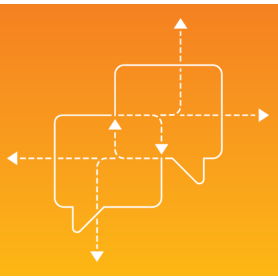
Qualcomm® Thinkabit Lab™

5 Day Lesson Plan

Arduino, LED's, and Servos



This Thinkabit Lab lesson plan is designed for a teacher to use in their class as a week-long activity using the same materials as used in the Qualcomm Thinkabit Lab. The lesson plan is structured for a 50 minute class period over 5 days.

Basic Overview:

- Day 1: Circuits and intro to LEDs
- Day 2: Program a blinking LED, make it blink faster
- Day 3: Intro to Servos
- Day 4: Robo crafting day
- Day 5: Final touches and class presentations

Before you begin:

- Download Arduino on all of your student computers. You may either have the students do this the week before or do it on your own when there are no students in the room.
- Make sure you have all of the parts needed for the week in each kit. Note: if you are using the Arduinos in each class, make sure that you have enough Servos for each class. (I.e. if you have 5 classes of 36 students, you will need 18 servos per class, for a total of 90 servos.)
- Let students know that on Thursday they will start to create and that they will need to bring in their own materials. If possible, you may want to provide basic supplies such as popsicle sticks, duct tape, pipe cleaners, etc. Larger items, toys, even leaves or pieces from nature can be brought in by the students for their creations.

Day 1: Circuits and intro to LEDs

- Objective: Students will understand how electricity flows through a circuit and will be able to use that knowledge to light an LED using the Arduino.
- Materials: Large Qualcomm battery, 1 full water bottle*, 1 empty water bottle*, funnel*, Arduino kits (1 per pair), laptops (for power only), USB cables
- Procedure:
    - Activity: Human Circuit
        - Description: Have students hold hands in a circle with the Qualcomm battery being held between two students. When students are all connected, the battery will make noise, when the chain is broken, the noise will stop. Have students take turns breaking the circuit and then reconnecting.
        - Notes: Let students have fun finding what will and will not complete the circuit. They can touch any part of the other person as long as there is skin to skin contact. You may also have students with braces try touching the metal of the braces to see if it will carry the flow of electricity.
        - Materials: Qualcomm Thinkabit Lab battery.
        - Time: 5-10 minutes

Authored By: Vanessa Myers 1/21/2016

- Activity: Drawing Circuits
  - Description: Students will learn how to draw a circuit. Start with a basic battery and LED (without a resistor) before moving on to a circuit with a resistor (which is what they will build).
  - Notes: Be sure to connect the circuit schematic to the circuit activity they just completed (i.e. they must complete the circuit by drawing a full circle).
  - Materials: Paper under doc cam or blank page of Promethean Board or Smart Board.
  - Time: 5 minutes
- Activity: Understanding Resistors and their purpose
  - Description: This activity will help students understand the need for a resistor when using a large power source such as ours. Start out with your two water bottles, and ask if the students think you can pour the water from one bottle to the other. Explain that if any of the water gets outside the bottle, then you have failed (you can make it dramatic!) Then pour the water from one to the other in very close proximity. Explain that this is what it's like when we use the right amount of electricity, but often we use more electricity than we actually need to power the LED, this would be like pouring the water from 3 feet above the empty bottle. Again, ask the students if they think it is possible to get all of the water in the bottle without spilling any. Demonstrate and really try to get it in the bottle, but you won't be able to from that height. Then ask what tool might help to get more of the water into the bottle. The correct answer is a funnel. Then use the funnel (in the empty bottle) and pour again and this time most of the water should get in the bottle. This is what a resistor does, it helps us control the flow of electricity so that the LED doesn't get too much power all at once and burn out.
  - Notes: Be slow and careful when pouring the water to really try to make this point. If you are worried about spilling water in your classroom set up towels or take your class outside. You may also consider having a student hold the empty bottle so that it doesn't accidentally tip over during the demonstration.
  - Materials: 1 empty water bottle, 1 full water bottle, 1 funnel, towels (optional)
  - Time: 10-15 minutes
- Drawing Circuits Cont'd
  - Description: Students will draw the same circuit as before, but this time they will add in a resistor between the positive end of the battery and the positive side of the LED.
  - Notes: Optionally you can make them use pencil on their initial circuit and then they can erase the space where they will add the resistor.
  - Materials: Same as above.
  - Time: 5 minutes
- Constructing a circuit
  - Description: Using their circuit schematic, students will construct the circuit using the Arduino, breadboard, LED, resistor, and wires. Start by laying out all of the parts and labeling each. It is important at this point that you talk about the polarity of the LED (short leg is negative, long leg is positive) and that resistors do not have polarity. Walk students slowly through the construction by following the schematic. Colored pencils or markers may also help to show what wires go where, but it usually is best to write out exactly where the wires go. Once all of the wires are attached, have them follow the flow of electricity to make sure it is a complete circuit before connecting the Arduino to the computer with the USB cable.
  - Notes:
    - Red wire: Vin to red row
    - Black wire: Gnd to black row
    - LED: short leg (negative) in black row, long leg (positive) in blue row

- Resistor: Red row to blue row
- You will also need to explain that electricity flows through a breadboard along the rows, so all of the black row is connected, same for red and blue, but they are not connected to each other.

  - Materials: Arduino, LED, red and black wires, resistor, USB Cable
  - Time: 20 minutes
- Clean up: Have students disconnect the wires, LED and resistor and put everything away for the next class.

Day 2: Program a blinking LED, make it blink faster

- Objective: Students will use their knowledge learned yesterday of how to power an LED and will add a programming component to make the LED's blink. They will then investigate at what time delay will they not be able to see the LED blink anymore because it is blinking too fast for their eyes to see.
- Materials: Arduino kits (1 per pair), laptops (with Arduino downloaded), USB cables
- Procedure:
  - Revisiting circuits
    - Description: Have students take out the materials used yesterday and their notes and challenge them to construct the circuit without any help.
    - Notes: Now would be a good time to explain that the only real purpose of the Arduino at this point is to provide power to the LED, but that it is not sending any information. We will program it in the next step.
    - Materials: Arduino, LED, red and black wires, resistor, USB Cable
    - Time: 5-10 minutes
  - Coding an LED to blink
    - Description: At this point, students will use the Arduino program on their computers to code the LED to blink. The first thing you will want to do is be sure the Arduino is connected to the computer through the USB cable. In order to upload code to the Arduino you will need to make sure it is connected to the right port. To change this, have students go to "Tools" then down to "Port" and then select the largest COM number. (Note: if the computers stay with the same Arduinos each day, they should only have to go through this process once.) Have students open the "blink" code by going to "File" down to "examples" then to "Basics" and select "blink". Have them upload the code to their Arduino by clicking the arrow at the top of the page (next to the check). At this point, nothing should happen with their LEDs, but you should make sure that everyone's code actually uploaded. At the bottom of the page on the blue bar it should say "upload complete" and there should be no orange writing in the black space below. To make the LED Blink they will need to take the red wire that is in Vin and move it to pin 13, on the other side of the Arduino board. At this point, their LED's should be blinking, you will want to walk the room and make sure that everyone's is working.
    - Notes: If the LED is not blinking, check all connections first, many times they put the wire in the wrong hole. If the circuit is correct, check to make sure that the code was uploaded properly and if necessary try uploading the code again.
    - Materials: Same as previous
    - Time: 5-10 minutes
  - Defining the parts of the code
    - Description: At this point you will want to go through the code that they uploaded line by line and explain what is happening. Students should take notes on this as you go over it.
      - 3 parts to coding in Arduino

- Part 1: defining variables – In this code we have nothing to define, so there is nothing here but comments.

- Part 2: setup – This is where we tell Arduino where things go and what they will be doing.
  `pinMode(13, OUTPUT)` tells the Arduino to use pin 13 as an output.
- Part 3: loop – This is the place in the code where the action happens. This part of the code will repeat forever.
  `digitalWrite(13, HIGH)` tells the LED (in pin 13) to turn on by making the voltage level high.
  `delay(1000)` tells the Arduino to wait for 1 second (1000 milliseconds) before reading the next line of code, so this mean the LED stays on.
  `digitalWrite(13, LOW)` tells the LED to turn off by making the voltage level low.
  `delay(1000)` tells the Arduino to wait for 1 second (1000 milliseconds) before reading the next line of code, so this mean the LED stays off.
- /* and // means you are writing a comment to humans. This does not get sent to the Arduino, it is to help humans understand the code.

  - Notes: Be sure that students understand what the delay is doing here, it will be crucial in tomorrow's lesson.
  - Materials: Paper and pencil for notes.
  - Time: 10 minutes
- Changing the delay until the blinking is no longer visible
  - Description: Have students alter the code by changing the delay value. They can change each one to the same value or have them be different, this is their chance to explore. They will need to upload their code each time, they can't just change it on their code and have it change automatically. Once they have played for a little while, stop them and tell them that all of the lights in the room are blinking, but they blink too fast for our eyes to see. Challenge them to change the code to find the largest delay value for which the LED blinks so fast they cannot see it blink. They will need to make the delay value the same for both delays. The correct value is around 10 or 11, but have them try a few values to check.
  - Notes: Let students explore during this time and play with the LED's. This should be a fun activity and should get the kids playing and exploring on their own.
  - Materials: Same as above.
  - Time: 10 minutes
- Optional Activity if Time (10 – 15 minutes): SOS Code
  - Description: SOS in Morse code is dot dot dot, dash dash dash, dot dot dot. You can demonstrate this with sound to help them get a better idea of what they need to code. Essentially they want it to blink a lot, but the delay will be different. Suggest a 500 delay for a dot, a 1000 delay for a dash, a 500 delay between each dot and dash, a 3000 delay between each letter, and a 5000 delay after the SOS code before it repeats in the loop. It should look like the code below:

```
digitalWrite(13, HIGH); //dot
delay(500);
digitalWrite(13, LOW);
delay(500);
digitalWrite(13, HIGH); //dot
delay(500);
```

```
digitalWrite(13, LOW);
delay(500);
digitalWrite(13, HIGH); //dot
delay(500);
digitalWrite(13, LOW);  //wait between letters
delay(3000);
digitalWrite(13, HIGH); //dash
delay(1000);
digitalWrite(13, LOW);
delay(500);
digitalWrite(13, HIGH); //dash
delay(1000);
digitalWrite(13, LOW);
delay(500);
digitalWrite(13, HIGH); //dash
delay(1000);
digitalWrite(13, LOW);  //wait between letters
delay(3000);
digitalWrite(13, HIGH); //dot
delay(500);
digitalWrite(13, LOW);
delay(500);
digitalWrite(13, HIGH); //dot
delay(500);
digitalWrite(13, LOW);
delay(500);
digitalWrite(13, HIGH); //dot
delay(500);
digitalWrite(13, LOW);  //wait before repeating
delay(5000);
```

        - Notes: This would be a good place to teach students how to copy and paste to make their code writing go much faster.
        - Materials: Same as above.
        - Time: 10-15 minutes
    - Optional Activity if Time (3-5 minutes): Fade Code
        - Description: another example you can open and upload to your Arduino is Fade. This will make your LED fade on and off rather than blink on and off. Open the example code by going to File -> Examples -> 1. Basics -> Fade. Upload the code, then change your red wire to pin 9. The light should now fade on and off. If they want to make the light fade slower, they can change the delay at the very bottom of the code.
        - Notes: This code uses an if statement, something that is rather complex, but also very helpful. A student could use an if statement to do something when something else happens, which is what most coding in the real world looks like.
        - Materials: Same as above.
        - Time: 3-5 minutes
    - Clean up: Have students disconnect the wires, LED and resistor and put everything away for the next class.

Day 3: Intro to Servos

- Objective: Students will use their knowledge learned yesterday of how to code the Arduino and today will learn how to code a Servo. They will then explore how the code controls the servos and explore the differences between the 2 types of servos. They will also revisit circuits as they wire their servos.
- Materials: Arduino kits (1 per pair), laptops (with Arduino downloaded), USB cables
- Procedure:
    - Coding a Servo
        - Description: This will probably the most difficult section of the week for students because there is a lot of typing and everything needs to be perfect in order for the code to work. Go slowly through it and make sure you have a way to connect your computer to a screen in the room so that they can follow along with you and see exactly how everything should be written. This time also needs to be quiet in the classroom so that everyone can focus. You may want to consider some way of checking on students' progress with writing the code so that you know when to move on.
            - Part 1: Defining variables

              ```
              #include <Servo.h>
              Servo myservo;
              ```

              Get this by going to Sketch -> import library -> servo

              This names our servo myservo, but you could name it whatever you want
            - Part 2: Setup

              ```
              void setup(){
                myServo.attach(3);
              }
              ```

              the Servo will be attached to pin 3
            - Part 3: Action We'll explain the action when we plug in the servos

              ```
              void loop(){
                myServo.write(0);
                delay(1000);
                myServo.write(180);
                delay(1000);
              }
              ```
            - Upload the code (remember you may need to change the COM port). If there are any errors, have them self-assess and then jump in if necessary.
        - Notes: As students are copying the code, it is important that everything is spelled correctly and there are capitals where there are supposed to be. The colored words are easy to tell if they are correct because if they don't change colors it's incorrect. Also make sure that there are semicolons at the end of each line and a closed bracket at the end of the setup and loop.
        - Materials: laptops, Arduino, USB Cable
        - Time: 10-15 minutes
    - Connecting the Servo and exploring the movement
        - Description: Now that the Arduino is coded to operate the Servo, we will connect the wires so that the Servo can actually operate. Again, walk them slowly through the connections. We will start with the regular (small) Servo. The easiest connections are to the Servo itself as it has red, white, and black wires that are pretty obvious to connect to. From there, the black wire again gets connected to GND, the red wire to Vin and then the white wire will be connected to pin 3 (as we coded it). Ask students to make observations about the motion of the Servo. Then have them disconnect the wires from the regular Servo and plug in the full rotation (large) Servo. Again have them make observations on the motion of the Servo.
            - Regular Servo: The value is the angle position of the Servo and the delay tells it to stay there for that long. Values can range from 0 to 180.

- Full Rotation Servo: The value tells the speed and direction of the Servo. 0 makes the Servo spin quickly to the left. 180 makes the Servo spin quickly to the right. 90 would make this Servo stop. The delay tells the Servo how long to spin at that speed and direction.
- When would you use each? Have an open discussion with the students about what possible projects would require each type of Servo.

  - Notes: Make sure that the students make use of their notebooks as you talk about how the values of myServo.write affect the movement of the Servo. This is also a good place to talk about the delay as many students think that it will spin and then stop and then spin the other way, not spin and then spin the other way without stopping.
  - Materials: laptop, Arduino, USB Cable, red, white, and black wires, regular and full rotation Servos
  - Time: 15 minutes
- Optional Activity if Time: Challenge Code
  - Description: Write up challenge codes on the board and have students work with their partners to try to match the code
    - Ex: spin fast to the left for 1 second, then slowly to the right for 2 seconds, then slowly to the left for 1 second, then fast to the right for 2 seconds.
  - Notes: Some may really struggle with this while others will sail right through. Again, copy and paste can be really handy here, so explain that to them if necessary.
  - Materials: Same as above
  - Time: 10-15 minutes
- Programming the Arduino for their project
  - Description: Give partners time to discuss their projects, peruse materials, and adjust their code to fit their project. Some may be fine with the original code and others may want to add more levels or just have a full rotation Servo spin one direction the entire time. When they have the final code for their project they need to save the code on their computer using their names so they can upload it again tomorrow and Friday without re-writing the entire code.
  - Notes: Some students may need extra help during this time to get their Servo to do exactly what they want it to do. If students are finished with their programming, have them design their project on paper before allowing them to start crafting tomorrow.
  - Materials: Same as above
  - Time: 5-10 minutes
- Clean up: Have students disconnect the wires and put everything away for the next class.

Day 4: Robo crafting day

- Objective: Students will use the code they wrote yesterday for their Servo to construct a project that uses their Servo. Today students are only limited by their creativity and the supplies in the room!
- Materials: Arduino kits (1 per pair), laptops (with Arduino downloaded), USB cables, crafting supplies, hot glue guns and glue
- Procedure:
  - No instruction today, this is a full day of crafting for the students to work on their projects and be creative. If they finish early, challenge them to add more. If they've added everything they want, have them write out a description of their project for the presentations tomorrow. They will need to try their project with the Servo spinning, so they will need to upload THEIR code to the Servo at the beginning of the day, but after that they only need the computers to power the Arduino.

- Clean up today will take a bit longer than other days, I would give at least 10 minutes to make sure the room is clean. Students should have their projects completed by the end of the day to ensure they have enough time to present tomorrow.

Day 5: Final touches and class presentations

- Objective: Students will present their final projects to the class, explaining what they have created and what is spinning and how they got it to do that motion.
- Materials: Projects created yesterday, Arduino, laptop, USB cable
- Procedure:
  - Give students time to finalize their projects and upload the correct code to their Arduino.
  - When everyone is ready, have each pair go in front of the room and give a 2 minute description and demonstration of their project. You may want to film them or allow them to film each other so that they have a take-away since they will not get to take their projects home.
  - Clean up: They will need to disconnect the Arduino and the wires and return them to the kit for the next class.
  - Optional project clean up: You will need to get the Servo off of their project, so you can either have them take it apart or you can do it yourself after they leave. You can also recycle some of the craft items, or you can let them take the craft part home without the Servo if they want to keep it (and if you don't mind giving up the supplies).