // read the ADC
    A0_val = analogRead(A0);

    // accumulate...
    A0_sum = A0_sum + A0_val;

    // for use in calculating an RMS get a 32-bit version of the ADC value,
    // then square it.
    A0_sq_val = A0_val;
    A0_sq_val = A0_sq_val * A0_sq_val;

    // accumulate...
    A0_sq_sum = A0_sq_sum + A0_sq_val;
  }

  // done, so take note of the time we spent inside the for loop.
  uint32_t t_elapsed = millis() - t_start;

  // calculate the average ADC value now...
  float A0_avg = float(A0_sum) / number_of_reads;

  // also calculate the RMS.
  float A0_sq_avg = float(A0_sq_sum) / number_of_reads;
  float A0_RMS = sqrt(A0_sq_avg - A0_avg * A0_avg);

  // report, then leave setup().
  Serial.print("Time to do ");
  Serial.print(number_of_reads);
  Serial.print(" ADC reads and associated calculations: ");
  Serial.print(t_elapsed);
  Serial.println(" milliseconds.");

  // print the average with two digits to the right of the decimal point as precision.
  Serial.print("Average ADC value was ");
  Serial.print(A0_avg, 2);
  Serial.print(" (");
  Serial.print(ADC_Vref * A0_avg / 1024., 2);
  Serial.println(" volts)");

  // print the RMS too, with voltage in millivolts.
  Serial.print("ADC RMS was ");
  Serial.print(A0_RMS, 2);
  Serial.print(" (");
  Serial.print(1000. * ADC_Vref * A0_RMS / 1024., 1);
  Serial.println(" millivolts)");

  // before we go... declare the LED pin to be an output so we can write values to it.
  pinMode(LED_pin, OUTPUT);

  // turn off the LED.
  LED_is_on = false;
  digitalWrite(LED_pin, LOW);

  Serial.println("All done with setup() function. Now enter loop() and make the LED 
blink.");

  // that's it for setup!
}

////////////////////////////////////////////////////////////////////////////////////

void loop() {
  // let's blink the LED on and off.

  if(LED_is_on)
  {
    // turn off the LED.
    digitalWrite(LED_pin, LOW);

  } else {

    // turn on the LED.
    digitalWrite(LED_pin, HIGH);
  }

  // now toggle the LED on/off switch. An exclamation point is negation.
  LED_is_on = !LED_is_on;

  // now delay before leaving/reentering loop
  delay(blinkdt);

  // that's it!

}
////////////////////////////////////////////////////////////////////////////////
```

Here's a full list of the program, as of today: it reads the analog signal from an electret micophone. When I run the code the following lands in the serial monitor window:



```
09:43:11.985 ->
09:43:11.985 ->
09:43:11.985 -> Arduino demo for Physics 398DLP on the air!
09:43:11.985 -> This file is /Users/g-gollin/SuperWoodchuckie2/Physics_education/physics_398dlp/p398dlp_sp2022/microtutc
09:43:11.985 -> Date and time of compilation: Jan 13 2022, 09:43:07
09:43:12.096 -> Time to do 1000 ADC reads and associated calculations: 113 milliseconds.
09:43:12.096 -> Average ADC value was 327.77 (1.60 volts)
09:43:12.132 -> ADC RMS was 2.09 (10.2 millivolts)
09:43:12.132 -> All done with setup() function. Now enter loop() and make the LED blink.
```

☑ Autoscroll ☑ Show timestamp    Carriage return    115200 baud    Clear output