DISCOVERING YOU

ENGINEERING YOUR WORLD

Chevron ASEE AMERICAN SOCIETY FOR ENGINEERING EDUCATION NBC NEWS LEARN NSF

ANGELIQUE JOHNSON

Engineering Cochlear Implant Model to Become a Great Engineer

STEM Lesson Plan for Grades 6-8

Video https://youtu.be/7NznUqoVXuc

NGSS Standards
CCC: Structure and Function
ETS1.A: Defining and Delimiting Engineering Problems
ETS1.B: Developing Possible Solutions
ETS1.C: Optimizing the Design Solution

Helping all students, especially girls, to be interested in engineering as a possible future can be challenging. Research suggests that role models are important for helping students to see themselves in a jobs where they have been underrepresented. In addition, having challenging and fun engineering experiences help students to want to become engineers.

Many engineering projects though are focused on competitions, but that isn't the essence of engineering. The cycle of determining a problem, identifying what's needed to solve the problem, trying and testing possible solutions, and optimizing and iterating to find an adequate solution is what makes an engineering project.

In this lesson plan, students will watch a video where Angelique Johnson explains how she became a successful engineer. After figuring out what she says are some key factors to being a great engineer, they will investigate an intersection to see if the yellow light is the right length of time. Later they will compare their work with what is accepted in the field. Finally, they will look back at their work habits to determine how closely they matched skills that the engineer suggested were important.

Part I: Watching the Angelique Johnson Video

Before the students watch the video, the teacher should explain that in this video an engineer will explain what makes her a successful engineer. The teacher should ask students to record what personality traits, desires, and behaviors are important to becoming an engineer.

For younger students, you may need to use sentence starters like

Angelique Johnson said that she had to overcome the obstacles of _________

Angelique Johnson said that she wants to _______

The video has on-screen icons that will help students when she is saying key components of her success. For some students, pausing the video at those moments will help them better record what is going on.

In small groups have the students summarize what they saw and then make sure that the entire class has all of the points. While they may have more than these, they should at least note:

- Angelique Johnson is an engineer because she wants to help people.
- Angelique Johnson is an engineer because she wants to be creative..

Making a Great Engineer Checklist

Students now should now make a checklist of things for themselves to do if they want to be a good engineer. Then when they do something on the checklist, they should mark it off. For example

Activity	
I helped someone	𝍸 I
I didn't give up when something didn't go the way I planned	𝍸 III

Students will use this checklist several times in the following projects. Don't assign points or give too much praise, otherwise students will just game the system. We just want them noting when they are doing something a good engineer does, helping them to internalize that they can be an engineer. Alternatively, you can make it the task of one of the members of the group to note when their groupmates are being good engineers.

Part II: Engineering Cycle

This engineering project can be fairly advanced if students are asked to do all the coding themselves. It can be made easier if they are given the sample code and then asked to modify it, and it can be used by even students new to microcontrollers if the code and microcontroller are treated as components of a larger device that is just used and left unexplained. Remember, that different students may be at different levels, and a project like this allows you to differentiate instruction to the levels of each student.

Students will simulate part of how a cochlear implant works by using a microcontroller to do the data collecting and manipulation of sound information and also using a string of LEDs to show how strongly parts of the cochlea would be stimulated by an array of electrodes.

The goal is to be able to use patterns brightness or colors on the string of LEDs to recognize two or three words.

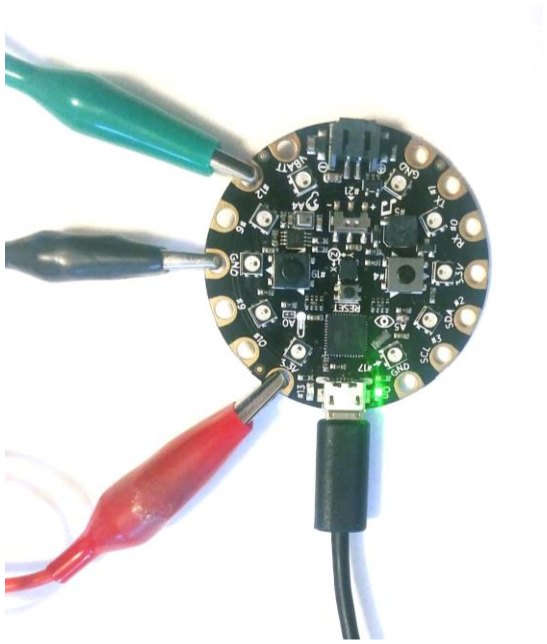
Materials

- Adafruit Circuit Playground or Circuit Playground Express
- 30 LED strip of NeoPixels with Alligator Clips (or another NeoPixel strip, see below)
- Computer

Although a variety of Arduino-based microcontrollers work well, the Adafruit Circuit Playground Classic and Adafruit CIrcuit Playground Express have built in microphones, come with FFT routines, and connect easily to NeoPixels. Adafruit's NeoPixel Strips with alligator clips makes them easy to connect to the Circuit Playgrounds' pads.

Setting Up the Computer, Board, and LED Strip

1. Download and install the Arduino IDE software or use the online tools from Arduino. Follow instructions at Adafruit for setting up and testing your Adafruit Circuit Playground or Adafruit Circuit Playground Express.
2. New installations of the Arduino IDE software should contain all necessary libraries, but you should check to be sure. Go to Tools-->Manage Libraries. Search for "Adafruit Neopixel". Hover your cursor over the row. If it offers you the option to "Install" then click on that. Repeat this for "Adafruit Circuit Playground". The online tools should have all the libraries pre-installed.
3. Attach the NeoPixel strip to the Circuit Playground. The NeoPixel strip has three wires connected to it: power (marked as "+5 V"), ground (gnd), and signal (sgn or sgl or data). If you buy the strip recommended in the materials section, the wires come color-coded, but other strips often seem to pick wire colors at random. Luckily, you can look on the strip itself to see what wire does what. The recommended strip from Adafruit also comes with alligator clips attached, but if your strip doesn't have alligator clips, you can strip the wires and use your own alligator clips.
4. It is safer to only attach wires to a microcontroller when it isn't connected to power or a computer, so before attaching wires, you should disconnect the Circuit Playground from the computer and any other power source. Connect the power wire to a 3.3 V pad. Most NeoPixel strips will happily work at only 3.3 V, but if yours doesn't, you can connect the power wire to the VBATT pad and connect to USB power which is 5 V. Don't connect to anything higher than 6 V, though, as that will destroy the NeoPixel strip.
5. Connect the ground wire to a GND pad.
6. Connect the signal wire to pad marked #12. which is connected to digital pin 12. You can

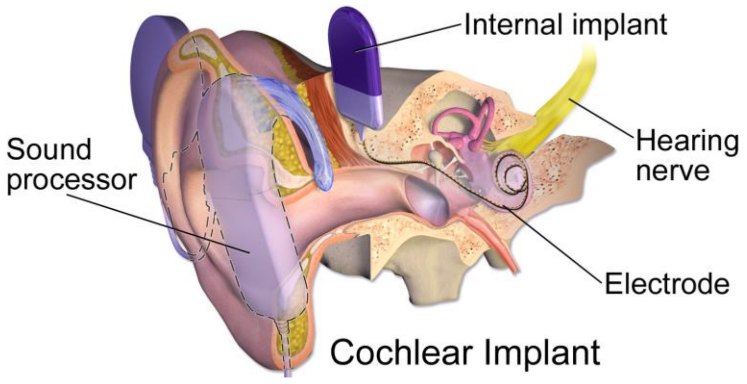


change which pin you use for the NeoPixels in the sketch.
7. After double checking your work, you can reconnect the Circuit Playground to your computer.

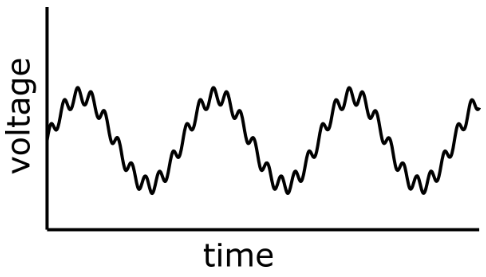
Background

The cochlea is the structure in the ear that converts mechanical motion conducted from the tympanum (eardrum) to an electrical signal in nerves sent to the brain. At the most basic level, inside the cochlea is long rolled-up structure called the basilar membrane. It's stiffness and size vary along its length such that it is most stiff and has the smallest fibers near the entrance and longer less stiff fibers near the end. Each part of the basilar membrane has a different natural frequency and vibrates in sympathy with the vibrations that have the same frequency. Specialized nerve cells called hair cells along the length of the membrane detect which spots are vibrating and send that information to the cochlear nerve and ultimately to the auditory nerve and brain.

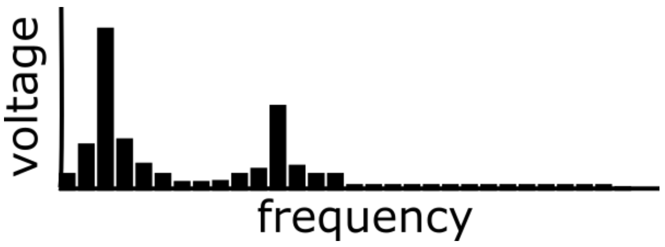
If a person has a problem in the vibratory conduction path or in the hair cells, a cochlear implant may be used to compensate. An external device receives sound waves and encodes the information into bands, often using an algorithm called the Fast Fourier Transform.



If we were to look at a typical signal from a microphone, it might look something like this



Although it is possible to look at this signal and see that it is composed of two sine waves on top of each other, it might be less obvious what the frequencies of those two signals are. Since the cochlea is activated by frequencies, we need to know which frequencies are there. That's where the Fast Fourier Transform (FFT) comes in. It is a series of equations that transforms a signal from having time as the x-axis to having frequency as the x-axis. A set of these algorithms comes with the Circuit Playground library. After using it, the data might look like this.



You will notice that the FFT is a bar graph. An FFT breaks up the signal into bins, usually of equal width, that cover a chosen range. In the Circuit Playground, the FFT library makes 32 bins of about 125 Hz in width.

Wires are inserted into the cochlea to stimulate ranges of the auditory nerve that correspond to the bands of frequency from the sound stimulus.

People who have had hearing and then use a cochlear implant say that it doesn't sound like their hearing before they became deaf. The basilar membrane and hair cells can detect thousands of individual frequencies, while the cochlear implant can only activate 20 or so bands of frequencies. The goal for the implants is improved sound perception and the recognition of words.

Making a First Attempt

Students don't really have to understand coding on an Arduino to do the engineering for this project. Novices could just download the sample code and move forward to characterizing. For some advanced groups, you might only need to give the groups the background about cochlear implants and a hint that the Circuit Playground and Express can do FFTs and ask them to write their own code. For students inbetween, they can look at the existing sample code and modify it to suit their needs.

The most current sample code can be found here
https://create.arduino.cc/editor/fftresonance/c72ee82f-11e8-421b-9fbd-4cb75add5c33/preview

How the Code in the Sample Sketch Works

```
// Cochlear Implant Model
// This software uses code from Adafruit's Mike Barela's
// FFT Test Program for Circuit Playground

#include <Adafruit_NeoPixel.h> // loads NeoPixel Library
#define PIN    12 // data wire is connected to Pin 12
#define NUMPIXELS 30 // strip is 30 pixels long

#include <Adafruit_CircuitPlayground.h> // loads library for using the Circuit Playground
#define BINS   32           // the CP has 32 FFT bins
#define FRAMES 4            // data is average over frames. More frames are more accurate but
                            // miss fast sounds
```

```
#define NOISE  5           // subtracts a fixed amount from each average due to
                           // ambient and electronic noise
```

The first part of the code imports libraries for the NeoPixels and the Adafruit Circuit Playground and gives names to some constants used in the sketch. A library is a set of premade routines that extends the Arduino environment by adding new commands. The constants are numbers that are used in various places in the sketch. By giving them names, it is easy to change them.

```
Adafruit_NeoPixel pixels = Adafruit_NeoPixel(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);
```

SInce it is possible to connect many NeoPixel LED strips to a Circuit Playground, the libraries have the code create an instance and give it a name so that the code will know which strip the code is referring to. The code has called the strip `pixels`, said that is `NUMPIXELS` (30) long at that we have connected it to `PIN` (12). The rest of the information specifies what kind of strip you have, but most strips these days are configured as listed.

```
void setup() {                // runs once
  CircuitPlayground.begin();  // set up the board library
  pixels.begin();             // initializes the NeoPixel library
  pixels.show();              // clears NeoPixels
  Serial.begin(9600);         // sends info to the serial monitor at 9600 baud
}
```

Setup runs only once. It turns on the Circuit Playground and sets up the string of LEDs called pixel and then clears them to make sure all are off when the program starts. The last command sets up communication back to the serial monitor in the Arduino IDE or the online editor which can display the data. To see the data, click either the magnifying glass icon in the Arduino IDE or online editor. The data can be copy and pasted from the serial monitor and pasted into something like Microsoft Excel or Google Sheets.

```
void loop() {                 // runs over and over
```

Loop runs over and over unless the Circuit Express is unplugged from power or the reset button is pressed.

```
  uint8_t i,j;                 // creates 8 bit counting variable i and j
```

Two variables i and j are created. `uint8_t` means that each variable is an integer from 0-255 (8 bits).

```
  uint16_t spectrum[BINS];     // creates 16 bit array for measured FFT values
  uint16_t avg[BINS];          // creates 16 bit array for average FFT values
```

Two arrays of variables. Each array has `BINS` (32) elements. Each element is an integer from 0 to 65,545.

```
// Gathering data FRAMES times, adding together,
// and dividing by FRAMES to find the average
for(j=1; j <= FRAMES; j++) {
   CircuitPlayground.mic.fft(spectrum);  // collects FFT data in spectrum array
```

This calls the Circuit Playground library and run the FFT routine which listens with the microphone, runs an FFT creating values in 32 bins, and outputs the results into the array `spectrum`. The `for` command has the code repeat the measure `FRAMES` number of times to average out errors.

```
   for(i=0; i < BINS; i++) {             // loop takes that data and adds it together
     if(spectrum[i] > 255) spectrum[i] = 255; // limit outlier data
     if(i == 0)                          // on the first loop just enter the data
       avg[i] = spectrum[i];
     else
       avg[i] = avg[i] + spectrum[i];    // next loops add the new data to what is there
   }
}
for(i=0; i < BINS; i++) {               // go through each BIN
   avg[i] = avg[i] / FRAMES;            // divide about the number of FRAMES to average
}

//Find the maximum values
int maxVal = 0, maxIndex = 0;
for(i=0; i < BINS; i++) {               // For each output bin average
   if(avg[i] >= maxVal) {               //  find the peak value
     maxVal = avg[i];
     maxIndex = i;                      //  and the bin that max value is in
   }
}

for(j=0; j < 32; j++) {                // print average FFT for all BINS
   if (NOISE>avg[j]) {                 // the value is less than noise
    avg[j]=0;                          // set the value to zero
   }
   else {                              // otherwise
    avg[j]=avg[j]-NOISE;               // subtract the noise level
   }
```

Both the FFT computations themselves and real electrical noise can cause the values of the bins to be not zero even when no signal exists in those bins. This section of the sketch subtracts a predetermined amount, `NOISE`, from each bin to account for this noise.

```
   Serial.print(avg[j]);               // print the value of that BIN
   Serial.print(" ");
   pixels.setPixelColor(j,pixels.Color(0,avg[j],0));  // set brightness on NeoPixel to
                                                      // average value
}
```

Finally, the data is output so that you can see what the code has done. The

`Serial.print(avg[j]);` command prints the data from the jth FFT bin to the serial monitor which you can look at by pressing the magnifying glass icon in either the Arduino IDE or online environment.

`pixels.setPixelColor(j,pixels.Color(0,avg[j],0));` sets the jth pixel to the output of the average value jth FFT bin. The three numbers are values of red, green, and blue. So, in this case, we are setting the green brightness and leaving the other two colors off.

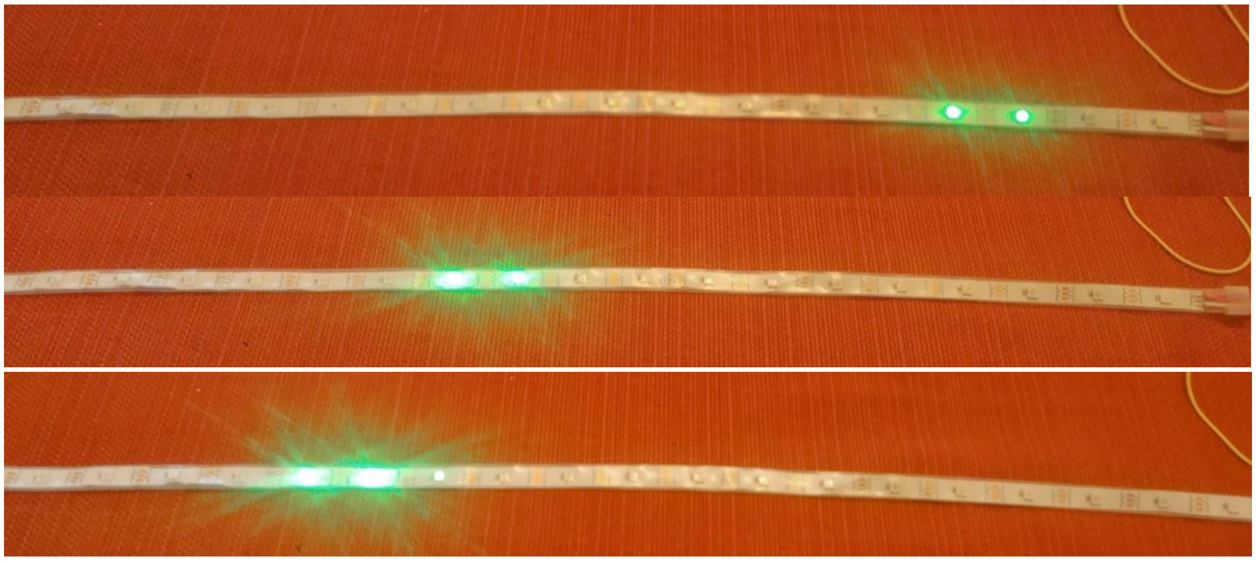
```
  pixels.show();                  // NeoPixels only change when this command is given

  Serial.print(" ");
  Serial.print("Max Value = "); Serial.print(maxVal);  // print the highest value
  Serial.print(", Index of Max Value = "); Serial.println(maxIndex); //and which BIN
}
```

Characterizing the Output

Have students play a school appropriate song through a small speaker (like from a cell phone) next to the microphone on the Circuit Playground while the code is running. Ask the students to record how the LED strip responds to the changing sounds.

Typically students will report that different parts of the strip light up at different times, but it is hard to tell exactly what is going on.

Playing only one frequency at a time can be more revealing. Have students use a computer or their phone to generate a test tone at a frequency 100 hertz and play that into the microphone. There are many apps that can do this; for example, http://onlinetonegenerator.com/ works on most computers, tablets, and phones. Then, they should play 200 hertz, 300 hertz, 400 hertz, and so on up the scale to about 4000 hertz.



Most students will notice that the brightest group of LED moves along the strip. Similarly, the data on the serial monitor goes up as the frequencies increase.

Getting Clearer Data

During characterizing, the students may have noticed that many LEDs turned on even when they were playing only one frequency. Even when the room is very quiet, some of the LEDs will light up. Both the FFT computations themselves and real electrical noise can cause the values of the bins to be higher than zero even when no real signal exists in those bins. In the code, we can subtract a constant value from all the bins to make sure that the LEDs go off when there is no sound to record.

Students can change the value of

```
#define NOISE  5
```

at the start of the code. A higher value means that a sound needs to be louder before it will light an LED or be recorded in the serial monitor. However, a higher value will make cut out quieter sounds. Some balance will be necessary.

In addition, during characterizing, students might see different LEDs light for the same frequency. For a variety of reasons, FFT measurements can be wonky and can yield spurious results from time to time. Running the FFT more than once can average out errors at the expense of observing short duration sounds. By adjusting the value of

```
#define FRAMES 4
```

they can alter the number of times the FFT result is averaged before the value is displayed. Higher values yield more consistent results be miss short sounds.

Figuring out a Word
Ask a student in each group to record themselves saying two different two-syllable words. The student should then play them back to the microphone on the Circuit Playground.

Ask students to describe how the words look on the LEDs. They should see a pattern of different LEDs lighting at different times. The patterns are fast, and often complicated, but they should notice that the pattern is different for different words.

Students should try different words as well. They should notice that many words have unique patterns. Have a different person in the group repeat the words. How is the pattern different for different people? How is the pattern the same?

After a while, students will develop a method of distinguishing between words. Encourage them to adjust the code, if necessary, to make recognizing words easier.

To test their method, have students put their hands over their ears, use earplugs, or play music over headphones so that they cannot hear what is played into the microphone. Use the recording to show different patterns. Can they recognize which word goes with which pattern? Can they recognize the word when a different person says it?

Part III: Evaluation

Groups should report out how well they could recognize LED patterns created by words. Some groups will be much better at this than others, and that is consistent with users of cochlear implants. Some people with cochlear implants are much better at being able to recognize speech than others.

Groups should also indicate what they did to improve recognition. Some will have adjusted the NOISE and FRAMES parameters in the code. Others might have recorded the LED patterns and looked at them in slow motion.

In addition, students could be asked to draw a model showing how the Circuit Playground and LEDs models a cochlear implant.

In addition, each group should report out on how well they worked together. Even for classes that didn't have time for the groups to work on their own project, having the students briefly present their work to their classmates tends to give the best opportunity to figure out what happened in their group. They should explain

- What their problem/goal was
- What they tried
- Whether or not it was successful
- How they could tell if it was working
- What they did if they didn't all agree on what to do

- How often did they get to put a mark on their checklists

Cochlear Implant drawing by Bruce Blaus. Used with permission.