Arduino Workshop

Author: Ammon Shepherd

Date: October 26, 2020

Workshop Notes

Experience

- This workshop is independent of any other workshops. No prior knowledge or experience is needed.

Requirements

- If done virtually, participants will need their own laptop and Arduino kit.
- If done in person, provide laptops and Arduino kits.

Time

- This workshop takes about 1 hour and 30 minutes to complete

Guiding questions

- How will knowing how electronics work help you understand and improve your use of technology?
- What can you make to enhance your life? What tool can you make to do something for you, to provide information?
- Where can you find out more about electronics, working with Arduino, making things?

Workshop Objectives

At the end of this workshop, participants should know the following facts and procedures, understand the following concepts, and possess the following skills:

KNOW
The participant will know that...

- Conventional current states electrons move from positive terminal to the negative.

- Electron Flow is the term for the actual, real, physical flow of electrons, from negative to positive terminals.
- Electrical current is the rate of flow of electrons past a given point for a certain time, and is measured in amps.
- Resistors block/restrict flow of electrons
    - Good resistors are glass, ceramics, wood, air
    - Resistance is measured in ohms
- Conductors allow the flow of electrons.
    - Good conductors are most metals like silver, copper, gold, aluminum, water
- Switches stop the flow of electrons.
- Motors create motion.
- LEDs create light. The long leg of an LED is positive and the short is negative. LEDs allow electrons to flow in only one direction.
- Photoresistors sense the amount of light.

UNDERSTAND

The participants will understand that...

- Electrical current can be understood similar to water flowing through pipes.
- Arduinos take in information from sensors (like temperature, sound, light, humidity, motion), do some logic on that info, and then do some output (like light, sound, motion).
- A breadboard makes it easy to put together a circuit by connecting positive holes in a column, negative holes in a column, and rows of holes together.

DO

The participant will be able to...

- Connect electrical components to a breadboard to make a functioning circuit.
- Write code in the Arduino IDE on a laptop.
- Send code from the Arduino software to the Arduino board.
- Identify several electrical components
    - LED, resistor, button, photoresistor

WORKSHOP DIFFERENTIATION

- Beginner participants
    - This workshop is geared towards those with no experience with electronics or Arduinos. The workshop facilitator should go slow enough to allow all participants to complete the steps.
- Intermediate participants
    - This is really a workshop for beginners, so those with intermediate and expert knowledge may not find this a suitable workshop. If such participants remain in

the workshop, praise them for their hard work in learning what they know and suggest that they can stay to help others who are struggling.

- Experienced participants
    - This is really a workshop for beginners, so those with intermediate and expert knowledge may not find this a suitable workshop. If such participants remain in the workshop, praise them for their hard work in learning what they know and suggest that they can stay to help others who are struggling.
- Participants without physical resources
    - Sign up for a free Autodesk account (student version) for access to TinkerCad and the Arduino Simulator in the Circuits section (as of September 2020).
        - https://tinkercad.com
    - Virtual Breadboard by Microsoft (only available for Windows OS, as of September 2020).
        - https://www.microsoft.com/en-us/p/virtual-breadboard/9nblggh4tj3w

WORKSHOP SCRIPT

You can follow this script verbatim, or pick and choose what to talk about and how. The accompanying slide show goes along with the script. Everytime you see the [Show slide #] tag, you can go to the next slide. [Progress slide] tags let you know when to advance the animation/transition of the current slide.

Teacher Behavior	Student Behavior
Introduce the Workshop Facilitator [Show slide 1] ● Who you are, what you do, how you got to this point, what you hope to learn in the workshop **Introduce the workshop** [Show slide 2] **Objectives of the workshop** ● In this workshop you will learn about and how to use some basic electronics components: resistor, LED, photoresistor, button. ● You will learn what an Arduino is and why they can be useful. ● You will learn how to build a circuit and program the Arduino. ● And, hopefully, you will come away with a feeling that you can create your own electronics projects that will improve your and other people's environments.	Listen

Participant introductions [Show slide 3]	● Introduce yourself ○ Your name ○ What you do ○ Why interested in Arduino
[Show slide 4] In this workshop, we'll go through the basics of electricity, learn about some of the components, and then put together a few circuits. **Basics of Electricity** [Show slide 5] Sometimes it's hard to grasp electrical current and what happens in a circuit because you can't see what is happening inside the components. To help visualize the electrical concepts, you can imagine electrical current flowing through a circuit similar to water flowing through a system of pipes. 1. Electrons = water 2. Wires = pipes 3. Battery = Reservoir and pump 4. Voltage = The pump creates pressure in the system. In a house, your water pressure is usually between 50 and 80 psi. This is like the difference in electrical potential between the negative and positive side of the battery. This potential difference is called voltage. 5. Amperage = The rate that the water flows through the pump is like the rate at which the electrons travel in a circuit. 6. Resistance = making the pipe smaller or putting in a sponge or filter. Water pipes need to be all connected in order for the water to flow. If there is a break in a pipe, then the system is broken. Similarly, in an electronic circuit, all the components need to connect to each other in such a way that the electrons can freely flow from the negative to the positive side of the power source. If there is a break, the electrons don't flow out like in a water circuit, they just don't flow at all. Sometimes this is done intentionally, this is called a switch.	Look at slides, take notes, thoughtfully ponder how a system of water pipes is like an electrical circuit.

Switch = intentionally opening and closing a gap in the circuit. This is perhaps the most important concept for this workshop and when working with Arduinos. If the circuit is not working, the circuit is probably incomplete somewhere. So that is the first thing to always check. Make sure everything is plugged in where it is supposed to be plugged in. Any questions so far?	
The Components [Show slide 6] ● Power source ○ [Progress slide] ○ An electrical circuit needs power, or a source of stored electrons. To power an Arduino, you can use a battery, a wall outlet adapter, or a USB cable to a computer. ● Wires/conductors ○ [Progress slide] ○ We use wires to connect the different components to each other and the Arduino. All of the wires are the same inside. The colors are to help you organize your circuit. ● Resistors ○ [Progress slide] ○ Resistors are bi-directional, meaning that they can be used in a circuit in any position. We use the resistors to restrict the current to certain components. Resistors have colored bands on them to denote how much resistance they have. ● Photoresistor ○ [Progress slide] ○ A photoresistor is a special kind of resistor. It changes resistance based on how much light is detected. In our pipe analogy, you can think of it like a sponge that grows more dense or more porous depending on the light. ● LED (light emitting diode) ○ [Progress slide]	Get out arduino kits and look at all the components as they are discussed.

○ The LED, or Light Emitting Diode, is a component that is directional, meaning it allows electrons to flow through in only one direction. This would be like a one-way valve in a water pipe. As electrons flow through the diode, they lose energy in the form of light. Because LEDs are directional, there are two ways to determine the negative or positive side. First, there is a flat spot on the “brim” of the LED to denote the negative side. Second, the “leads” or legs of the LED are different lengths. The positive, or “cathode” side is longer. The negative, or “anode” side is shorter. Just as a plus sign has more to it than a negative sign, the positive lead is longer than the negative. ○ [Progress slide] ○ In a wiring diagram, the positive side of the LED is denoted with a bent lead. ● Breadboard ○ [Progress slide] ○ The breadboard is simply an easy way to connect a bunch of components together. The holes are connected in the back by rows of metal. If you were to remove the protective paper from the back, you would see the image on the left, how the rows are connected and the two columns on either side are connected. ○ [Progress slide] ○ For instance, the holes a, b, c, d, e in row 1 are all connected to each other, but nothing else. ● Arduino ○ [Progress slide] ● Arduinos are small mico-controllers that allow you to gather input from sensors (like sound, motion, temperature, humidity, light, etc), through code impart some logic based on that input, and then generate some kind of output (like sound, motion, light, etc). ● For example, you can have a sensor that detects light, then your code can determine to turn on or off an LED based on how much light was detected. That’s the basics of a nightlight, which we will construct at the end of this workshop.	

Assessment Pause for questions and assessment of knowledge and understanding. Ask questions like: ● Is there any confusion about any of the components talked about so far? ● What's the difference between a resistor and a diode? ○ Resistor = restricts flow of electrons ○ Diode = only allows electrons to flow in one direction ● How does a photoresistor affect a circuit? ○ It changes resistance based on the amount of light. ● How would you use the breadboard to connect an LED to a resistor? ○ One lead of the LED should be in the same row as a lead of the resistor. For example, one lead from the LED in hole 1A, and one lead from the resistor in hole 1D. ● What do you think would happen to the brightness of an LED if you changed a small resistor for a larger resistor? ○ The brightness would decrease.	Ask questions or answer questions
The Arduino IDE [Show slide 7] ● The Arduino IDE is the interface we use to write the code, or the logic, that we will then send to the Arduino board. [Show slide 8] ● The software is free and available at https://www.arduino.cc/en/Main/Software ● There is a download version (for any operating system), or a web version. ● While you install the IDE, I'll go over some of the basics of the interface. [Show slide 9] ● The two most important things to check after opening the Arduino software are 1) the **board**, and 2) the **port**. ● If you go to the **Tools** menu, you'll see a submenu for **Board:** and a submenu for **Port.** **Check The Board** ● Click on the Board submenu to see the various types of Arduino boards you can connect to. For this workshop, you'll either use the Arduino Uno or the Arduino 101. ● [Progress slide]	Download the Arduino IDE for your operating system and install. Select the correct board from the **Board** submenu. Connect the Arduino to the computer. Select the correct port from the **Port** submenu.

- You may need to look down at the very bottom of the list to find your board.

[Show slide 10]

- If your specific Arduino board is not present, click on the **Board Manager…** menu to bring up the board manager.
- [Progress slide]
- Search for your board and click the **Install** button.

Check The Port

[Show slide 11]

- Next, make sure the Arduino is plugged into the computer with the USB port. There is only one way the cable can go.
- Select the **Tools** menu, then the **Port** submenu. There should be an option there that has the name of the Arduino board. Select that port.

Check Code and Upload

[Show slide 12]

- Next we'll look at two buttons in the IDE that are useful.
- [Progress slide] Zoom in to the buttons..
- First is the code check button. This checks your code for errors, and if there are any errors, it will let you know.
- [Progress slide]
- The second button checks your code, and then sends it to the Arduino if there are no errors.
- You usually just need the second button.

Writing Your Code

[Show slide 13]

- Each new document you start is pre-filled with a couple of pieces of code. In Arduino language, these documents are called sketches. They can also be called scripts, or code files.
- [Progress slide]
- The first piece of code is called the setup function. A function is like a bunch of code you write that can be used again later, just by using the function's name.
- Line 1 starts our function. Here we use the word 'void' to tell Arduino what kind of function it is. It is a function that doesn't return any information, it just does something, so it is a 'void' function. The name of the function is 'setup'. The parentheses designate if we can pass information to the variable when it is called/run/used. In this case there

is nothing. Then finally, we have an open curly brace to say, this is where the function's code starts, and a closing curly brace on line 4 to say, this is where the setup function's code ends. ● Line 2 is just a comment. The Arduino just ignores this line. It's just there for us humans. ● Line 3 is left blank. ● Line 5 is just an empty line. ● We would use the setup function to tell the Arduino to do stuff as soon as it loads the code. It does all of the stuff in the function once, then moves on to the next function. ● [Progress slide] ● The second piece of code is another 'void' function called loop. ● Line 6 starts the function, it's type is void, there are no parameters passed into the function (nothing between the parentheses). ● Line 7 is another comment. ● Line 8 is blank. ● And line 9 is the closing curly brace. ● When the Arduino gets to this function, it runs the code inside over and over again as fast as it can. It loops over the code continuously until the Arduino loses power or you send it new code.	
Make a Circuit! [Show slide 14] Finally, we get to do what we came here to do. Make a circuit! [Show slide 15] Put together a circuit like this. An LED connected to a 100Ohm resistor connected to pin 12, and one lead connected to GND on the Arduino. ● Reminder: the LED might start blinking, depending on what code already is on the Arduino. [Show slide 16] Next, write the code needed to turn on the LED. ● [Progress slide] ● In the setup function, you'll tell the Arduino to send out a current on pin 12 using a new function called pinMode. This tells the Arduino that a specific pin is either sending out current (OUTPUT) or receiving current (INPUT). ● [Progress slide] ● In the loop function, you'll use a new function to tell the	Connect the circuit. Write the code. Ask questions if it doesn't work.

Arduino to actually send current to the specified pin using the digitalWrite function. This function needs a pin number and a value of HIGH or LOW. (See the Deeper Understanding section for more on why HIGH or LOW instead of ON or OFF.) ● [Progress slide] ● And this is how you could write those two functions inside the setup and loop functions. Allow time for participants to put together the circuit, write the code, and troubleshoot any issues. Check in with each participant to make sure the LED turns on. Troubleshooting: ● Check Board and Port settings ● Check that wires are plugged in firmly ● Make sure all connections are made ● Check that the correct pin # is used in the code	
Blinking Light! [Show slide 17] Next, we'll make the light blink. With just one more function and a few lines of code we can turn the LED into a blinking light. ● The delay function allows us to pause the script for a set number of milliseconds. 1000 seconds = 1 second. ● See if you can write the code that will allow the LED to blink. That means it will turn on for 1 second, then turn off for 1 second, then turn on for 1 second, then turn off for 1 second, etc. After sufficient time for participants to try to write the code, and after working with participants individually to get their code working, you can progress the slide to show the solution. ● [Progress slide] ● Here is one way to write the code. Offer a challenge or exercise for further study: ● Play around with the number in the delay function to see what that does to the blinking light.	Write code to make the LED blink.
Assessment ● What is the least number of lines of code needed to blink the LED? ● Is the order of the delay and digitalWrite arbitrary? How can you reorder the code and still get the LED to blink?	Answer assessment questions.

● What happens if the LED is flipped in the circuit? ● Why can you put the resistor on the negative or positive side of the LED?	
Night Light Circuit! Now we move on to the next component needed for making our nightlight, the photoresistor. Remember the photoresistor changes its resistance based on the amount of light it senses. We'll use this ability to determine when to turn the LED on or off. [Show slide 18] First, you'll need to add the photoresistor to your circuit. This will need to be connected to a voltage output, the 5v pin, and to ground, the GND pin, with a 10K Ohm resistor. A third wire will connect to the photoresistor at the point where the resistor connects and then connect to analog pin zero on the Arduino, marked as A0. The Arduino can't actually detect the amount of resistance a component is contributing to the circuit, but it can detect the difference in current or voltage. The photoresistor and the 10K Ohm resistor create a circuit called a voltage divider. Think of it as two sponges in a water pipe. If you know the flow of water going into the pipe and the "resistance" of one spong, then as the other sponge changes and you observe the new flow rate, you can calculate the resistance of the second sponge. In any case, we are able to use the change in potential current (the voltage) using the photoresistor and a second resistor, which allows us to calculate the change in resistance. We then use that number in the logic of our software to determine when to turn on the LED.	Connect the photoresistor and 10K Ohm resistor to the Arduino following the diagram given.
Night Light Code! [Show slide 19] Now we can update our code in the Arduino IDE program. This sketch introduces two new programming concepts; variables and the if...else logic structure. The first, as seen on lines 3, 4, and 5 is that of creating a variable. A variable, in programming, is like a box that we put information in. There are four parts to making a variable in the Arduino language. ● [Progress slide] ● First, we must specify what kind of information can be stored in this variable. In this case we need a number, so we use integer, or int.	Type code into the Arduino IDE and send it to the Arduino. Cover the photoresistor with your hand, and the LED should light up.

- [Progress slide]
- Some of the most used options are
    - int = numbers
    - float = numbers with a decimal point
    - char = all characters
- [Progress slide]
- Second, the label, or name, of the variable.
- [Progress slide]
- Third, we use the single equal sign to assign a value to the variable. This is what puts the information into the box.
- [Progress slide]
- Fourth, the value that we want to store in the variable.

On line 3, we create a variable to hold the sensitivity value. The higher the number the more sensitive the nightlight is. The photoresistor will send a value from 0 to 1023. Let's say that the amount of light in the room means the photoresistor gives us a value of 500. A value of 100 for the threshold means that when the value of the sensor drops below 400, then the LED will turn on. If we set the threshold to 20, then the LED would turn on when the photoresistor sends a value of 480.

On lines 4 and 5, we create two variables that have no value. We create, or 'instantiate', the variables here so that we can use them later. This has to do with the scope of the variable, or which parts of the code can 'see' or recognize the variable and its value or not. The 'resistance' variable will eventually hold the value of the photoresistor everytime the code checks for it. The 'initialPhotoValue' will hold the value of the photoresistor when the code runs for the first time.

On line 7, we start the setup function.

On line 8, we set the mode for pin 12 to be output, we'll be sending current to the LED on pin 12.

On line 9, we grab the value of the photoresistor and assign the value to the variable 'initialPhotoValue' that we created on line 5. We want to grab the value of the photoristor here because this value totally depends on how much light there is in the room. This value is dependent upon the amount of light in the room each time the Arduino is turned on and the code is run.

We use the analogRead() function, and pass zero as the parameter. This means we will read in the value of the

photoresistor that is plugged into pin A0 on the Arduino. The A stands for Analog.

On line 12, we start the loop function.

On line 15, we read the current value of the photoresistor. This value possibly changes each time the code loops through.

On line 19, we start the other new programming concept, the if...else statement. This is a logic statement that allows us to make a test of some values, and then based on the result the Arduino will run certain code. This line starts the if statement with the word ‘if’. Then in parenthesis we put the condition that is tested. In this case, we want to see if the current value of the photoresistor is less than the initial value minus the threshold. Let’s say that the current value of the photoresistor is 400, and the initial value was 645, and our threshold is 100. We can substitute those values in for the variables to test whether the equation is true or not. Is 400 less than 645 - 100?

If that is true, the code on line 20 is run, or executed, and lines 21-23 are ignored. If that is false, then line 20 is ignored, and line 22 is executed.

Notice the curly braces on lines 19, 21 and 23. Their purpose is similar to the curly braces used by the functions setup and loop. They let the program know what code is encapsulated, or enclosed, or belongs, to the ‘if’ or the ‘else’ part of the statement.

So, if the current value of the photoresistor is less than the initial value minus the threshold, then line 20 is executed. The digitalWrite function, or method, is used to send current to pin 12 which is connected to the LED.

Else/otherwise, line 22 is executed, and the digitalWrite method is used to stop current to pin 12.

Troubleshooting Tips:

- Check that the circuit is correct. Plug each wire and component in exactly as shown on the diagram if necessary.
- Check that there are no missing semicolons. The Arduino IDE will give visual clues in the code (squiggle underlines, etc).
- Check for errors in the output section of the IDE.
- Double check that the Board and Port are correct.

Assessment ● What happens when you change the sensitivity of the circuit by changing the value of the 'threshold' variable in the code, line 3? ● Can you make it so the LED turns on when a shadow passes over the photoresistor?	Change the sensitivity of the circuit by changing the value of the 'threshold' variable in the code, line 3.
Proclamation [Show slide 20] Proclaim that the participants are not bonafide electrical engineers and coders! Thank each participant for coming. Answer any questions.	Pack up and fill out survey

Resources

- Equipment Needed (determine beforehand if the instructor provides the equipment or if the participants must bring their own)
    - Arduino kit, or Arduino with
        - 1 USB cable
        - 5 jumper wires, any color
        - 1 LED
        - 1 10K Ohm resistor
        - 1 100 Ohm resistor
        - 1 photoresistor
        - 1 breadboard
    - Laptop with Arduino IDE installed
        - Power cable if needed and access to power outlet
- Slides: https://docs.google.com/presentation/d/1nAePGoZULM71w17anXhqHNs5JbMpbm6SNq2v2b3B24Y/edit#slide=id.g97cc113b5f_0_287

Assessment

- See assessment questions in the workshop script.
- Post Workshop Assessment Questions.
    - These can be asked or presented as a survey on paper for quick feedback.
    - What other things do you want to learn about?
    - Was there anything you were hoping we would cover that we didn't?
    - Are you confused about anything that we did cover.

Homework

Take this project to the next level!

- Make the LED blink using the delay() function.
- Make a traffic light with red, green and yellow LEDs.
- Complete more experiments from
    - https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v41
    - https://create.arduino.cc/projecthub/projects/tags/arduino
    - https://www.instructables.com/Arduino-Projects/

Promote

- This workshop showed you how to use an Arduino Micro controller. We have other workshops that show you how to use a Raspberry Pi. It's very similar to a micro-controller, but the Raspberry Pi is a full computer but the size of an Arduino. So it opens up so many more possibilities.
- Throw in the DH angle, give examples of how technology is used in the humanities fields

Further Resources

- Arduino Guides and Projects
    - https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-the-arduino-101genuino-101-board/all#experiment-1-blinking-an-led

    - https://learn.sparkfun.com/tutorials/sparkfun-inventors-kit-experiment-guide---v40/all#project-1-light
    - https://create.arduino.cc/projecthub/projects/tags/arduino
    - https://www.instructables.com/Arduino-Projects/
    - https://maker.pro/arduino
    - https://www.hackster.io/arduino
- Arduino Software IDE
    - https://www.arduino.cc/en/Main/Software
- ELEGOO Arduino Kit
    - https://www.amazon.com/ELEGOO-Project-Tutorial-Controller-Projects/dp/B01D8KOZF4
- Online Arduino Simulator
    - https://www.tinkercad.com/learn/circuits
- Dig deeper into how electricity works
    - https://sites.google.com/a/acsbr.org/mr-leong-chuen-kit-physics-resources-site/17-current-of-electricity
    - https://theengineeringmindset.com/how-electricity-works/
    - https://www.youtube.com/watch?v=mc979OhitAg

Deeper Understanding

Q: Why are HIGH and LOW used to designate a pin as on or off?
A: https://www.arduino.cc/reference/en/language/variables/constants/constants/

Defining Pin Levels: HIGH and LOW

When reading or writing to a digital pin there are only two possible values a pin can take/be-set-to: HIGH and LOW.

HIGH

The meaning of HIGH (in reference to a pin) is somewhat different depending on whether a pin is set to an INPUT or OUTPUT. When a pin is configured as an INPUT with pinMode(), and read with digitalRead(), the Arduino (ATmega) will report HIGH if:

- a voltage greater than 3.0V is present at the pin (5V boards)
- a voltage greater than 2.0V volts is present at the pin (3.3V boards)

A pin may also be configured as an INPUT with pinMode(), and subsequently made HIGH with digitalWrite(). This will enable the internal 20K pullup resistors, which will pull up the input pin to a HIGH reading unless it is pulled LOW by external circuitry. This can be done alternatively by passing INPUT_PULLUP as argument to the pinMode() function, as explained in more detail in the section "Defining Digital Pins modes: INPUT, INPUT_PULLUP, and OUTPUT" further below. When a pin is configured to OUTPUT with pinMode(), and set to HIGH with digitalWrite(), the pin is at:

- 5 volts (5V boards)

- 3.3 volts (3.3V boards)

In this state it can source current, e.g. light an LED that is connected through a series resistor to ground.

LOW

The meaning of LOW also has a different meaning depending on whether a pin is set to INPUT or OUTPUT. When a pin is configured as an INPUT with pinMode(), and read with digitalRead(), the Arduino (ATmega) will report LOW if:

- a voltage less than 1.5V is present at the pin (5V boards)
- a voltage less than 1.0V (Approx) is present at the pin (3.3V boards)

When a pin is configured to OUTPUT with pinMode(), and set to LOW with digitalWrite(), the pin is at 0 volts (both 5V and 3.3V boards). In this state it can sink current, e.g. light an LED that is connected through a series resistor to +5 volts (or +3.3 volts).