**Deadline: Jan 22, 2023 – 11:30 pm**

**Section 1. Getting Started with PEEB**

This kit includes our fully assembled **portable experimenter engineering board (PEEB)**, a USB cable, and an **Arduino**. The Arduino hardware and software are open source. The Arduino software, known as the Integrated Development Environment (IDE), is free. You can download it from [www.arduino.cc](http://www.arduino.cc). The PEEB uses the Arduino IDE and Arduino pinout (see Figure 01). The pinout refers to the pins or contacts that connect an electrical device or connector. It describes the functions of transmitted signals and the circuit input/output (I/O) requirements.

* 1. **Arduino Pinout Guide**

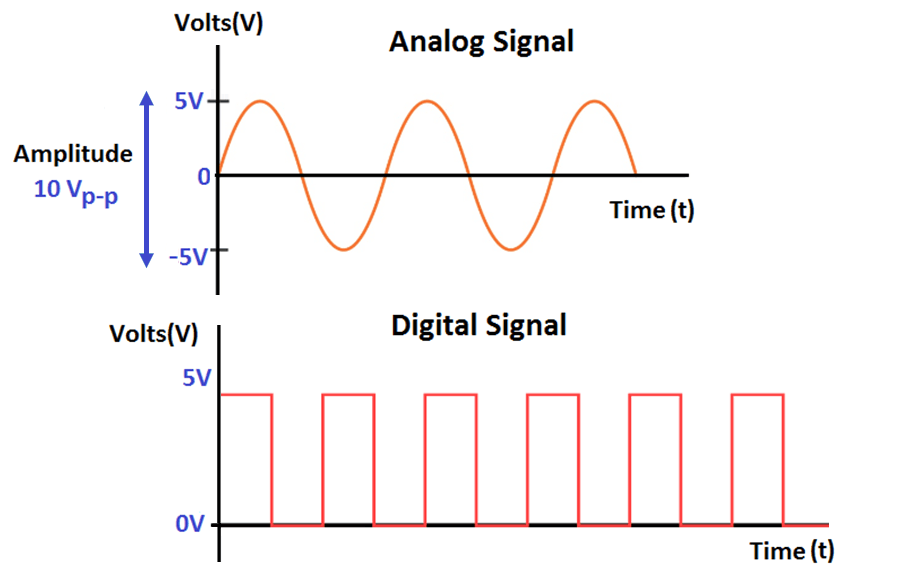
Shape

Description automatically generated A picture containing schematic

Description automatically generated

**Figure 1.** Board pinout

The Arduino Uno has analog and digital pins. An Analog signal is a continuous signal in which one time-varying quantity represents another time-based variable; for example, human voice, thermometer, and pressure are some examples of analog signals. A digital signal is a signal that is used to represent data as a sequence of separate values at any point in time. Phone signals, digital television, and computer use digital signals. Figure X describes one example of two different signals, analog and digital.



**Figure 2.** Analog and digital signal

The Arduino Uno has **6 analog pins**, which utilize ADC (Analog to Digital converter). ADC stands for Analog to Digital Converter. ADC is an electronic circuit used to convert analog signals into digital signals. This digital representation of analog signals allows the processor (which is a digital device) to measure the analog signal and use it throughout its operation.

Pins 0-13 of the Arduino Uno serve as **13 digital input/output pins**. Digital pins on the Arduino are designed to be configured as inputs or outputs according to the user's needs. Digital pins are either on or off. When ON they are in a HIGH voltage state of 5V and when OFF they are in a LOW voltage state of 0V. Each pin can provide/sink up to 40 mA max. But the recommended current is 20 mA. The maximum current provided (or sank) from all pins is 200mA.

**5v** and **3v3** provide regulated 5 and 3.3v to power external components according to manufacturer specifications.

You can find **3 GND** pins, which are all interconnected. Ground (GND), in the context of electronics, is the reference point for all signals or a common path in an electrical circuit where all of the voltages can be measured from. This is also called the common drain since the voltage measurement along it is zero. The GND pins are used to close the electrical circuit and provide a common logic reference level throughout your circuit. Always make sure that all GNDs (of the Arduino, peripherals and components) are connected to one another and have a common ground.

* 1. **Power Supply**

Power is fundamental. It is even better when you have the opportunity to get different values. This kit includes an adjustable power supply regulator module to offer a wide range of power values with a range from 0.5V – 30V; working current 4A; and power 35W.

A close-up of a circuit board

Description automatically generated with medium confidence

**Figure 2.** Adjustable Buck Boost Converter

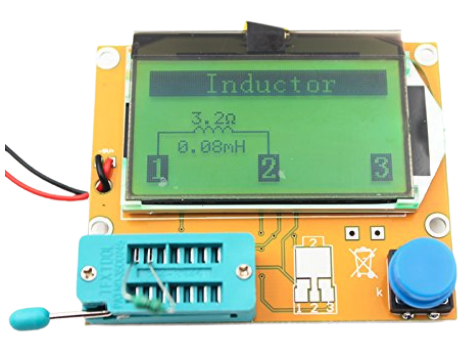
Diagram

Description automatically generated

**Figure 3.** Module description

* 1. **Component Tester**

It's easy to make mistakes with several components due to their codes and colors. In order to reduce this problem, the kit includes a multifunction meter kit. This kit performs automatic detection of NPN and PNP transistors, n-channel and p-channel MOSFET, diode (including double diode), thyristor, transistor, resistor and capacitor, and other components.



**Figure 4.** Multifunction meter kit

Section 2. Developing your electronic skills

Let’s put into practice your electronic skills. The PEEB will help you with that. In this activity, please make sure you have the following items:

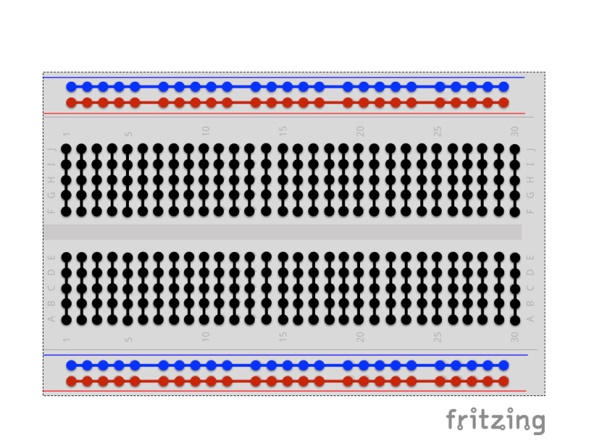
* 2x jumper wires (male to female)
* 1x resistor (any value above 220 Ohms). **DO NOT USE ANY VALUE BELOW 220 OHMS!**
* Multimeter

**2.1 Experiment with +5V voltage source:**

Step 1: Let’s get started! First, mate the PEEB board on top of the Arduino, aligning all the pins on the shield to the female headers on the Arduino. Your Arduino can run on an external USB power source, but you need to use your computer to upload your code.

Step 2: Connect a resistor to a breadboard.

**What is a breadboard?** A breadboard is a small plastic slab drilled with holes at intervals of 0.1”. Hidden inside the plastic are little spring clips which establish connections between the wires and components, which you push into the holes.

 A picture containing diagram

Description automatically generated

**Figure 5.** Breadboard setup

**Step 3:** Connect each leg to Con1 and Con2 on the PEEB circuit board.

A picture containing text, different, various, several

Description automatically generated

**Figure 6.** Resistor connection

The internal connection is equivalent to the following circuit where +5V connects to the resistor, which is connected to the **LED** **5**:

Diagram

Description automatically generated

+5V

GND

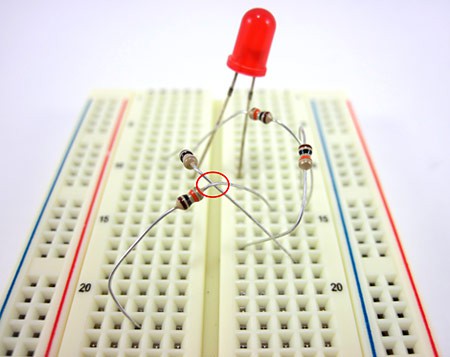
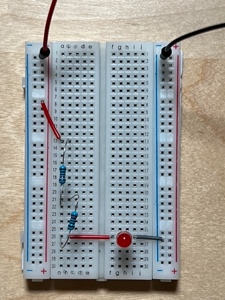
**Figure 7.** Equivalent circuit from Figure 6

**What is an LED?** The symbol below represents a light-emitting diode (LED) which is a semiconductor device that emits light when current flows through it. The LED usually needs between 2 and 3 volts, and it depends on the color and type of LED.

A screenshot of a video game

Description automatically generated with medium confidence

It is important to wire a circuit neatly right from the beginning because it will minimize mistakes. Figure 6 describes a few examples of messy wiring and good wiring. Also, keep the power off while you are wiring your circuit.

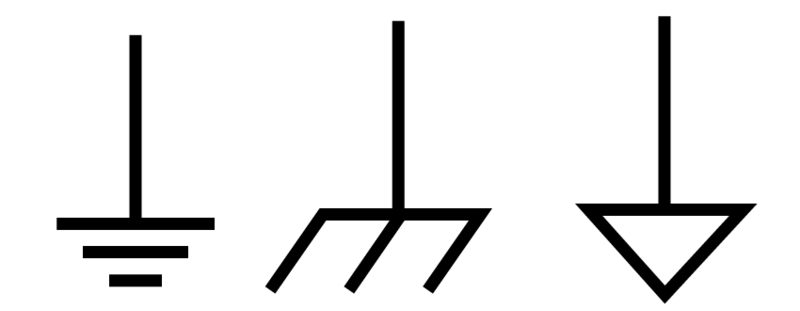
 

**Figure 6.** (left) Messy connections and (right) neatly wired circuit

Here are some additional tips for you:

* Use connections as short as possible. For this studio, you can use the wire kit provided into the box.
* If you can connect components such as resistors directly, do so.
* Use the longer strings of holes as power and **ground**.

**What is a ground connection (GND)?** In electrical and electronics engineering, we define a point in a circuit as a reference point. This reference is point is known as ground (0V). The ground is often indicated by one of the symbols below. Grounds are important avoid parasitic voltages that can interfere in your circuit or can even damage them. GND can also avoid an electric shock.



**Step 4:**  Measure the voltage across the resistor using the **multimeter**. (Before measuring it, calculate how many volts you expect to measure across the resistor. Then, compare with your measured value)

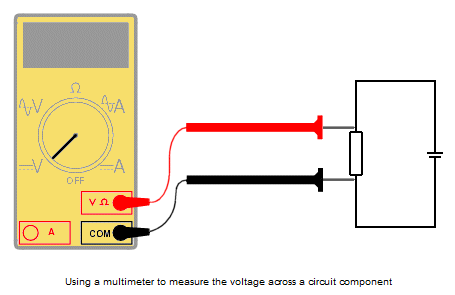
What is a **multimeter?** A **multimeter** is an electrical tool used to measure electricity. Multimeters are supplied with two leads, one red and one black. Labels and options differ from one meter to the next, but they often come with the following parts:

* On/Off Switch: Some multimeters have an on/off button or switch. If yours does, turn it off when you’re done testing. Other models have an auto-off feature to save battery.
* Display: This can be digital or analog. A digital multimeter shows a number when you measure. An analog display has a meter and an indicator pointing to a number.
* Meter Pointer (analog only): A straight line, often red, moves to show the electricity property you selected.
* Selector Knob or Button: Turn this knob or press the button to select your unit of measure. The choices on display match the choices on the knob.
* V: The V stands for voltage.
* A: The A stands for amperes or amps.
* Ohms or Ω: This curvy upside-down letter U is the Greek letter omega. It’s the symbol for ohms.
* M: Milliamps are tiny fractions of amperes. Each one is 1/1000 of an amp.

**How to measure the voltage across the resistor?**

The voltage across a component measures the difference in electrical potential from one side of the component to the other. First, set up the knob as a voltmeter (V) with a straight line next to it. Then, you must connect the multimeter as shown:

A picture containing diagram

Description automatically generated

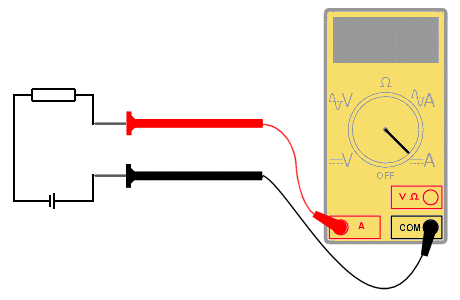
**Figure 8.** Current measurement with multimeter

**Step 5:**  Measure the current across the resistor using the multimeter.

Current is a measure of the rate of flow of electrons through the circuit. To measure the flow of current, the circuit must be broken, and the meter must be placed in the circuit such that the current flow goes through it.

First, you need to prepare to measure the current through an element by disconnecting one leg from the preceding circuit, leaving the rest of the circuit connected. Plug the black probe into the “COM” port. There will likely be two options for the red probe, a higher amperage rating, and a lower either milli- or micro-amperage rating. If unsure, start with the higher option and move it to the lower one for a more accurate reading. Then, turn the dial to the closest amperage option to what you will measure. If you have no idea, you’ll want to start higher and move lower as you go. After that, insert the two leads in the circuit; it acts approximately as a wire.

A picture containing diagram

Description automatically generated 

**Figure 9.** Voltage measurement with a multimeter

**2.2. Experiment with +3V3 voltage source:**

Repeat the previous steps using Con3 and Con4.

Section 3. Developing your programming skills

3.1. The Blink Test

Let’s test your PEEB with a simple test called “The Blink Test.” In this test, we will reprogram the PEEB with a code to active LED 1, LED 2, and LED 3. After that, change the rate at which it blinks.

A screenshot of a video game

Description automatically generated with medium confidence

**Figure 10.** LED1 on PEEB

**Step 1:** Connect the PEEB to your computer using the blue USB cable.

**Step 2:** On your computer, run the Arduino IDE. Select File → New. Save your file as Blinking\_LED and click OK.

Graphical user interface, text, application

Description automatically generated

**Figure 11.** Arduino IDE interface

Important: Make sure your computer can recognize your device.

Go to **Tools** → **Port** → *Find the respective port*

In the default template, you see:

void setup() {

// put your setup code here, to run once:

}

This is a line where you explain basic information about your code to the **compiler**. A compiler is a special program in every computer that translates a programming language's source code into machine code. This line has to be at the beginning of your code all the time.

The void setup is the first function to be executed in the sketch and it is executed only once. It usually contains statements that set the pin modes on the Arduino to OUTPUT and INPUT, example: pinMode (12, OUTPUT); pinMode(11, INPUT); Or to start the serial monitor example: serial.

The symbols **{** and **}** define the limits of the function void setup.

We need to test three different LEDs (1, 2, and 3) for this test. Let’s begin with LED 1. If you take a closer look at your board, you will that a small white address D1 is printed alongside the LED 1. D1 indicates that LED 1 is connected to pin 1. If you recall from **section 1.1**, the programmable input and output pins follow the Arduino pinout, numbered from 0 to 13.

With the opening brace, you will type your very first block of code. Type the following code:

pinMode(1, OUTPUT);

pinMode tells PEEB how to setup a specific pin. You can setup a pin as input or output, but you need to make it very specific in your code. How? Simply by saying which pin you intend to use and what configuration you expect from that pin. In our case, we want to set pin 1 to blink LED1. So, (1, OUTPUT) .

We are done with void setup. Let’s go to the next step.

Before you move to the next step, make sure you are including useful **comments** in your code. The Arduino language is merely a set of C/C++ functions. There are three ways to comment your code:

* **(Single lines)** You can put two forward slashes in front of the line //
* **(Multi-line)** By using both a forward slash & asterisk between any amount of code /\* … \*/

You will see a different function named as void loop(). In this function, you will explain your code. Here is the space where you will change the world! Let’s remember our problem first. We need to blink a specific LED. At this moment, don’t worry about time or blinking rate. We want to make it work. First, you need to be familiar with two Arduino functions digitalWrite and digitalRead.

digitalWrite Write a HIGH or a LOW value to a digital pin. If the pin has been configured as an OUTPUT with pinMode() , its voltage will be set to the corresponding value: 5V for HIGH , 0V (ground) for LOW. We often refer to HIGH and LOW to indicate ON and OFF, respectively.

digitalRead Reads the value from a specified digital pin, either HIGH or LOW.

**Hint:** Feel free to go to <https://www.arduino.cc/reference/en/> to find more information about other useful functions.

We need to set pin 1 as HIGH. Then, after a few times, we set it as LOW. We can use the function delay to pause the program for the amount of time (in milliseconds) specified as parameter. (**Hint: There are 1000 milliseconds in a second**) For example, you need to set your delay as 2000 if you want to pause your program for two seconds.

This is all you need to complete your very first code. Let’s do it.

**Step 1.** Set Pin 1 as HIGH.

digitalWrite(1, HIGH); // turn the LED on (HIGH is the voltage level)

**Step 2.** Let’s wait for 3 seconds.

delay(3000); // wait for three seconds

**Step 3.** Set Pin 1 as LOW.

digitalWrite(1, LOW); // turn the LED off by making the voltage LOW

**Step 4.** Let’s wait for 3 seconds.

delay(3000); // wait for three seconds

**Step 5.** Repeat the entire process! It will happen automatically since we are using the function void loop.

**Your turn.** Repeat the same process for LED2 and LED3 using a blinking rate of 0.5 seconds and 0.2 seconds, respectively.

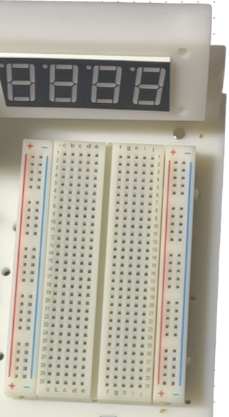
3.2. The Sound Test

In this test, we will use sound instead of light. The process is simple. Follow the steps described in “The Blink Test.” But you must switch your output from the LED to the speaker. You can find the respective pin connected to the speaker printed on the board alongside the speaker (D6).

Assignment

Complete each challenge described below. Then, report your solutions in a single document via Brightspace as a **team**. Show your work by describing your process (including equations and pictures). Paste and copy your code into the solution document. Do not take a screenshot of your code.

NOTE: Use the two wires connected to the power supply voltage to solve Challenge 1.

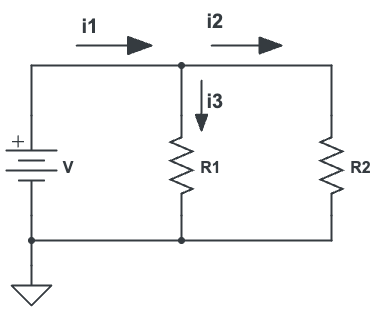


**+V**

**GND**

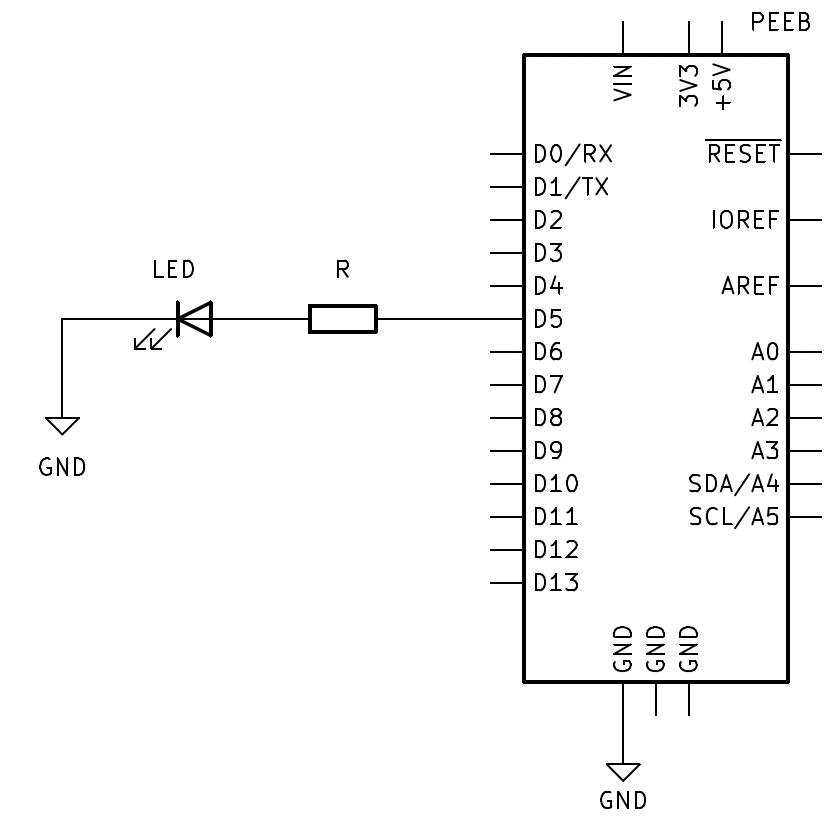
**Challenge 1.** Reproduce the following circuits, **a** and **b**, using resistors of your choice and breadboard. Then, follow the steps below.

Diagram, schematic

Description automatically generated 

1. (b)
2. Construct the series circuit shown in the circuit (a). For this experiment, R1 and R2 will be resistive elements of your choice. Set the power supply voltage to an arbitrary value from 4V to 7V.
3. Calculate the total current in the circuit. Then, measure the current in your circuit. Verify if the values match or not. Explain your results.
4. Measure the voltages across R1 and R2.
5. Verify if these voltages satisfy KVL.
6. Construct the parallel circuit in circuit (b). For this experiment, R1 and R2 will be resistive elements of your choice. Set the power supply voltage to an arbitrary value from 4V to 7V.
7. Measure the currents i1, i2, and i3. (***Hint:*** *Think of which connection you need to break in each case to use the ammeter*)
8. Measure the voltage across R1 and R2.
9. Verify if these currents and voltages satisfy KCL and parallel circuit laws, respectively.
10. Show your equations and results using measured values and calculated values.

**Challenge 2.** Using the circuit configuration below, build a Morse code generator to send an S.O.S distress signal using light. **Components needed: 1x LED and 1x resistor from the resistor set provided to you.**



**Challenge 3.** Using the previous setup, build a Morse code generator to send an S.O.S distress signal using sound.

**Deliverable:** SOS Project – 1 submission per team

1. Memo with solutions from each challenge described in the assignment

* Include a copy of your code into the document (copy/paste). Do not take a screenshot.
* Follow memo guidelines
* Submit your assignment as a PDF
* Due before Sunday (01/22/2023) at 11:30pm on Brightspace

**Grading Rubric**

|  |  |  |
| --- | --- | --- |
| Description | Expectation | Met? |
| General | All instructions and formatting standards were followed. | /1 |
| Document is consistent throughout and reads as a professional document. | /1 |
| Formatting | Complete memo header following format guidelines | /2 |
| Well-written unique subject line. | /2 |
| Figure labels formatted correctly | /2 |
| Results | KVL and KCL calculations are correct. | /5 |
| Current and voltage values are correct. | /5 |
| Units are correct. | /2 |
| Challenge 1 correctly answered with complete sentences. | /10 |
| Code | | |
| Challenge 2 | Functional code | /10 |
| Correct output | /10 |
| Commented properly | /5 |
| Organization is clear and professional | /5 |
| Code inserted properly into the document according to the instructions in the assignment | /5 |
| Challenge 3 | Functional code | /10 |
| Correct output | /10 |
| Commented properly | /5 |
| Organization is clear and professional | /5 |
| Code inserted properly into the document according to the instructions in the assignment | /5 |
| Total score = | | /100 |