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.

Edit Sketch	Tools Help			イ 🚬 🕙 🜗 🖇 💻 100% 🐲 🙃 Q	8
dlp_arduino_de mo 998dlp_arduino_ .ronment) requi arduino_demo.	Auto Format Archive Sketch Fix Encoding & Reload Manage Libraries Serial Monitor Serial Plotter WiFi101 / WiFiNINA Firmware Update		0-96 I 0-96M 0-96 L		
	Board: "Arduino/Genuino Mega or M Processor: "ATmega2560 (Mega 256 Port: "/dev/cu.usbmodem14201 (Ard Get Board Info Programmer Burn Bootloader Jino immediately executes the seco e Arduino immediately executes loo	o)" duino/Genuino Mega or Meg" Arduino Yún Arduino/Genuino Uno Arduino Duemilanove or Diecimil Arduino Nano ✓ Arduino/Genuino Mega or Mega	7.	Boards Manager Adafruit Boards Adafruit nRF52 Boards Adafruit SAMD (32-bits ARM Cortex-M0+ and Cortex-M4) Boards Arduino ARM (32-bits) Boards Arduino AVR Boards Arduino megaAVR Boards Arduino SAMD (32-bits ARM Cortex-M0+) Boards	>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
variables (whic	ch are known to all functions) can ing ADC channel 0: an LED:	Arduino Mega ADK Arduino Leonardo Arduino Leonardo ETH Arduino/Genuino Micro Arduino Esplora Arduino Mini Arduino Ethernet Arduino Fio Arduino BT		Atmel AVR Xplained minis Spresense Boards	>

Here's a screen shot of the IDE as I tell it I am using an Arduino 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			0	4 9)	* 💻	100%	ŝ	e q
p398dlp_arduino_de	Auto Format Archive Sketch Fix Encoding & Reload Manage Libraries	081							
10/*	Serial Monitor	① 第M	And in the second						
2 this file is p398dlp_arduino_ 3 developers environment) requi	Serial Plotter	⊕ ¥ L	100						
4 name: p398dlp_arduino_demo. 5	WiFi101 / WiFiNINA Firmware Updater		140.00						
6 George Gollin 7 Physics 3980LP, University of 8 January 2022 9 */	Board: "Arduino/Genuino Mega or Mega 2560" Processor: "ATmega2560 (Mega 2560)"	>							
10	Port: "/dev/cu.usbmodem14201 (Arduino/Genuino Mega or Meg"	>	Serial ports						
110/*	Get Board Info		/dev/cu.Bluetoor	h-Incom	ing-Po	rt			
12 Here's the basic structure of 13 functions, neither of which r 14 as soon as the program is com 15	Programmer Burn Bootloader	>	✓ /dev/cu.usbmod	em1420	1 (Ardı	iino/Genu	ino Mega o	Mega :	2560)
16 After setup finishes, the Ardu 17 loop. When loop terminates the 18 */ 19 20 ////////////////////////////////////	in LED:								

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:

3 4 // open the se 5 <mark>Serial.begin(</mark> 1 6	erial communication line L15200);	
Arduino		P 0 📲
	/dev/cu.usbmodem14201	14
	Send	
		e can run
		in milliseconds
		reached. I'll to spit out som ait*/}
Autoscroll 🗹 Show timestamp	Carriage return 🟮 115200 baud 🧿 Clear output	5
	<pre>75 Serial.print("Date and time of compilation: "); 76 Serial.print(_DATE); 77 Serial.print(_TDME); 78 Serial.println(_TDME);</pre>	
	and the second se	
	Dane Saving. 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 l	ocal variables. Max

0 4	New 36 N	Built-in Examples	orial
me Ins	Open %O	01.Basics	> /eloper Acrobat
× 1	Open Recent >	02.Digital	> 98dlp_template Ard
0	Sketchbook >	03.Analog	>
4	Examples >	04.Communication	>
	Close #W	05.Control	>
	Save #S	06.Sensors	>
	Save As ☆# S	07.Display	> us with the following
	Page Setup ①第 P	08.Strings	>
		09.USB	>
	Print # P	10.StarterKit_BasicKit	assignment/problem
		11.ArduinoISP	>
		Examples for any board	file:
		Adafruit Circuit Playground	> of this program:
		Bridge	>
		Esplora	<pre>ime spent on this as:</pre>
		Ethernet	> whom you worked whi
		Firmata	> u used in developing
		GSM	> discu in developing
		LiquidCrystal	> Autoscroll
		Robot Control	> Blink
	Phy	Robot Motor	> Cursor
	I IIJ	SD	> CustomCharacter
		Servo	> Display
		SpacebrewYun	> HelloWorld
	-	Stepper	> Scroll

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 again.

```
*/
```



```
// 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'11
  // 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*/}</pre>
  // 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++)</pre>
    // 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)
  ł
    // turn off the LED.
```

}

6

```
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:

• • •		/dev/cu.usbmodem14201	
l,			Send
9:43:11.985 - 9:43:11.985 - 9:43:12.096 - 9:43:12.096 - 9:43:12.096 - 9:43:12.132 -	Arduino demo for Physics 398DLP This file is /Users/g-gollin/Sup Date and time of compilation: Ja Time to do 1000 ADC reads and as Average ADC value was 327.77 (1. ADC RMS was 2.09 (10.2 millivolt	erWoodchuckie2/Physics_education/physics_398dlp/p398dlp_sp2 n 13 2022, 09:43:07 sociated calculations: 113 milliseconds. 60 volts)	022/microtut
Autoscroll	Show timestamp	Carriage return 📀 115200 baud 📀 C	ear output