

Lab 2

Published

1/3

Required lab tools: Arduino and Arduino IDE

[Step 1: Downloading Tools]

If you have not already loaded the software to your personal computer or laptop, the links are provided below:

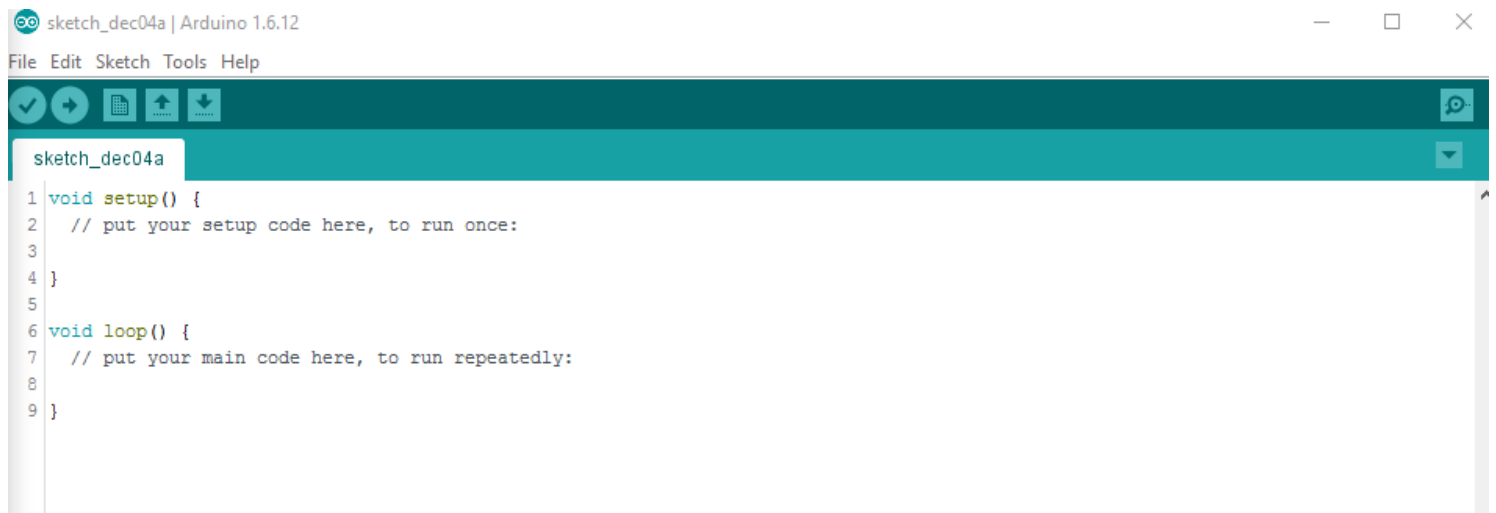
[Arduino IDE](https://www.arduino.cc/en/Main/Software) (<https://www.arduino.cc/en/Main/Software>)

Your TA may guide you through installing these programs onto your personal computer or laptop.

[Step 2: Familiarizing Arduino IDE]

1. Using your personal laptop or the Hydra machine (personal laptop is preferred), open the Arduino IDE.

You should get a screen like:



```
sketch_dec04a | Arduino 1.6.12
File Edit Sketch Tools Help
sketch_dec04a
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

2. In here, two function templates are provided. One called "setup" and one called "loop".

void setup() is called when your program starts running on the Arduino and is ran only once.

void loop() is called when your program starts running, but runs in a continuous loop as fast as possible.

3. Most of Arduino programming is just like C++, except there is no cin or cout since there is no monitor or console connected.

[Step 3: Programming Arduino]

1. Programming the Arduino is done through the Arduino IDE. However, unlike your laptop, the Arduino doesn't have a console. Instead, it communicates with your laptop or the Hydra machine via a "serial" connection. Your

console is provided under the "Serial Monitor" in the "Tools" menu.

2. You will interact with the Arduino through two ways: 1) The hardware buttons on the multifunction shield or 2) the serial monitor.

3. Download [lab2a.ino](#)

4. Open lab2a.ino using the Arduino IDE (your TA may show you how to do this).

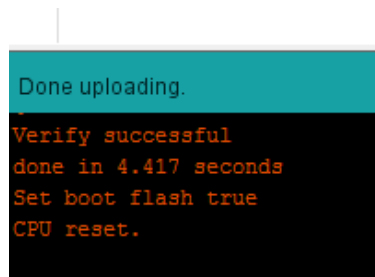
5. Your IDE should look like:



```
File Edit Sketch Tools Help
lab2a
1 void setup() {
2   // put your setup code here, to run once:
3   Serial.begin(115200);
4
5 }
6
7 void loop() {
8   // put your main code here, to run repeatedly:
9   String what_to_print = "Print this once every 1000 ms\n";
10  Serial.print(what_to_print);
11  delay(1000);
12 }
```

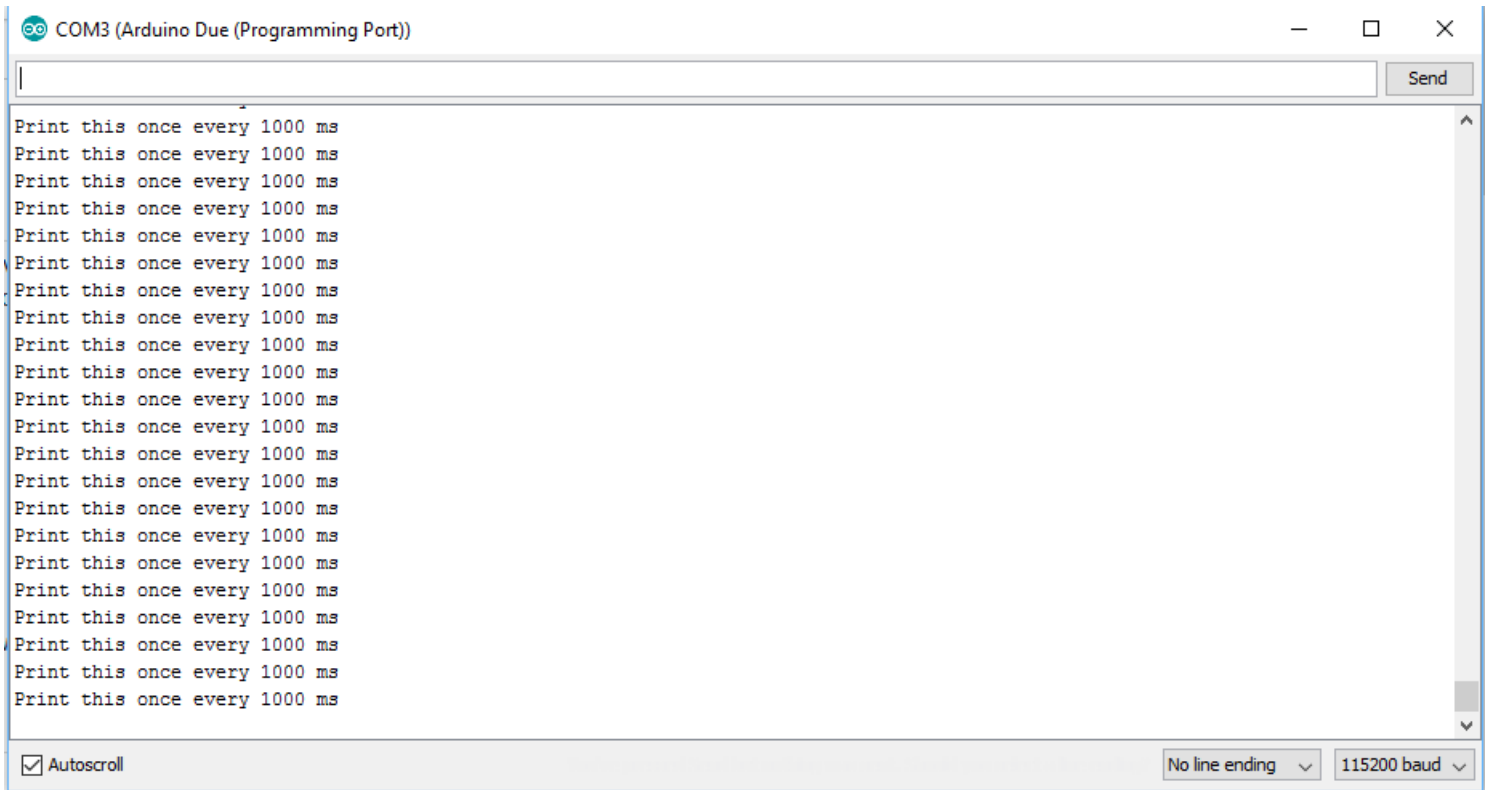
6. The checkmark on the toolbar verifies your file. Verify means that it simply checks for syntax errors.

7. The right arrow on the toolbar uploads your program to the Arduino. When you click this, it will automatically save your file, verify it, and then begin uploading it. It does take some time to upload your file, so wait until the status screen reads "CPU Reset."



```
Done uploading.
Verify successful
done in 4.417 seconds
Set boot flash true
CPU reset.
```

8. Open the serial monitor by clicking "Serial Monitor" under "Tools". Make sure your baud rate is 115200, or you will see weird characters. You should see something printed to the screen every second:



```
COM3 (Arduino Due (Programming Port))

Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms
Print this once every 1000 ms

 Autoscroll
No line ending
115200 baud
```

9. If you see this, you are finished with step 3. Otherwise, see your TA.

[Step 4: Problem Solving & Binary]

1. Create a new sketch called "lab2b"
2. Copy the file [lab2bstud.ino](#) into your sketch
3. There are three functions that you will need to write: `decimal_to_bin`, `decimal_to_hex`, and `decimal_to_oct`
4. `decimal_to_bin` converts a given integer into a binary string. You will need to use the `concat` function (`output += '0'` or `output += '1'`) to write a 0 or a 1 to the string. When your function finishes, output must contain 32 digits.
5. Upload and test your file. Using the serial monitor, type a number into the input box and press Enter. See if Bin: has the correct binary representation of the number. NOTE: Hex and Oct will be blank because you haven't written those functions, yet.
6. `decimal_to_hex` converts a given integer into a hexadecimal string. To show the versatility of binary, you can solve this problem by taking the same approach as `decimal_to_bin`, however, instead of 1 bit at a time, you will be taking 4 bits at a time.
7. Upload and test your file. Using the serial monitor, type a number into the input box and press Enter. See if Bin: and Hex: both have the correct representation of the number. NOTE: Oct will still be blank because you haven't written that function yet.
8. `decimal_to_oct` converts a given integer into an octal string. You will also be using binary to solve this problem by taking 3 bits at a time.

Criteria	Ratings			Pts
<p>Decimal To Binary</p> <p>-You must manually determine the binary value of a decimal by using a mask and logic operators.</p>	<p>40.0 pts Full Marks</p>	<p>20.0 pts Minor Binary Output Problems</p>	<p>0.0 pts No Marks</p>	<p>40.0 pts</p>
<p>Decimal To Hex</p> <p>-You must manually determine the binary value of a decimal by using a mask and logic operators.</p>	<p>35.0 pts Full Marks</p>	<p>18.0 pts Minor Hex Output Problems</p>	<p>0.0 pts No Marks</p>	<p>35.0 pts</p>
<p>Decimal To Octal</p> <p>-You must manually determine the binary value of a decimal by using a mask and logic operators.</p>	<p>25.0 pts Full Marks</p>	<p>13.0 pts Minor Octal Output Problems</p>	<p>0.0 pts No Marks</p>	<p>25.0 pts</p>
<p>Total Points: 100.0</p>				

