**Plant Bio-acoustic Sound Generation**

**Options**

1. **Standard EMG Sensor Module with Built-In AD8226:** This option offers a balance of simplicity and effectiveness. The EMG module is specifically designed to handle bioelectric signals, making it a reliable choice for this type of project.



**Option 1. Standard EMG Module with in-built AD8226**

1. **Standard ECG Module with Built-In AD8232:** Similar to the EMG module, an ECG module is designed to handle bioelectric signals. The AD8232 has built-in filters that can help reduce noise and artifacts, which could be beneficial for this application.
2. **Generic Headphone to TRRS Module to AD620:** This option provides you with more control over your setup and might allow for more fine-tuning, but it's also more complex and has a greater chance of setup errors. The AD620 is a solid instrumentation amplifier, but this option requires a good understanding of electronics to implement correctly.



**Option 3: Generic Headphone to TRRS Module to AD620**

In a TRRS (Tip, Ring, Ring, Sleeve) 3.5mm audio jack, the connections usually stand for Left Audio, Right Audio, Ground, and Microphone. The order can vary depending on the specific jack, but one common configuration is:

Tip: Left Audio

Ring1: Right Audio

Ring2: Ground

Sleeve: Microphone

For this setup, you could use either the Left or Right Audio for your signal, and use the Ground as your reference. The Microphone channel would not be used.

Here's a general guide to connect the TRRS 3.5mm Jack, AD620 breakout board, and Arduino Mega:

1. Connect the Tip (Left Audio) of the TRRS Jack to the IN+ pin on the AD620 breakout board.
2. Connect the Ring1 (Right Audio) of the TRRS Jack to the IN- pin on the AD620 breakout board.
3. Connect the Ring2 (Ground) of the TRRS Jack to the GND pin on the AD620 breakout board, and also to a GND pin on the Arduino Mega.
4. Connect the V+ or Vcc pin on the AD620 breakout board to a +5V pin on the Arduino Mega.
5. Connect the OUT pin on the AD620 breakout board to an Analog Input pin on the Arduino Mega (A0 - A15).

This setup assumes that the EMG patches are connected to the Left and Right Audio channels on the TRRS jack. If this is not the case, you'll need to adjust the connections accordingly.

Again, these instructions are quite general and may not apply exactly to your specific boards. Always refer to the datasheets or user manuals of your components for the most accurate information. Also, please ensure your experiments with plants are safe and non-damaging to them.

|  |
| --- |
| MIDI  <https://github.com/FortySevenEffects/arduino_midi_library>  <https://github.com/bitluni/arduinoMIDISynth>  <https://github.com/sensorium/Mozzi>  <https://www.arduino.cc/reference/en/libraries/audiozero/> only works with Allows playing audio files from an SD card. For Arduino Zero, MKRZero and MKR1000 only. |

|  |
| --- |
| Project  <https://jaworskygeek.blogspot.com/2017/09/how-to-connect-speaker-to-arduino.html> |

|  |
| --- |
| Stereo  <https://www.adafruit.com/product/1788>  Adafruit Music Maker Shield  SparkFun MP3 Player Shield |

In this example, the analog value from the AD620 is read and then mapped to a MIDI note value between C2 (36) and C7 (96). A Note On message is then sent with this note value, followed by a delay and then a Note Off message. This will create a sequence of MIDI notes based on the signal from the plant.

This is a very basic example and does not include any filtering or processing of the signal, so the output might not be very musical. You could extend this code by adding more complex mapping or processing to make the output more interesting or musical.

Keep in mind that the MIDI library will not produce sound itself. You'll need to connect the Arduino to a MIDI synthesizer or to a computer running a software synthesizer to actually hear the notes.

Remember to ensure your experiments with plants are safe and non-damaging to them.

However, if you only have one bio-signal source and still want to generate polyphonic sound, you would need to devise a scheme to split or interpret the signal in a way that can control multiple tones. One possibility could be to use the amplitude of the signal to control one tone and the rate of change of the signal (or some other characteristic) to control another tone.

Given that, the complexity of creating polyphonic sounds directly from a single bio-signal source in real-time might exceed the capabilities of the Arduino, especially if you want high-quality sounds. As previously suggested, for high-quality polyphonic sound synthesis, you might want to consider using more advanced sound-generation hardware or sending the data to a computer or another device that is better equipped for sound synthesis.

Here's a simplified example of how you could generate polyphonic sound with a single sensor input. This example will use the **Mozzi library** and an Arduino. The sensor reading will be used to control the frequency of one oscillator, and the rate of change of the sensor reading will control the frequency of a second oscillator.

|  |
| --- |
| #include <MozziGuts.h>  #include <Oscil.h>  #include <tables/sin2048\_int8.h>  // Instantiate two Oscil objects, for two tones  Oscil<SIN2048\_NUM\_CELLS, AUDIO\_RATE> osc1(SIN2048\_DATA, 0), osc2(SIN2048\_DATA, 0);  // Variable to store the last sensor reading  int lastSensorValue = 0;  void setup(){  startMozzi(AUDIO\_RATE\_16384); // start with a 16384 Hz audio rate.  }  void updateControl(){  // Read from your sensor  int sensorValue = analogRead(A0);  // Map the sensor value to a frequency for the first oscillator  float freq1 = map(sensorValue, 0, 1023, 50, 1000); // Adjust as necessary  osc1.setFreq(freq1);  // Calculate the rate of change of the sensor value and map it to a frequency for the second oscillator  int rateOfChange = sensorValue - lastSensorValue;  float freq2 = map(abs(rateOfChange), 0, 1023, 50, 1000); // Adjust as necessary  osc2.setFreq(freq2);  // Store the current sensor value for the next loop  lastSensorValue = sensorValue;  }  int updateAudio(){  return (osc1.next() + osc2.next()) >> 1; // Mix the two oscillators by averaging their outputs  }  void loop(){  audioHook(); // required here  } |

This is still a very simple example and the sound generation might not be very musical or symphonic. You would likely need to do additional processing on the sensor data to create a more musical result.

Please note that the quality of sound directly from the Arduino's PWM output can be quite limited. For more complex and high-quality sound generation, you might want to consider using more advanced sound-generation hardware or sending the data to a computer or another device for sound synthesis.

To connect a **speaker** to an Arduino, you typically need a **digital-to-analog converter (DAC)** or an **audio amplifier** to properly drive the speaker.

Here's an updated code example that includes the connection for a speaker using an external audio amplifier:

|  |
| --- |
| #include <MozziGuts.h>  #include <Oscil.h>  #include <tables/sin2048\_int8.h>  // Instantiate two Oscil objects, for two tones  Oscil<SIN2048\_NUM\_CELLS, AUDIO\_RATE> osc1(SIN2048\_DATA, 0), osc2(SIN2048\_DATA, 0);  // Variable to store the last sensor reading  int lastSensorValue = 0;  // Digital pin connected to the audio amplifier or DAC  const int speakerPin = 9;  void setup(){  startMozzi(AUDIO\_RATE\_16384); // start with a 16384 Hz audio rate.  pinMode(speakerPin, OUTPUT); // Set the speaker pin as an output  }  void updateControl(){  // Read from your sensor  int sensorValue = analogRead(A0);  // Map the sensor value to a frequency for the first oscillator  float freq1 = map(sensorValue, 0, 1023, 50, 1000); // Adjust as necessary  osc1.setFreq(freq1);  // Calculate the rate of change of the sensor value and map it to a frequency for the second oscillator  int rateOfChange = sensorValue - lastSensorValue;  float freq2 = map(abs(rateOfChange), 0, 1023, 50, 1000); // Adjust as necessary  osc2.setFreq(freq2);  // Store the current sensor value for the next loop  lastSensorValue = sensorValue;  }  int updateAudio(){  int output = (osc1.next() + osc2.next()) >> 1; // Mix the two oscillators by averaging their outputs  analogWrite(speakerPin, output); // Send the audio output to the speaker  return output;  }  void loop(){  audioHook(); // required here  } |

Can I use LM386 audio amplifier?

ChatGPT

Yes, you can use the LM386 audio amplifier to amplify the audio signal generated by the Arduino and drive a speaker. The LM386 is a commonly used and easily accessible audio amplifier IC that is suitable for small audio applications.

Here's an example of how you can connect the LM386 audio amplifier to the Arduino to drive a speaker:

In this setup, you connect the audio output pin of the Arduino to the audio input pin (IN+) of the LM386 amplifier using the audioInPin. The audioOutPin is connected to the speaker pin (OUT) of the LM386 amplifier. The audio signal generated by the Arduino's PWM output is then amplified by the LM386 and passed to the speaker.

Please refer to the LM386 datasheet and the documentation of your specific LM386 module for further details on wiring and any additional components required. Also, ensure that the power supply and voltage levels are within the specifications of the LM386.

|  |
| --- |
| cpp  Copy code  #include <MozziGuts.h>  #include <Oscil.h>  #include <tables/sin2048\_int8.h>  // Instantiate two Oscil objects, for two tones  Oscil<SIN2048\_NUM\_CELLS, AUDIO\_RATE> osc1(SIN2048\_DATA, 0), osc2(SIN2048\_DATA, 0);  // Variable to store the last sensor reading  int lastSensorValue = 0;  // Pin connections for LM386 audio amplifier  const int audioInPin = A0; // Connect to the audio output pin of the Arduino  const int audioOutPin = 9; // Connect to the audio input pin (IN+) of the LM386  const int speakerPin = 10; // Connect to the speaker pin (OUT) of the LM386  void setup() {  startMozzi(AUDIO\_RATE\_16384); // start with a 16384 Hz audio rate.  pinMode(audioInPin, INPUT);  pinMode(audioOutPin, OUTPUT);  pinMode(speakerPin, OUTPUT);  }  void updateControl() {  // Read from your sensor  int sensorValue = analogRead(A0);  // Map the sensor value to a frequency for the first oscillator  float freq1 = map(sensorValue, 0, 1023, 50, 1000); // Adjust as necessary  osc1.setFreq(freq1);  // Calculate the rate of change of the sensor value and map it to a frequency for the second oscillator  int rateOfChange = sensorValue - lastSensorValue;  float freq2 = map(abs(rateOfChange), 0, 1023, 50, 1000); // Adjust as necessary  osc2.setFreq(freq2);  // Store the current sensor value for the next loop  lastSensorValue = sensorValue;  }  int updateAudio() {  int audioInput = analogRead(audioInPin);  analogWrite(audioOutPin, audioInput);  int output = (osc1.next() + osc2.next()) >> 1; // Mix the two oscillators by averaging their outputs  analogWrite(speakerPin, output); // Send the audio output to the LM386  return output;  }  void loop() {  audioHook(); // required here  } |

**Comparison of Amplifiers**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Amplifier | Operating Voltage | Amplification: Adjustable | Signal Input Voltage Range | Signal Output Range | Gain Range | Application |
| AD620 Instrumentation Amplifier | ±2.3 V to ±18 V | Yes | ±10 V (for a ±15 V supply) | Depends on supply voltage and load, typically close to supply rails | 1 to 10,000 | Precision data acquisition, medical instrumentation |
| INA128/INA129 Instrumentation Amplifier | ±2.25 V to ±18 V | Yes | ±10 V (for a ±15 V supply) | Depends on supply voltage and load, typically close to supply rails | 1 to 10,000 | Precision data acquisition, medical instrumentation |
| AD8232 Heart Rate Monitor | 1.8 V to 3.3 V | Yes (with external resistors) | -0.3 V to +0.3 V | Analog output, depends on load and supply voltage | Adjustable with external resistors | ECG and fitness applications |
| PGA2311 Stereo Audio Volume Control | ±2.5 V to ±15 V | Yes (digital control) | ±5 V (for a ±15 V supply) | Depends on supply voltage and load, typically close to supply rails | -95.5 dB to +31.5 dB | Audio applications |
| OpenBCI Cyton Biosensing Board | Powered via USB | Yes (digital control) | Depends on the specific configuration | Depends on the specific configuration | 1, 2, 4, 6, 8, 12, 24 (times) | EEG, EMG, ECG biosensing |

**USE EMG SENSOR MODULE TO MEASURE PLANT BIO-ACOUSTIC SIGNAL**

Bill of Materials (BOM)

|  |
| --- |
| LM386 Audio Amplifier Module |
| Advance Technologies EMG Muscle Sensor V3.0 With Cable And Electrodes |
| 9V Original HW High-Quality Battery-10Pcs/Box |
| 9V 10cm Battery Connector |
| Adafruit Accessories Muscle Sensor Surface EMG Electrodes - H124SG Covidien - Pack of 6 |
| AD620 Microvolt/Millivolt Voltage Amplifier Module |
| CJMCU TCA9548A I2C 8 Channel Multiple Extensions Development Board |
| 3W 8Ohm 2Inch 53mm Speaker |