



Jan 31, 2025 > Lab 1: Introduction to Arduino - Due



Print



Settings

Lab 1: Introduction to Arduino - Due

Jan 31, 2025 11:59 PM

● Winter 2025 - ECSE-421-001 - Embedded Systems

This lab is to be done **individually**.

Academic Integrity

The following is offered with apologies to the vast majority of students who do their work honestly and take their university learning seriously:

Despite the university's questionable approach toward violations of academic integrity, your instructor takes such offences seriously, and has no tolerance for plagiarism or any other form of academic misconduct. Failure to respect these guidelines will result in you receiving a grade of **zero** on this lab assignment.

Acceptable collaboration between students, provided it is acknowledged explicitly in your report and code, might include:

1. discussing some aspect of the lab specifications in attempting to understand a particular point
2. discussing a problem you encountered while communicating with your Arduino
3. discussing high-level design choices for your pilot cockpit

Sharing of any Arduino code between students, re-using any code from a third party (e.g., open source), or making use of LLM-generated code is acceptable, **provided that you indicate this explicitly at the start of your report and (as comments) in the code**.

Unacceptable collaboration and violations of academic integrity include, but are not limited to:

1. including any code that was not your own (other than those elements of code provided or described in the specifications) and failing to indicate so
2. copying data, or any parts of answers to questions or generated graphs from other students
3. making use of LLM-generated responses to questions without acknowledging this use

If you are uncertain about any of these guidelines, please discuss with your instructor as soon as possible.

Objectives

The aim of this lab is to familiarize you with the basic operation of Arduino boards and their programming. By the end of this lab, you will:

- Install the required software to program an Arduino;
- Connect your Arduino to a computer;
- Connect and control various sensor modules to your Arduino via a breadboard.

Materials

You are expected to have your Arduino and sensor kit, along with the necessary USB cable to connect it to your computer.

Background

When talking about embedded systems development, we often take into account the following:

- An understanding of the problem we are trying to solve;
- A target platform on which to develop the application;
- A mechanism for programming the target platform.

The target platform is a term used loosely to refer to the actual hardware that performs the computation or control—the place where the application, or software, will ultimately reside and run. The target platform can be as simple as a single chip, or as complex as a feature-rich single board computer. Despite the differences in physical characteristics, the target platform's purpose is the same: to execute the software written by the developer.

Arduino is a cross-platform and open-source physical computing platform based on a simple microcontroller board. You can connect components to an Arduino to take in information from the outside world, and write code to control their interactions. The Arduino is often used for prototyping because of its accessibility, vast amounts of documentation and support from the user community, and the high level of abstraction it is designed to provide.

Prelab

1. Download and install the Arduino software and IDE for your OS from <https://www.arduino.cc/en/software/>

2. Launch the Arduino application, which loads the Arduino IDE.
3. Connect your Arduino using its USB cable.
4. Configure the IDE for your board by selecting the appropriate Arduino type from the **Tools -> Boards** menu.
5. Verify that your board is connected properly by checking the **Tools -> Serial Port** menu:
 1. On Windows, you should see a “COMx” port, where x is an integer.
 2. On OS X and Linux, you should see a port with the name /dev/ttyUSB0, /dev/tty.usbmodem, /dev/ttyACM0, or similar.
6. Test the board by loading and running your first program. The Arduino IDE is equipped with a series of examples for demonstration and testing. The simplest of these, which just blinks an LED, can be found from **File -> Examples -> 01. Basics -> Blink**.
7. A new window with the source code of the program should load. This window is called a “sketch” in Arduino jargon. Briefly observe the code and supporting comments. The sketch is written in C, but the Arduino takes care of a lot of the nuts and bolts behind the scenes, wrapping your sketch into an encompassing code. Your codes must necessarily contain the two structures below:

```
1 void setup() {  
2 // setup code here, configure input/output pins, etc.  
3 }  
4  
5 void loop() {  
6 // put action code here, to be repeated indefinitely  
7 }
```

Arduino

8. Find and click the **Upload** button on the toolbar. The Arduino will take some time to compile the program. After a few seconds, you should see the IDE say “done uploading” on the console output at the bottom of the program. You should now see an LED blinking on your board!

Lab Activities

We will now connect a variety of sensors your Arduino. Before doing so, ensure that your Arduino is disconnected from power. You can use the breadboard to organize the power distribution to each sensor.

Exercise 1: Working with an RGB LED

Your sensor kit comes with two RGB LED modules. You may use either of them (or both!). Plug the module into your breadboard, then connect each connector to an appropriate pin on your Arduino board ("—" to GND, then each "R", "G", and "B" to a digital pin).

As you will see in your readings, it is important to add a resistor in series to prevent the LED from frying; fortunately, this resistor is already built into the sensor module.

Task: Using the blink sketch as an example, blink your external LED in white color using `delay (amount)`, where `amount` is measured in milliseconds.

Task: Modify your previous code to make the LED cycle through 6 distinct colors every 500ms.

Remember to replace the address of the Arduino's on-board LED, ***LED_BUILTIN***, referenced in the code, with the "slot" numbers in which your LED's "R", "G", and "B" pins are connected, to reflect the addresses of your external LED's pins.

Exercise 2: Using the Rotary Encoder, the Buzzer, and the Microphone

The sensor kit contains a rotary encoder. Connect it to your Arduino board. A rotary encoder works differently from a potentiometer. You can read more about them here: https://en.wikipedia.org/wiki/Incremental_encoder. Pins "CLK" and "DT" act as the signal pins. You may need to look at their raw outputs first.

- In the `set up ()` function of the Arduino, configure the "CLK" and "DT", and "SW" pins as inputs <https://www.arduino.cc/en/Reference/pinMode>

Task: Write a program that outputs a counter on the serial monitor. The counter is initialized at 0 and is incremented by 1 for each clockwise increment of the rotary encoder. Similarly, it is decremented by 1 for each counterclockwise increment. The counter should be reset to 0 when the encoder is pressed down like a button.

In addition to simple binary values of the push-button switch, we can measure continuous values such as the temperature of a room, or the value of a knob. The Arduino Uno has six analog inputs, which convert an external voltage from an analog signal module, between 0 and 5 V, to a digital value, expressed to 10 bits of precision. The microphone is such an analog signal module.

Now, connect the passive buzzer and the microphone to your Arduino. The passive buzzer is the one without the sticker attached to it. In the `setup()` function of the Arduino, configure the buzzer pin as an output and the microphone pin as an input.

Task: Use the rotary encoder to select the buzzing frequency of the buzzer. Using the serial plotter and the microphone, find the approximate frequency at which the buzzer appears to be the loudest from the microphone's point of view.

Task: Repeat the previous task with the other microphone module. How does the sensitivity change?

Question: What is the smallest value of change in voltage that the analog inputs can report? Explain your reasoning.

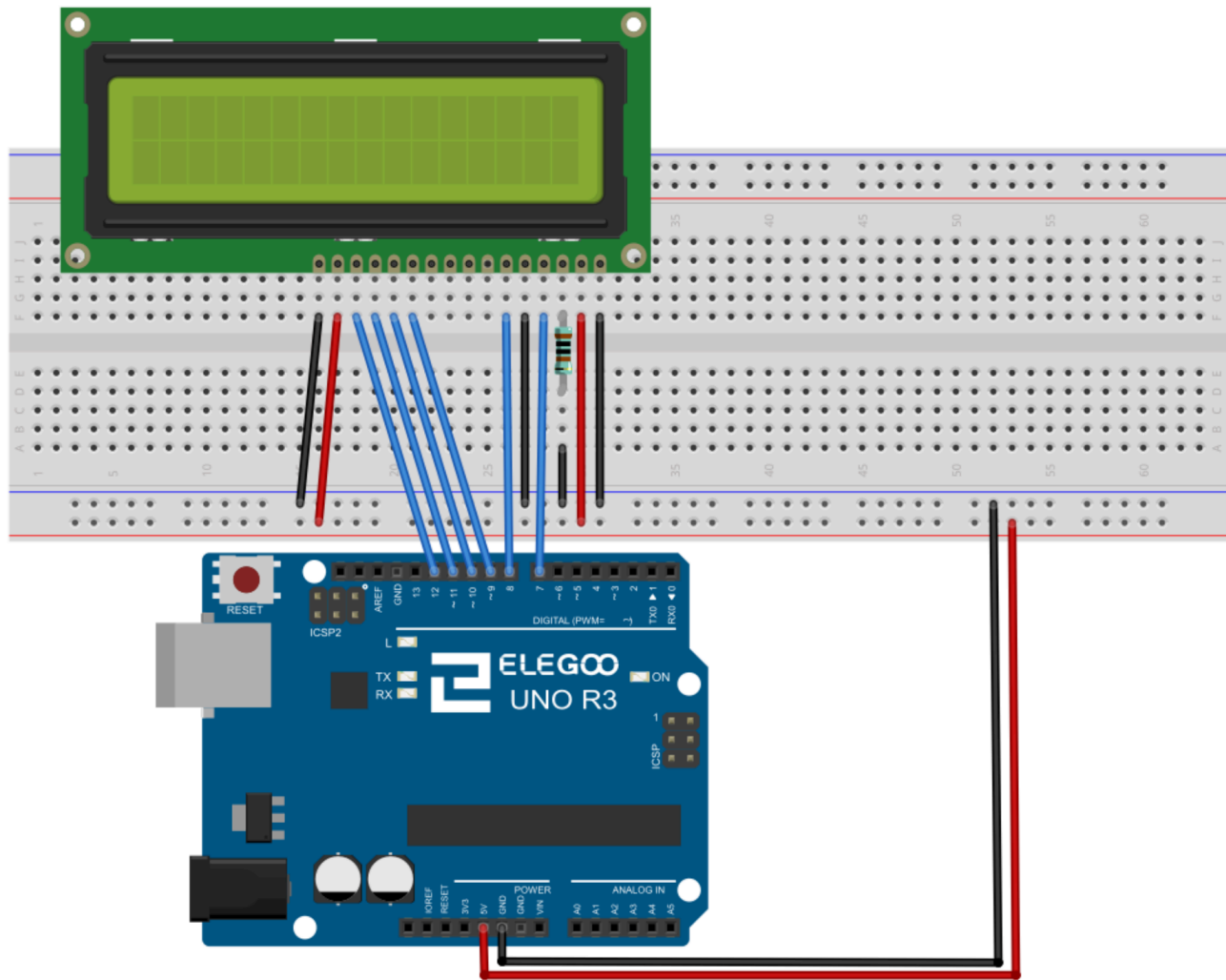
Exercise 3: The Weather

We all know that one of the easy conversations to have when you can't think of a good conversation topic is to talk about the weather. To impress your crush, you want to add to this conversation, so you decide to create a device to monitor the weather to the best of your abilities.

For the next part of the lab, connect the Temperature and Humidity sensor to your Arduino board. To obtain data from this sensor, you will have to install the DHT library. This can be accomplished through the library manager in the Arduino IDE, accessible from the **Sketch -> Include Library -> Manage Libraries** menu. Note that this sensor has a maximum sampling period of 2 seconds.

Task: Write a program that displays the temperature and humidity in the serial monitor.

You will now connect the LCD to your Arduino. Below is a wiring diagram to help you. Note the resistor connecting to the V0 pin. This adjusts the contrast ratio. You can choose a resistor that leads to the contrast ratio of your choice. You may also use a potentiometer if you have one.



Source: *Elegoo sensor kit V2 instructions*

Install the Liquid Crystal library to use the LCD display. Below is some code you can use to get started.

```
1  #include<LiquidCrystal.h>
2
3  LiquidCrystal lcd(7, 8, 9, 10, 11,12);
4
5  void setup() {
6
7      lcd.begin(16, 2);  // Set up the LCD's number of columns and rows:
8
9      lcd.print("hello, world!");    // Print a message to the LCD.
10 }
11 void loop() {
12     lcd.setCursor(0, 1);    // Set the position of the cursor
13     lcd.print(millis() / 1000);    // print the number of seconds since reset:
14 }
```

Arduino

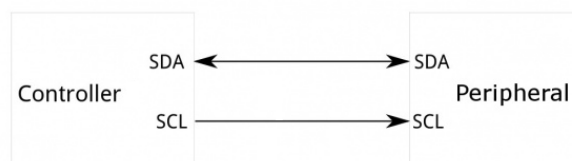
Task: Repeat the previous task but display the temperature and humidity on the LCD display.

Exercise 4: You Are a Pilot

Raise your hand with your fingers outstretched. You will notice that your fingertips are moving very slightly. These movements are normal and called tremor. The study of this movement is important in clinical and research settings, e.g., to measure its impact on fine motor skills or limb dysfunction. How can we measure fingertip tremor? One solution is to purchase an expensive instrument specifically designed for this purpose, but a more cost-effective approach is to use an Arduino board to collect data from a tri-axial accelerometer, a sensor that measures accelerations in three orthogonal directions.

You can find the specifications for your accelerometer, including its sensitivity and resolution, [here](https://wiki.seeedstudio.com/Sensor_accelerometer/). Sensitivity defines the rate at which the sensor converts mechanical movement into an electrical signal, usually expressed in mV/g. Resolution describes the smallest detectable change in acceleration values, specified in bits. You will notice that there are a variety of accelerometers with different resolution and sensitivity; a table can be found at https://wiki.seeedstudio.com/Sensor_accelerometer/ [possibly to be updated]

Just as we did for the previous tasks, we need to write a sketch to measure acceleration. The accelerometer and Arduino communicate via a protocol known as I2C. The I2C bus has two physical wires: *SDA* (data line) and *SCL* (clock line) that connect two components: the controller (Arduino), which drives the clock, and the peripheral (accelerometer). Data are transferred in 8-bit packets (bytes) every clock pulse. After data are transferred, an “acknowledgement” bit is sent from the controller to the peripheral as confirmation.



Working with the accelerometer

Connect the accelerometer module to your Arduino board, connecting the SDA and SCL pins to the corresponding ones on the board. To program the accelerometer, you will have to install the FastIMU library. This can be accomplished through the library manager in the Arduino IDE, accessible from the **Sketch -> Include Library -> Manage Libraries** menu. Alternatively, you can download the FastIMU library and include it inside the Arduino libraries folder on your system.

Use the following starter code to print raw values of the acceleration in each axis to the serial monitor. Check your sensor board and make sure the model matches the one defined in the code. Once you’ve made the connections, upload your sketch.

```

1  #include "FastIMU.h"
2  #include <Wire.h>
3
4  #define IMU_ADDRESS 0x68    //Change to the address of the IMU
5  MPU6500 IMU;               //Change to the name of any supported IMU!
6
7
8  calData calib = { 0 };    //Calibration data
9  AccelData accelData;      //Sensor data
10 GyroData gyroData;
11
12 void setup() {
13     Wire.begin();
14     Wire.setClock(400000); //400khz clock
15     Serial.begin(115200);
16     while (!Serial) {
17         ;
18     }
19
20     int err = IMU.init(calib, IMU_ADDRESS);
21     if (err != 0) {
22         Serial.print("Error initializing IMU: ");
23         Serial.println(err);
24         while (true) {
25             ;
26         }
27     }
28 }
29
30
31 void loop() {
32     IMU.update();
33     IMU.getAccel(&accelData);
34     Serial.print(accelData.accelX);
35     Serial.print("\t");
36     Serial.print(accelData.accelY);
37     Serial.print("\t");
38     Serial.print(accelData.accelZ);
39     Serial.print("\t");
40     IMU.getGyro(&gyroData);
41     Serial.print(gyroData.gyroX);
42     Serial.print("\t");
43     Serial.print(gyroData.gyroY);
44     Serial.print("\t");
45     Serial.print(gyroData.gyroZ);
46     Serial.println();
47     delay(50);
48 }

```

Task: Open the serial plotter from the **Tools -> Serial Plotter** menu. Shake the sensor around in each axis. Do you see the values changing? What do they mean?

In order to make meaningful inference from the data, we must first calibrate the accelerometer. Calibration involves testing the 6 positive and negative cases for each axis. When you've finished, the last serial output value will contain the minima and maxima for each axis. Using these values you can apply the following two-point calibration formula to calibrate each axis:

$$\text{CorrectedValue} = \frac{(\text{RawValue} - \text{RawLow}) \times \text{ReferenceRange}}{\text{RawRange}} + \text{ReferenceLow}$$

Remember that we are using gravity as our reference, so RawLow and RawHigh correspond to the sensor's measurements of +/-1g.

Once the accelerometer has been calibrated, rewrite your code in the `loop()` function so that it will print out x, y, and z data, along with the timestamp, in an organized, row-wise fashion. Move the accelerometer module back and forth and observe changes in acceleration and deceleration. Start initially in one plane, and gradually make movements triaxially.

With the serial monitor window active, record and save your accelerometer movements, ideally, involving some repetitive patterns of motion and some abrupt movements, to a .csv file (e.g., by copy/pasting the serial monitor data). Make sure you have at least a few seconds of data.

Task: Plot these data using any tool of your choice (Excel, Python, MATLAB, etc.). Use these values to estimate the displacement of the accelerometer and plot displacement on a separate graph.

As you will learn from your upcoming reading, the values reported by a sensor exhibit various forms of noise, which could be caused by the lack of accuracy inherent to the sensor, white noise from the environment, and quantization. One method of dealing with noise is to estimate an average measurement from multiple sensors, in what is known as sensor fusion. However, since we only have one accelerometer, we'll use an alternative approach: filtering the signal. Those of you who have taken a course in signal processing may know of methods to "smoothen" the signal, for example, making use of a moving average window by taking an unweighted mean of the previous N samples, given at time t by:

$$MA_t = \frac{x_t + x_{t-1} + x_{t-2} + \cdots + x_{N-(t-1)}}{N}$$

Task: Choose a suitable value of N and write a moving average filter. Run your data through it. Plot this against the raw (unfiltered) signal. What are the benefits/drawbacks of having a high N compared to a low N ?

Task: With your accelerometer and gyroscope calibrated, connect the LCD to your board. Design an airplane instrument panel. Display the yaw, pitch, and roll of your plane, as well as the elevation (estimated through the accelerometer readings).

Task: Flesh out your cockpit. Use the buzzer as a [stall](#) warning for your plane. If the pitch goes above a certain threshold, the buzzer should be heard.

Recap: So far, we've worked with various digital and analog I/O and put together some "mini" projects to familiarize you with the core functionality of the Arduino. Take some time to go over the Language Reference page.

Functions are largely Arduino-specific, and you have used some of these for the exercises in this lab. **Variables** are mostly C syntax, but there are some Arduino-defined commands such as HIGH and LOW. The **Structure** panel contains basic C syntax, aside from the `setup()` and `loop()` pieces that define an Arduino sketch.

What to turn in


Your submission, through FeedbackFruits on myCourses (link will be provided shortly), should consist of the following:

- all the required calculations, answers, and outputs in response to the identified Questions and Tasks
- code for each task—make sure this is adequately commented so that the logic can be understood by the reviewers
- URL(s) to one or more **brief** video(s) that demonstrate all five of the exercises described in this laboratory specification

Guidelines for video submission:

- Keep the video(s) short and to the point; your peer-reviewers will not want to spend more than a few minutes watching your video submissions
- What is happening in the video(s) should be clearly understood by the reviewers;
- Check your video(s) to ensure nothing is missing! Test the URL to ensure that the video is working.

- We recommend that you upload the video(s) to YouTube, but you may use any online platform available to you, so as long as it is accessible to reviewers without the need for download


 Due January 31 at 11:59 PM



100 % 1 of 1 topics complete

Lab 1: Introduction to Arduino (ECSE-421-001, Winter 2024)



 External Learning Tool

Tasks

Add a task...