Lab 4: Arduino

Reference Resource (*optional and available in the GWC lab*):
Practical Electronics for Inventors, Paul Scherz and Simon Monk, 2016, ISBN 978-1-25-958754-2

Arduino is Open-Source Code and Content and Support are available via the website:
https://www.arduino.cc/

TinkerCAD Tutorials for Simulation are available via the website:
https://www.tinkercad.com/learn/circuits/

Lab Overview:

In preparing for the final design project, you'll need to get familiar with using a microcontroller and LCD display for automation. Sparky Solar really wants to be a lead innovator in their field, which is why they've come to ASU (tops in innovation). Automation is a cutting-edge trend that will distinguish their products and services from their competitors. The use of an Arduino (or equivalent) board system will allow you to build automation into the final design project.

Automation is a huge field of advancement today to yield higher safety and awareness of everyday objects. This can exist in terms of digital or analog applications; where digital means that a value is either 0 or 1, mimicking the theory of digital logic seen in modern microcontrollers. Analog sensors however have a range of values that can indicate different situations for different values. In this lab, you will learn how to incorporate digital and analog sensors and control an LCD display. An important outcome of this lab is to familiarize yourself with the Arduino hardware and sensor uses, and see the potential of future automation of everyday applications.

Inventory List:

The following is a list of available inventory for this project:

Part	Details
Resistors	1Ω, 5Ω, 10Ω, 100Ω, 220Ω, 330Ω, 1kΩ, 2kΩ, 5kΩ, 10kΩ, 20kΩ, 100kΩ, 200kΩ, 1MΩ
Alligator clips	Connectors
Assorted jumper wire	Red, black, white, green, blue, orange
Potentiometer	10K
Slide Switch	Breadboard-friendly SPDT Slide Switch
DC Supply	LPS 161A DC Supply
Breadboard	830 Tie Points
Digital Multimeter	HP or Agilent 34401A
UNO Controller Board	Elegoo
USB Cable	Controller board to USB
9V Battery with Snap-On Connector Clip	9V
LCD Display	LCD1602 Module with Pin Header
Photoresistor	Photocell (1528-2141-ND)

Lab Deliverables

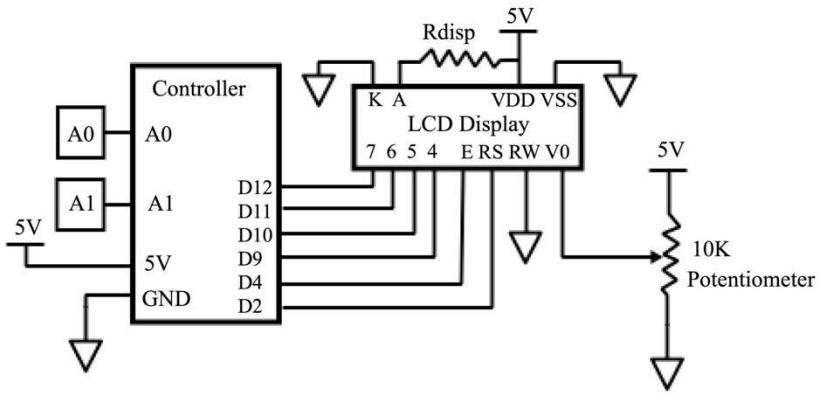
1. As you work through the lab, complete the required work on the Data sheet.
2. Submit your data sheet online (Individual Submission).

***** IMPORTANT NOTE REGARDING THE ARDUINO BOARDS*****

Never connect the USB cable to a computer and a battery or power supply to the board at the same time or you risk sending a charge in through the USB port and damaging the computer. Keep any power supplies separate and distinct from the 5V or 3V pin of the Arduino.

Lab Instructions:

The final design project will require the use of an Arduino (or equivalent) type board and an LCD display as shown:



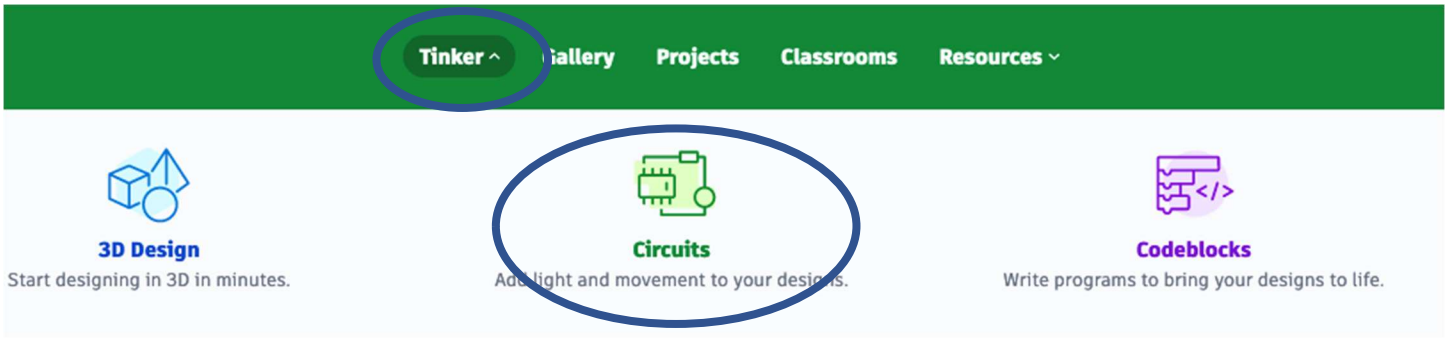
In this lab you will learn how to build an Arduino circuit with analog and digital inputs and program the controller to perform specific tasks.

If you'd like additional information to better understand how the Arduino works for this lab, you can find a set of slides here and a video tutorial on the lab Canvas page.

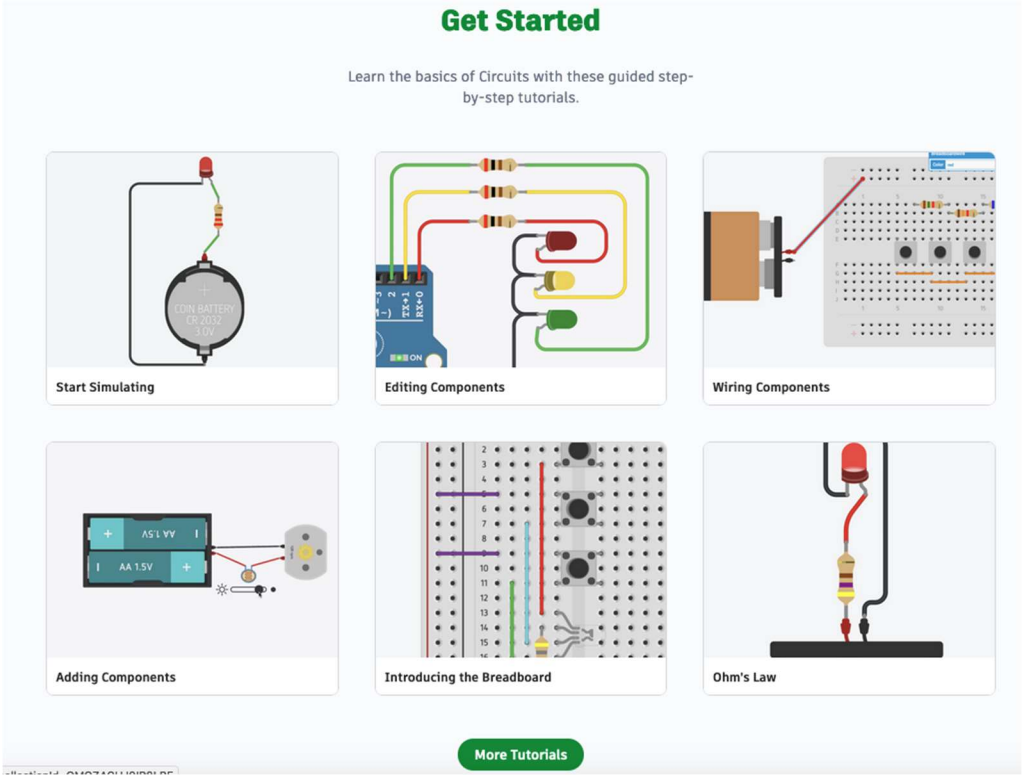
The tasks in red below should be done before coming to lab.

Pre-Lab Work (complete before attending the Lab session):

Arduino cannot be easily simulated in LTSpice, but it can be virtually simulated using a tool called TinkerCAD. All buttons, lights, switches, boards, LCD displays etc. can be simulated. Please beware that the TinkerCAD compiling tool can have issues. If you are working in TinkerCAD and you get unexpected compilation errors, copy your code to a new file and retry, as this may solve your problems. After you create a TinkerCAD account: https://www.tinkercad.com there are lessons and tutorials available to get you started:



Scroll down and you'll find tutorials to get you started:



The Lessons you should complete prior to the lab are:

Tutorials

1. Start simulating
2. Editing components
3. Wiring Components
4. Adding Components
5. Introducing the Breadboard
6. Blink an LED with Digital Output (*found under "More Tutorials"*)
7. Multiple LEDs and Breadboards (*found under "More Tutorials"*)

If you have no prior Arduino experience, be sure to complete a few of the lessons/projects to get familiar with the TinkerCAD tool and the Arduino system. If you attempt to complete this lab without the prior work and with no experience, you may find the lab frustrating – help yourself by being prepared.

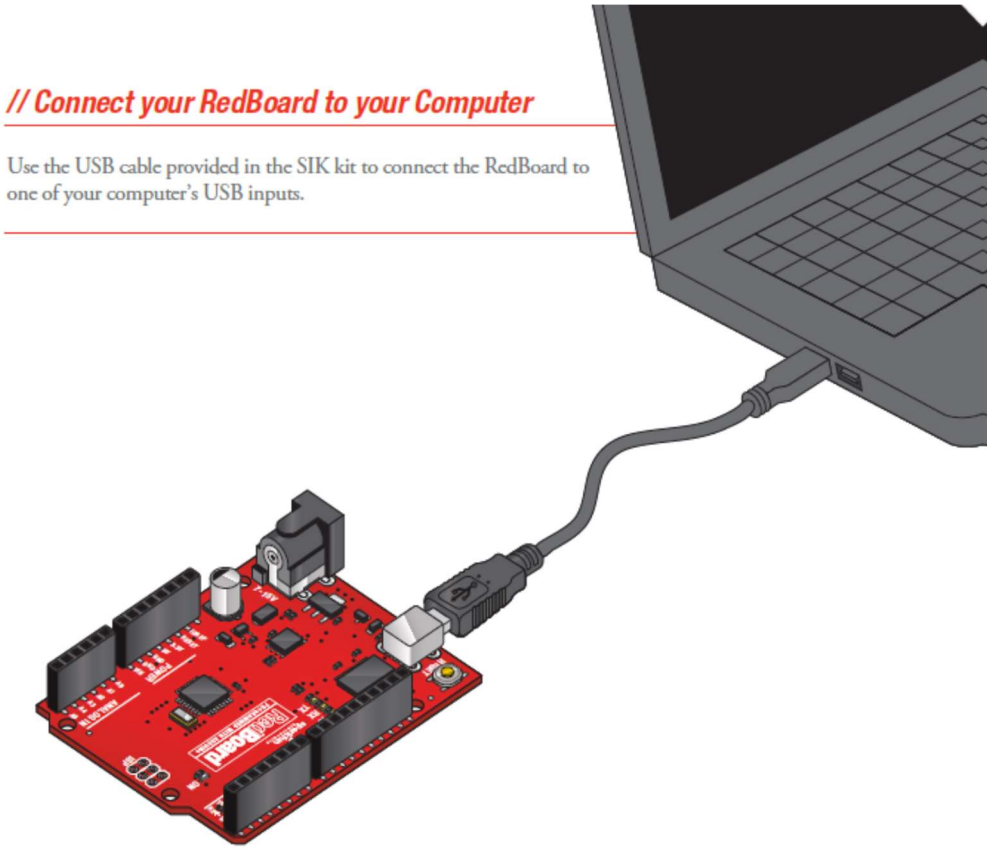
The Arduino software for creating code can be found here: https://www.arduino.cc/en/software . It's free and doesn't take much space – install it on your computer prior to attending the lab session. The software will also be available on the computers in the lab.

If you have any issues building the physical Arduino circuits, TinkerCAD is the simulation tool that you can use to plan and troubleshoot your designs. *(This will be very valuable for your Final Design Project)*

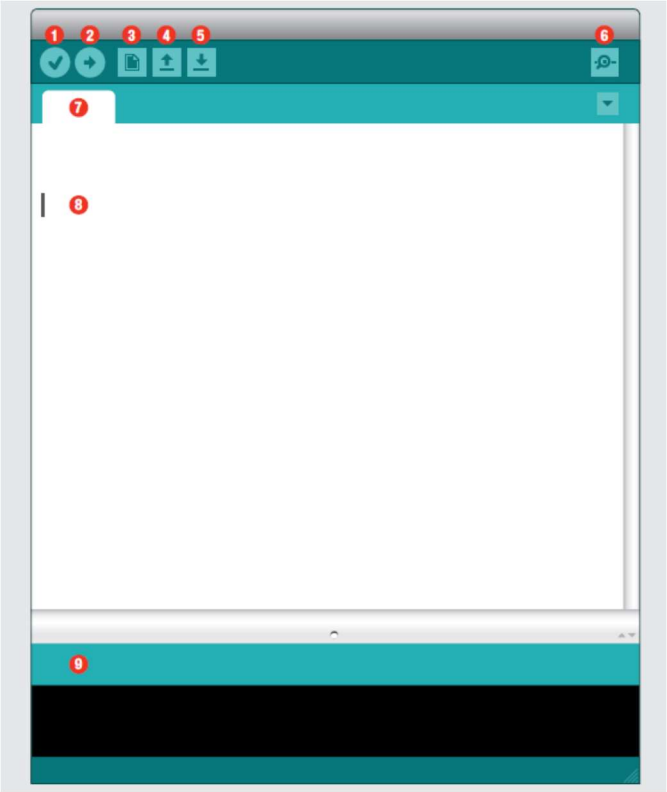
Lab Procedure

Part 1: Blinky Test

For this part, connect your UNO board (or equivalent) to your computer with the Arduino IDE installed via the USB cable - *be sure to have no other power source connected to the Arduino at this time, or you may accidentally fry your laptop!*

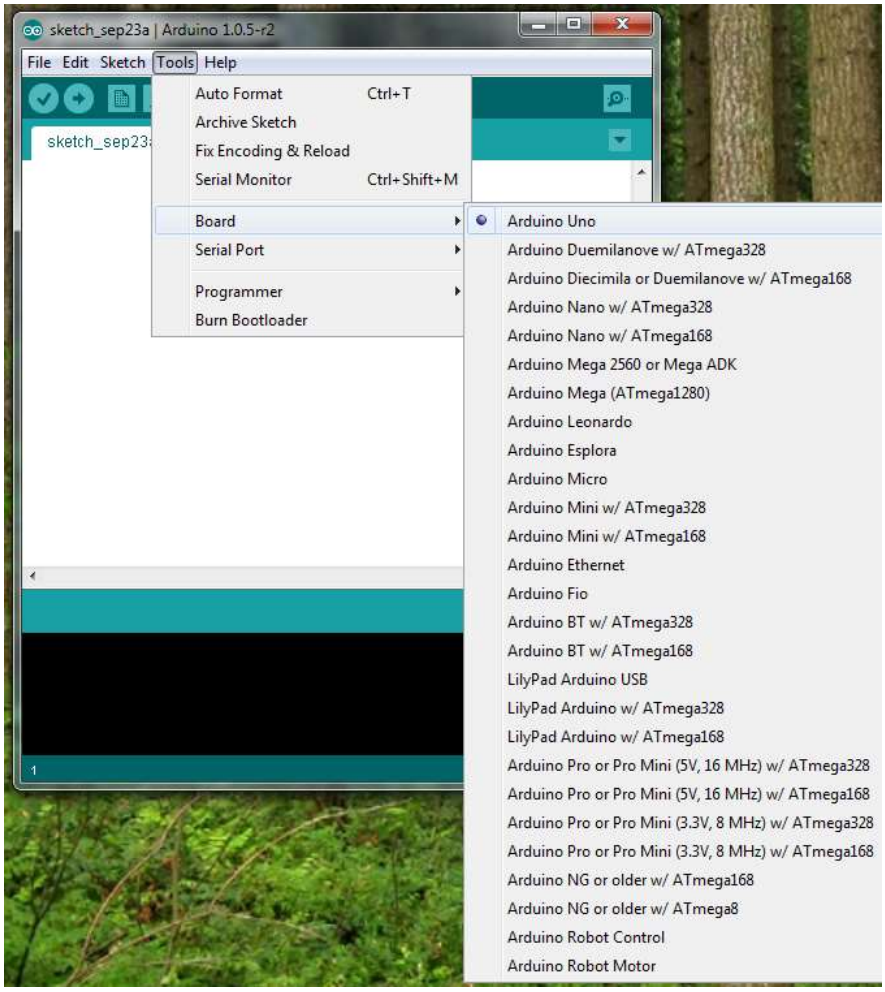


Now, open the Arduino IDE software on your computer. Poke around to get to know the interface. This step is to set your IDE to identify your UNO board. You will be building a basic circuit to blink an LED on and off for 1 second.



1. **Verify:** Compiles and approves your code. It will catch errors in syntax (like missing semi-colons or parenthesis).
2. **Upload:** Sends your code to the UNO board. When you click it, you should see the lights on your board blink rapidly.
3. **New:** The button opens up a new code window tab.
4. **Open:** This button will let you open up and existing sketch.
5. **Save:** This saves the currently active sketch.
6. **Serial Monitor:** This will open a window that displays any serial information your UNO board is transmitting. It is very useful for debugging.
7. **Sketch Name:** This shows the name of the sketch you are currently working on.
8. **Code Area:** This is the area where you compose the code for your sketch.
9. **Message Area:** This is where the IDE tells you if there were any errors in your code.

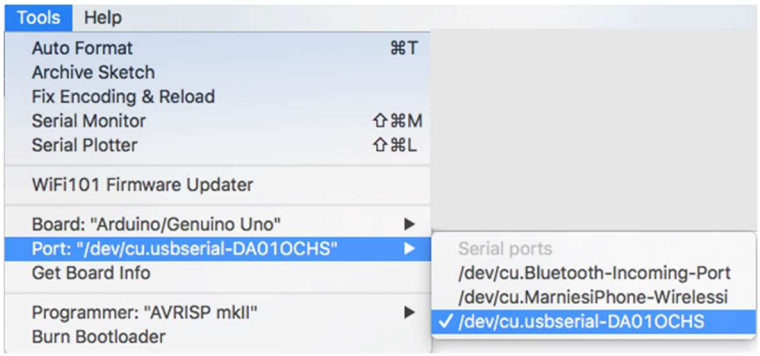
Next, select Tools->Board-> Arduino Uno (or Arduino/Genuino Uno). This will select the type of board which you are using.



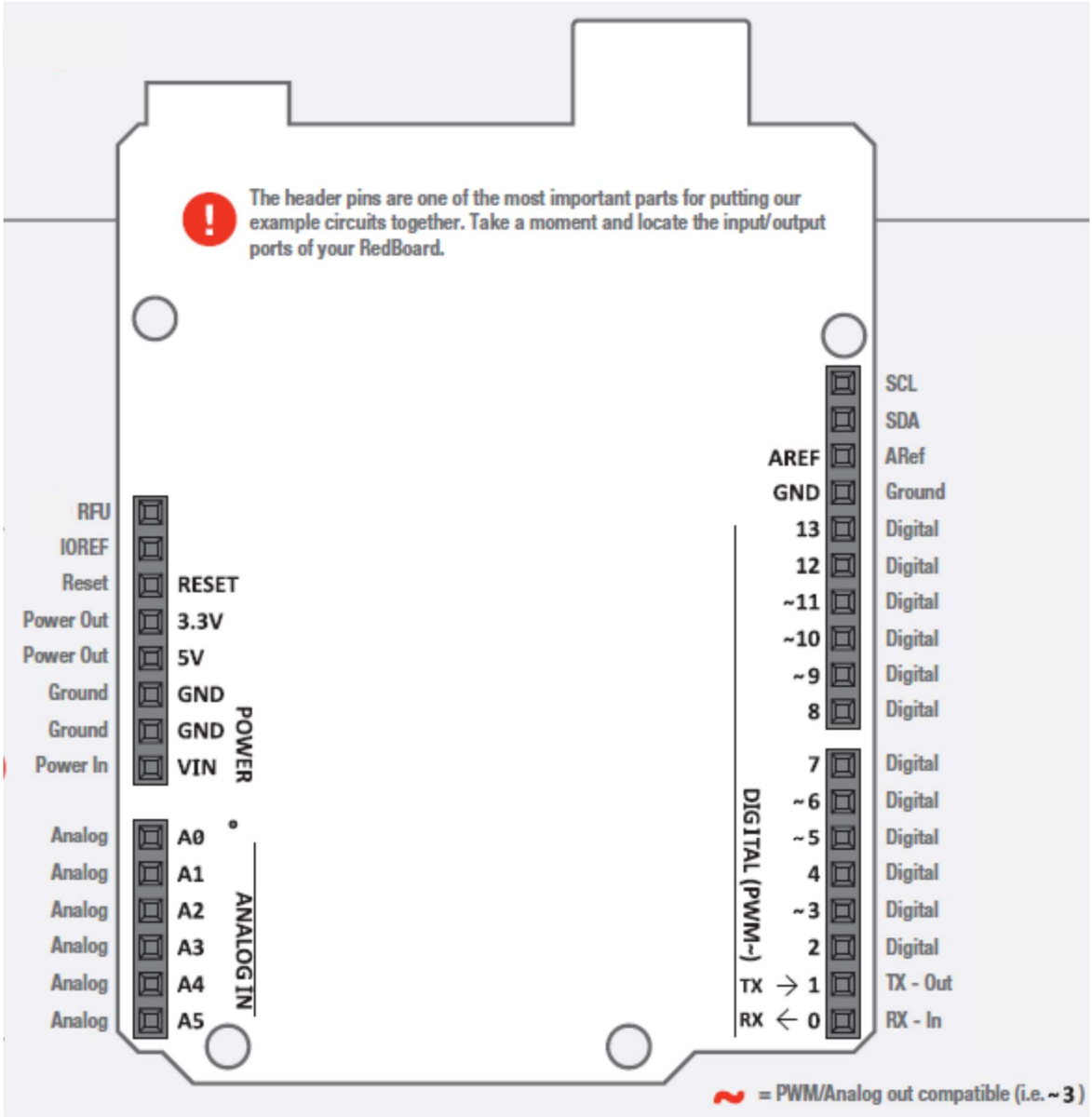
Then Tools->Serial Port-> COM3 (Or higher, whichever shows up). *(For Macs, Tools->Port->*usbserial*)*

Windows:

Mac:

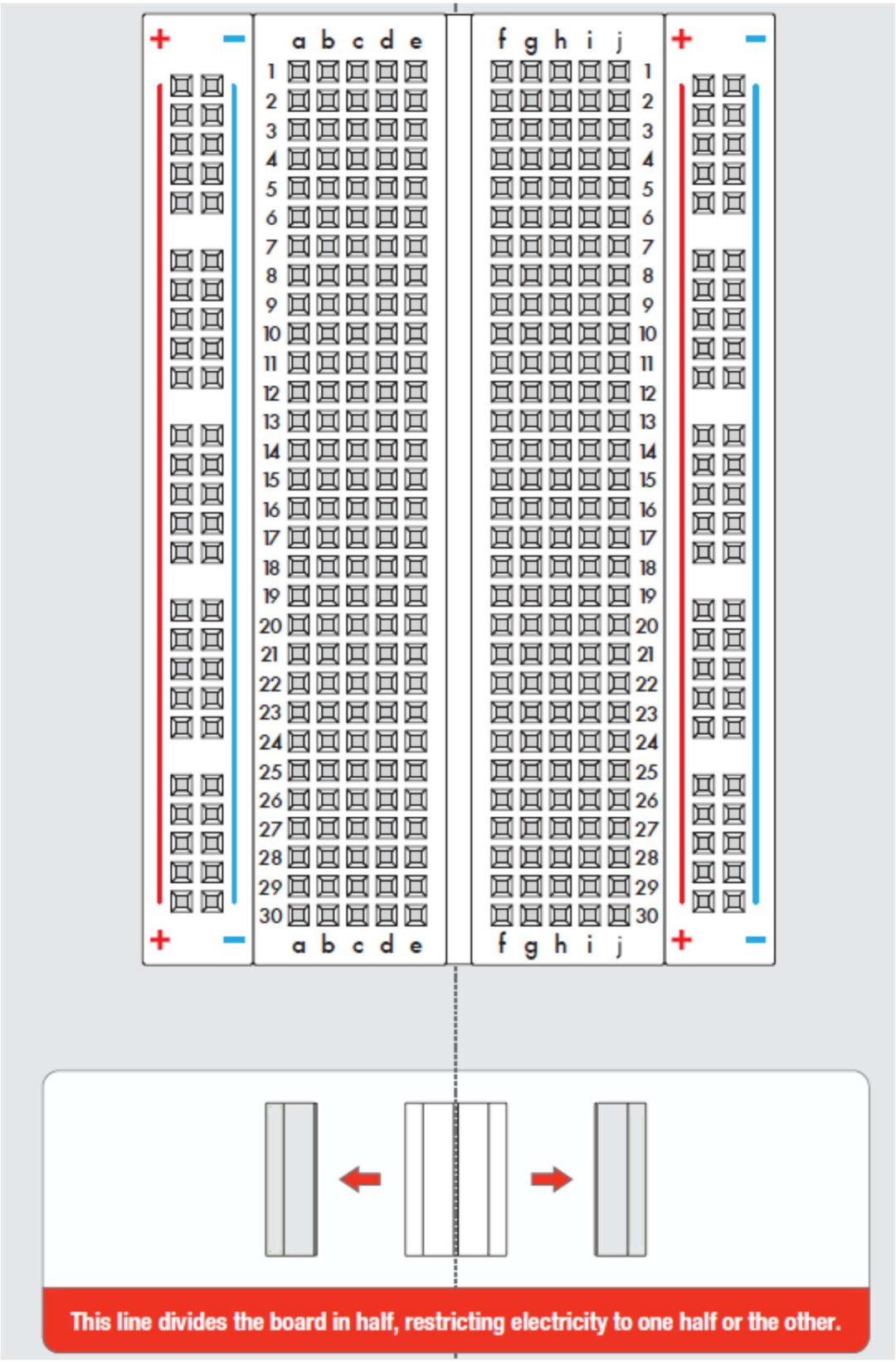


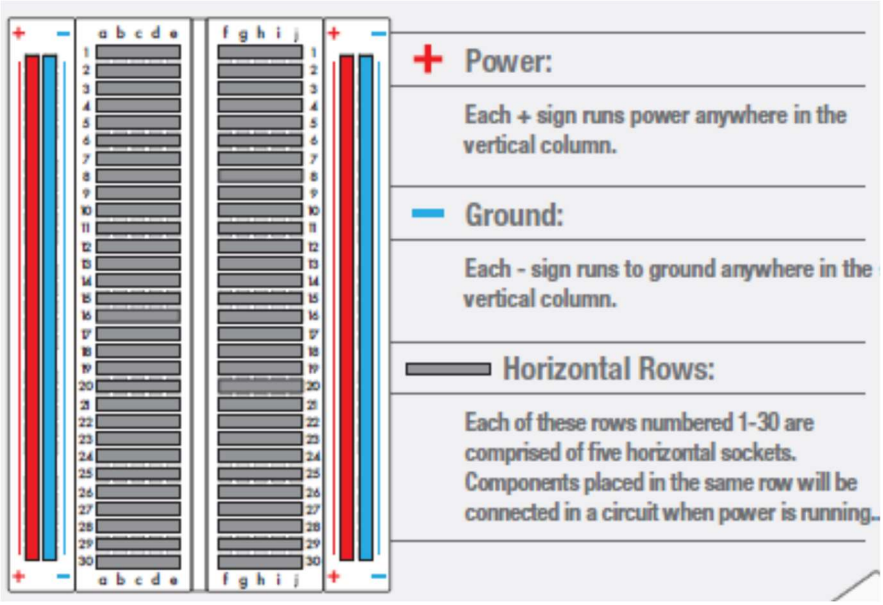
This is the pin diagram of the UNO board.



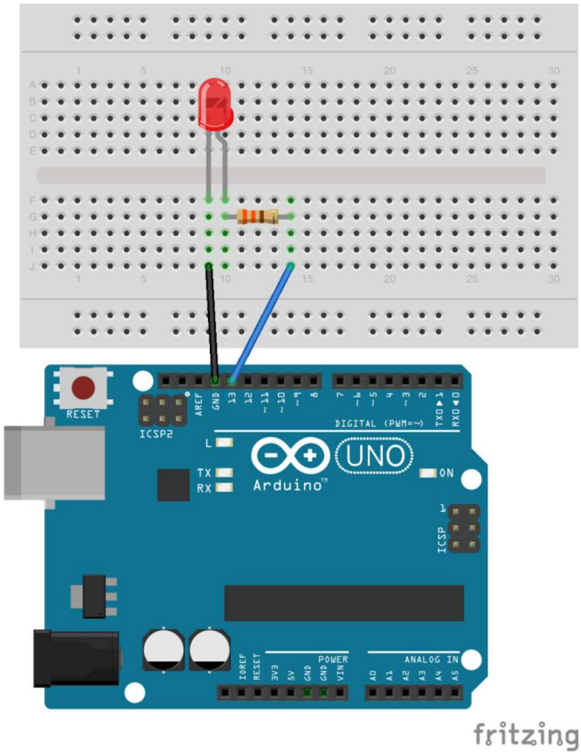
The digital pins (0 through 13) are either ON (HIGH/5V) or OFF (LOW/0V). Digital pins can be either inputs or outputs.
The analog pins (A0 through A5) can read any voltage between 0V and 5V. The microcontroller converts the analog voltage to a number between 0 (0V) and 1024 (5V). They can also act as additional digital inputs and outputs.
The power pins (VIN, GND, 5V, 3.3V, RESET, IOREF, RFU) are supplies, grounds, and references.

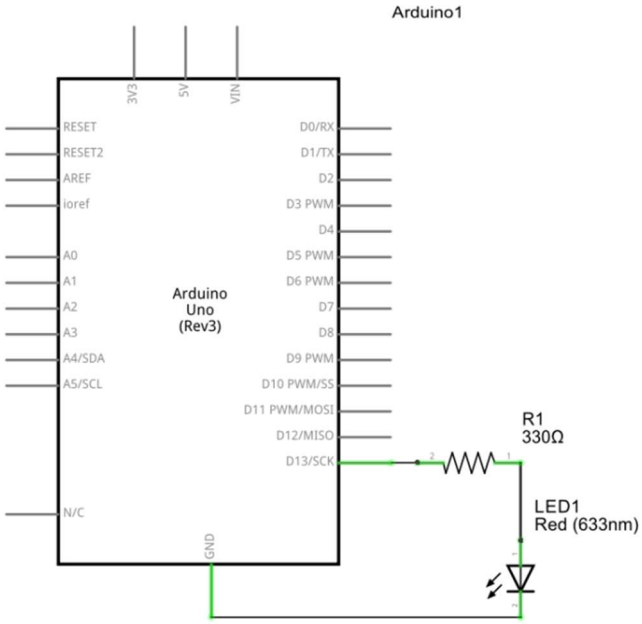
This is the layout of the breadboard.



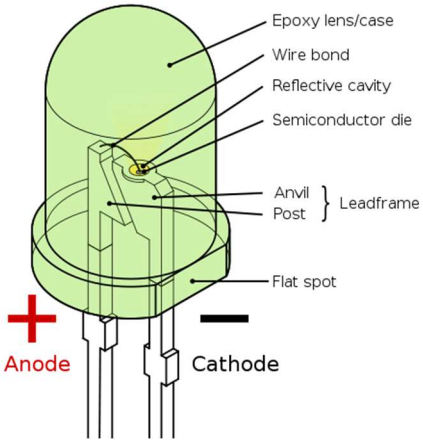


Using the hookup wires, a 330-ohm resistor, and an LED, construct the following circuit:





Remember LED polarity. The cathode is the side that has the flat edge on the bulb, and that indicates the negative side of the light which should be connected to Ground. The anode is the round side which should be connected to the resistor.



In the editor, enter the following code:

```
/* Hello World! */
int led = 13;

void setup() {
  pinMode(led, OUTPUT);
}

void loop() {
  digitalWrite(led, HIGH);
  delay(1000);
  digitalWrite(led, LOW);
  delay(1000);
}
```

Comment – Arduino won't even see this code.

Defining pin 13 as "led" (int is an integer number)

Everything in "setup" is done one time at the beginning.

Setting the function of the "led" pin to be an output.

Everything in "loop" repeats continuously until instructed to stop.

Setting the output "led" pin high (on).

A set delay with a duration of 1000mS between steps.

Setting the output "led" pin low (off)

Press the "Upload" button and observe the effect on the Arduino.

You can also create functions in the code (placed **<u>after</u>** the loop) that will perform tasks based on a given input. For example, if I want the LED to flash on an off with a set duration, I could create a function called "flash":

```
void flash(int duration) {
  digitalWrite(led, HIGH);
  delay(duration);
  digitalWrite(led, LOW);
  delay(duration);
}
```

Inside the loop, I can call up the function:

```
void loop() {
  flash(1000);
}
```

The LED will flash high for 1000mS, then go low for 1000mS.

Data Sheet Part 1

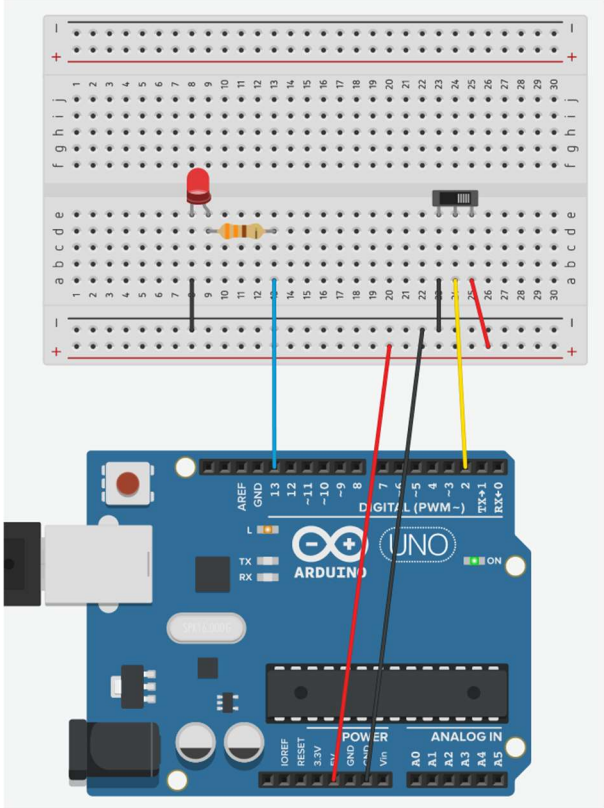
Answer all of the questions in the data sheet.

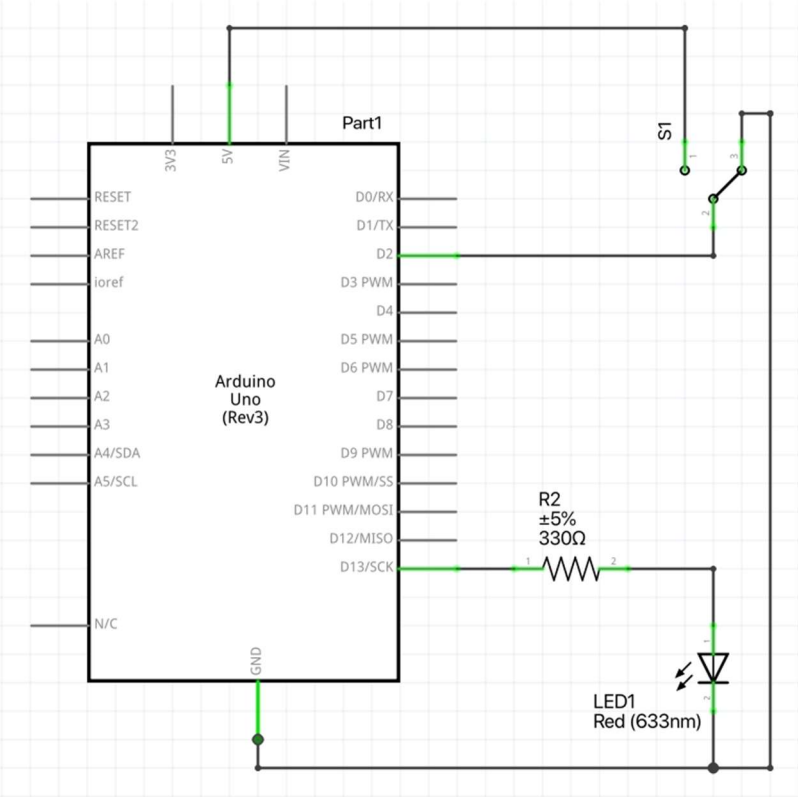
<u>ON YOUR OWN:</u> Modify the program to blink "S-O-S" in Morse Code. Make sure there is a clear long pause between the end of the message and the start of the next one. **Record your code on your Data sheet.** *<u>Hint:</u> if you aren't sure what S-O-S is in Morse code, look it up online.*

Syntax matters! If you don't use the correct syntax, the Arduino will not understand what you are trying to do.

Part 2: Digital Input with Switch

In this part, you will build a circuit that turns on a light when a SPDT (single pole-dual throw) slide switch is in the correct position, to only act when an input has been activated. Build the following circuit:





In the sketch editor, enter the following code:

```
int switchPin = 2;
int ledPin =  13;

int switchState = 0;

void setup() {

  pinMode(ledPin, OUTPUT);
  pinMode(switchPin, INPUT);
}

void loop(){

  switchState = digitalRead(switchPin);
  if (switchState == HIGH) {
    digitalWrite(ledPin, HIGH);
  }
  else {
    digitalWrite(ledPin, LOW);
  }
}
```

Data Sheet Part 2

Upload the code to your UNO board and observe the effect of toggling the switch. **Record your observations on the Data sheet.**

Using your existing SPDT switch and 1 LED circuit, run the following program to time how long the LED is on:

```
int switchPin = 2;
int ledPin =  13;
int switchState = 0;
unsigned long startTime = 0;
unsigned long timeON = 0;

void setup() {

  pinMode(ledPin, OUTPUT);
  pinMode(switchPin, INPUT);
  Serial.begin(9600);
}

void loop(){
  unsigned long now = millis(); //running counter in milliseconds

  switchState = digitalRead(switchPin);

  if (switchState == HIGH) {
    digitalWrite(ledPin, HIGH);
    timeON = now - startTime; //in milliseconds
  }
  else {
    digitalWrite(ledPin, LOW);
    startTime = now;
  }

    Serial.print("Time On: ");
    Serial.print(timeON);
    Serial.println(" milliSeconds");
}
```

Upload the program to the Arduino board, then click the Serial Monitor Button:

Toggle the switch and observe the effect.

Record your observations in the Data Sheet.

ON YOUR OWN: Write a program that implements the following functionality:

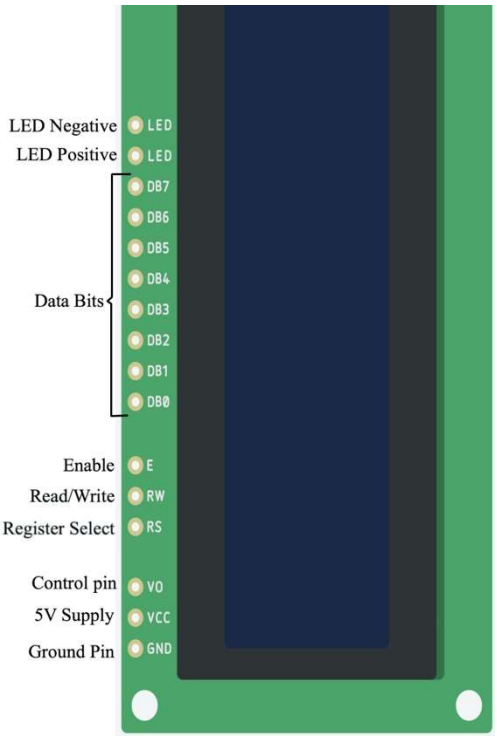
- Add a variable to also determine the current "off time" (how long has the switch currently been turned off) as well as the current "on time"
- Send the "off time" output to the Serial Monitor to print with the "on time"

Record your counting code on the Data sheet.

Part 3: Writing to the LCD Display

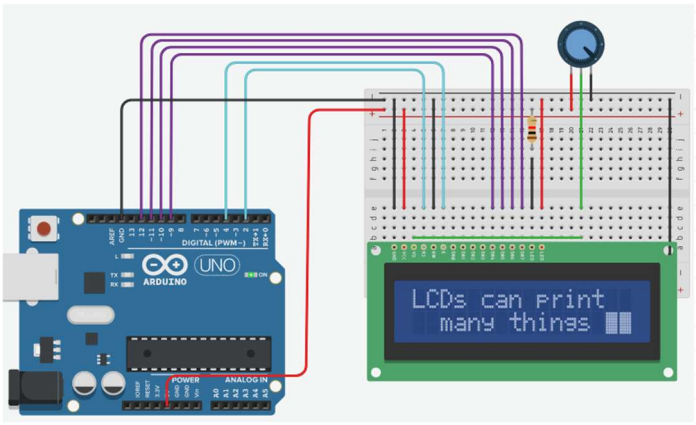
In this part of the lab, you will write to an LCD Display using the LCD1602 module with pin header. The pinouts of the LCD header are shown below:

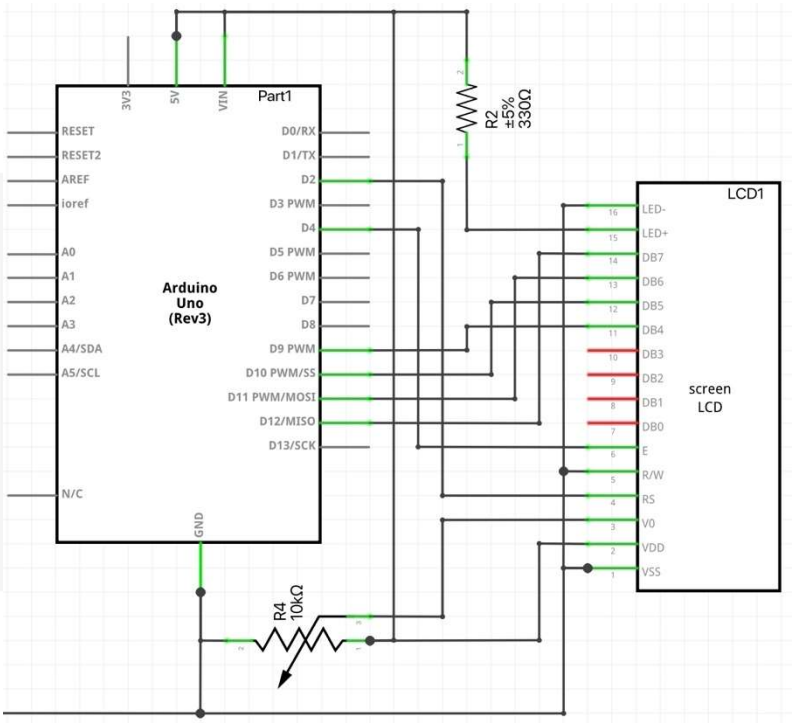
Pin Name	Pin Type	Description	Connection
GND / VSS	Ground	LCD ground	gnd
VCC / VDD	5V Supply	LCD supply	5V
V0 / VEE	Contrast Control	Adjusts contrast of LCD	0-5V through 10k POT
RS	Register Select	Toggles command/data	Digital Arduino output
RW	Read / Write	Toggle read/write	Set to 0V for read
E	Enable	Enable LCD functions	Digital Arduino output
D0 – D7	Data Bits	8 bits of data	Digital Arduino output
LED / A	LED Supply	Supply for LED backlight	5V through 220 Ohm
LED / K	LED Ground	Ground for LED backlight	gnd



The LCD screen is capable of displaying alphanumeric text. The LCD is a 16 x 2 Dot matrix so you have (16x2=32) 32 characters in total, and each character will be made up of 5x8 pixel dots. The data from the Arduino will instruct the LCD display on the state of each pixel dot, which is challenging if you had to determine it on your own, so the additional circuitry of the header block processes information to and from the Arduino to display meaningful information in a user-friendly format. The LCD can work in 4-bit or 8-bit mode. In 4-bit mode, the data is sent nibble by nibble (a nibble is a group of 4 bits), first the upper nibble, then the lower nibble, to form 8-bit data. The 8-bit mode is faster and allows you to send the 8-bit data directly, but it does require more data bits from the microcontroller. For this lab, you will use the 4-bit mode, as it's more commonly used. The LCD header component has a number of preset command instructions, that allow it to seamlessly communicate with the Arduino.

Using the UNO board, LCD display, 10K potentiometer, and a 220 Ohm resistor, build the following schematic:





The 220 Ohm resistor sets the current for the backlight LED of the screen and the potentiometer is used to adjust the contrast of the LCD screen.

In the sketch editor, enter the following code:

```
#include <LiquidCrystal.h>
const int rs = 2, en = 4, d4 = 9, d5 = 10, d6 = 11, d7 = 12; //set the pins
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);   //assign the display variables

void setup()
{
  lcd.begin(16, 2);

  lcd.print("   Welcome    ");

  delay(2000);
  lcd.clear();

}

void loop()
{
 lcd.setCursor(0, 0);
 lcd.print("LCDs can print");
 lcd.setCursor(0, 1);
 lcd.print("  many things ");
 delay(2000);
}
```

The code is written to add the functions for the LCD display from an Arduino library file (LiquidCrystal.h), initialize the connections between the Arduino and the LCD display, and then allow the Arduino and LCD display to communicate using the set functions programed on the header board. For this lab, you'll use the LCD functions:

begin - set's the size of the LCD screen (16 characters per line, 2 lines)

print - sends the alphanumeric text to the display

clear - clears the display

setCursor - sets cursor position (character #, line #)

In the set-up section, the LCD is initialized to verify that it starts up (this will only run at start-up), and then in the loop section, the LCD is used for the intended purpose (this will repeat until commanded to end).

Data Sheet Part 3

Upload the code to your UNO board and observe the effect on the LCD display. **Record your observations on the Data sheet.**

ON YOUR OWN: Write a program that implements the following functionality:

- The LCD display initializes with:

  Welcome to
  EEE 202

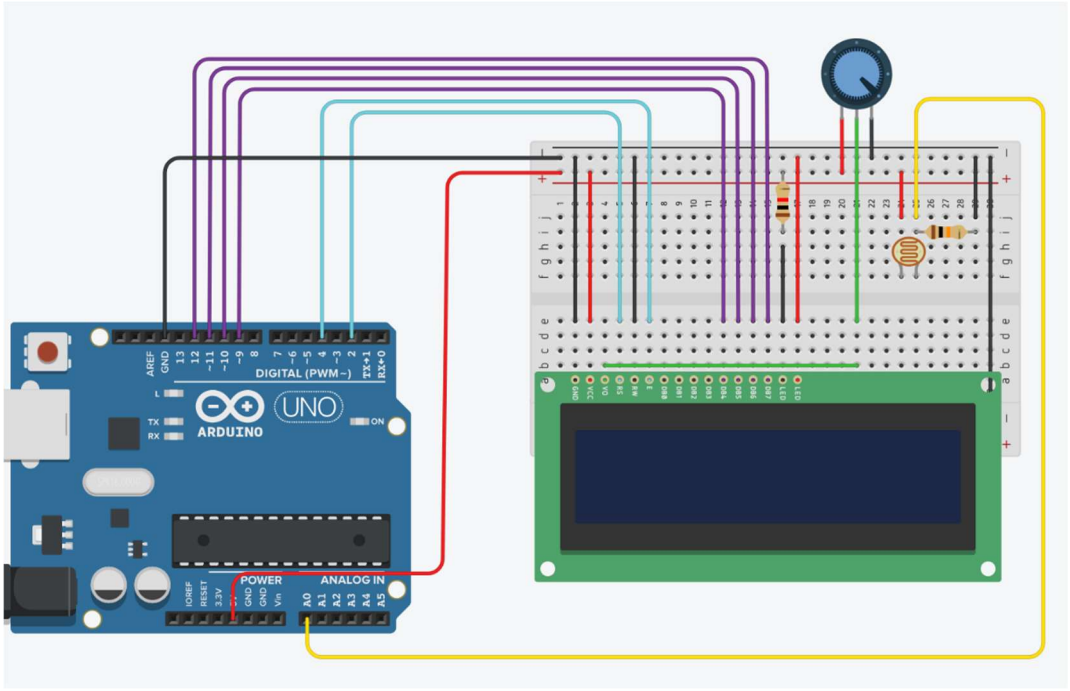
- Set a variable for the time the system has been running (like you did in part two with the "now" variable) and then display:

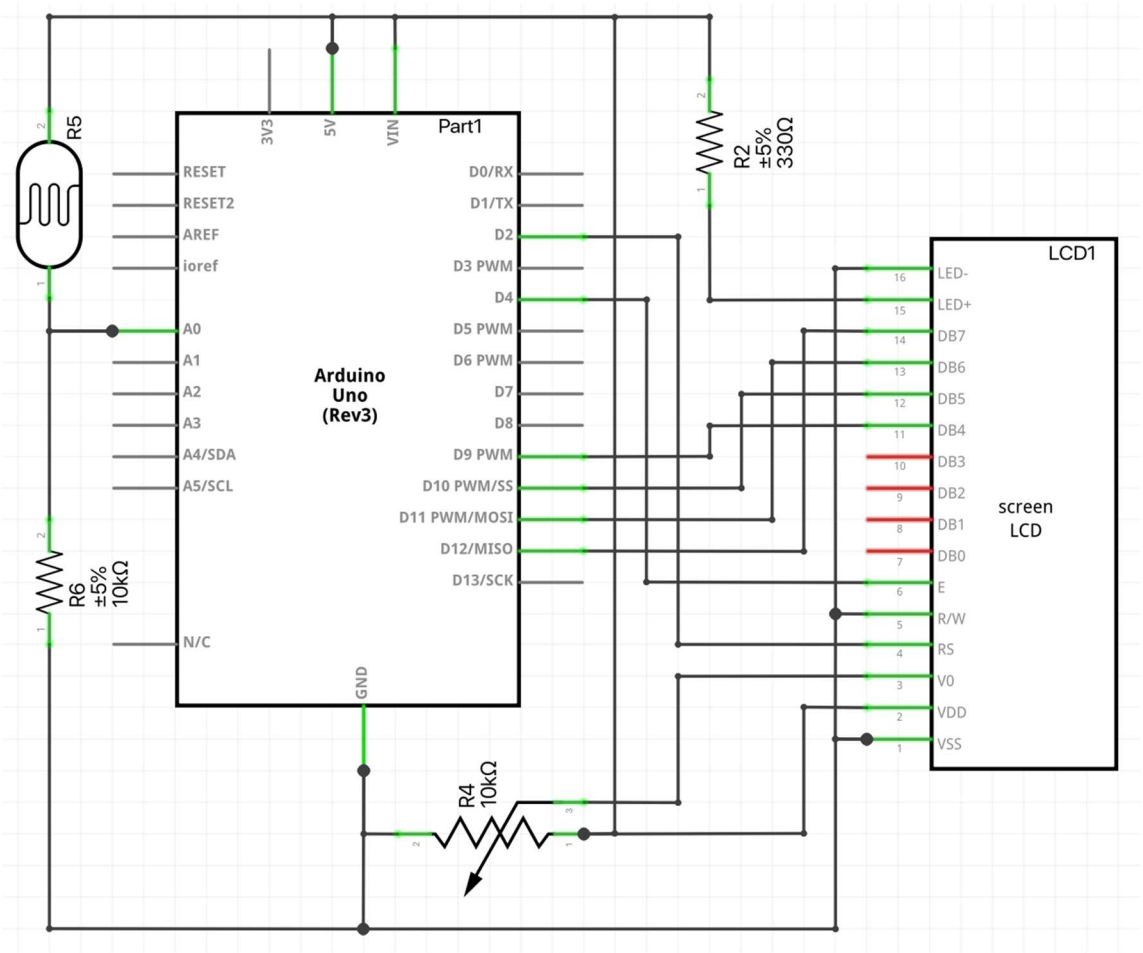
  Run time
  (*run time in milliseconds*) mS

Record your code on the Data sheet.

Part 4: Analog Inputs with Light Sensors and LCD Display

In this part of the lab, you will learn how to incorporate an analog sensor and use a range of values to trigger a reaction. Using the photoresistor and a 10K Ohm resistor, and your existing LCD circuit, build the following schematic:





Upload the following code to test your photoresistor circuit:

```
void setup()
{
  Serial.begin(9600);
}

void loop()
{
  int val = analogRead(0); //set analog pin 0 to sense photoresistor
  Serial.print("Sensor Value: ");
  Serial.println(val); //print value to Serial Monitor for testing
  delay(100);
}
```

Open the Serial Monitor. Wave your hand over the photoresistor. **Do not disconnect USB before closing the Serial Monitor.**

Data Sheet Part 4

Record your observations on your Data sheet.

ON YOUR OWN: Write code to send the LCD message "Day Time" when the photoresistor is exposed to light and "Nighttime" when the photoresistor is covered (combine lab parts 2, 3 & 4 together).

Copy your code to the Data sheet.

Fill in your data sheet. Labs can be done as a team, but each member must complete and submit their own data sheet.

Lab CLOS:

- Understand and apply the application of electrical circuits
- Design and build circuits using resistors
- Applies technical skills/knowledge to the development of a technology/product
- Integrates/synthesizes different kinds of knowledge
- Understanding of Arduino's input, output pins and serial ports
- Arduino LCD display observations as per different inputs