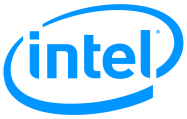
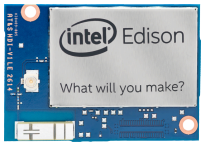
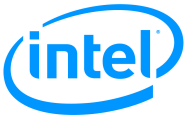




Audio Interface



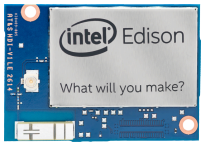
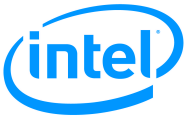
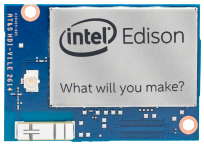


Table of Contents

Revision history		
Version	**Date**	**Comment**
1.0	4/11/2016	Initial release





Introduction

In this tutorial, you will:

1. Set up a microphone and a speaker on the Edison,
2. Record and play audio, and
3. Perform speech recognition and voice synthesis.

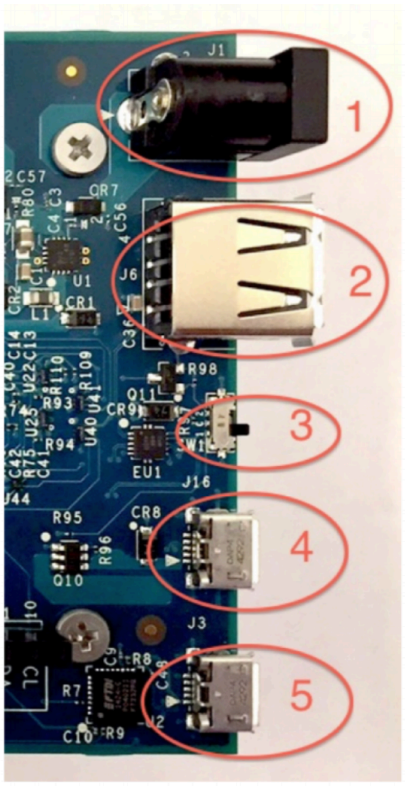
Things Needed

1. An Intel Edison with Arduino-compatible breakout,
2. A powered USB hub or a power adapter,
3. An ALSA-compatible USB sound card,
4. A microphone with a 3.5 mm audio jack,
5. A speaker/headphone with 3.5 mm audio jack,
6. A Grove Starter kit, and
7. A PC or Mac

USB Audio Interface Setup

Hardware Setup

With the Arduino-compatible breakout board, the Intel Edison can accept peripheral via its USB 2.0 host port.



1) **Barrel Connector for Power**

2) **USB 2.0 Host Port**
- Accepts peripherals such as a mouse, a keyboard, and a webcam.

3) **Switch**
- Upper position: USB Host mode – USB 2.0 Host Port is enabled.
- Lower position: Device mode – Multi-gadget Micro USB port and UART serial USB port are enabled.

4) **Multi-gadget Micro USB Port** lets you
- Supply 5V power via USB, but if you are using more power intensive features, you need to use the barrel connector to supply more power,
- Interact with Arduino IDE,
- Use ethernet over USB, and
- Write to the on-board flash memory.

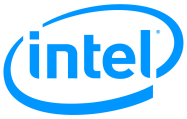
5) **UART Serial USB port**

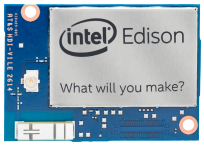
Figure 1 Port

In order to use this port, the Intel Edison must be in USB host mode. With this configuration the Edison can longer be powered through the multi-gadget micro USB port. However, the USB 2.0 host port can also be used to supply 5V power via USB. We can connect a power USB hub to the host port to supply power and accept peripherals.

The following guidance will help set up the hardware for the audio interface.

1. Move the switch to the upper position to enable USB host mode.
2. Do one of the following options:
    a. *Option 1*
        i. Connect a power adapter to the barrel connector on the breakout board.
        ii. Connect a USB sound card to the USB 2.0 host port.





b. Option 2

i. Connect the male end of a power USB hub to the USB 2.0 host port on the breakout board.

ii. Connect a USB sound card to the powered USB hub.

3. Connect a microphone and a speaker/headphone to the USB sound card.
4. Let's check whether the sound card is seen by the Edison.
5. SSH or serial connect to the Edison.
6. Disconnect the sound card
7. **$ lsusb**

   We will see the following output. (note: If you are using a USB hub, the USB hub should also be displayed.) **lsusb** is a Linux command that displays all USB devices.

```
root@chris_edison:~# lsusb
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
```

Figure 2 List of USB Devices

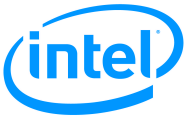
8. Connect the sound card and enter "**lsusb**" again.

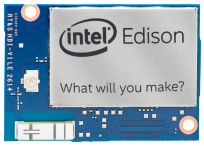
   The output should now include the sound card.

```
root@chris_edison:~# lsusb
Bus 001 Device 016: ID 0d8c:0008 C-Media Electronics, Inc.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 002 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
```

Figure 3 List of USB Devices

Note that the sound card used in this example is C-Media USB Audio Device. If you do not see your sound card, check if your sound card is ALSA-compatible. (note: You can find ALSA-compatible devices at http://www.alsa-project.org/main/index.php/Matrix:Main.)





Software Setup

In order to interact with the USB sound card, we will use the Advanced Linux Sound Architecture (ALSA), which provides audio and MIDI functionality to Linux.

1. **$ opkg install alsa-utils**
2. **$ aplay -l**

   This command lists playback devices seen by ALSA. The output should be similar to the one below.

```
root@chris_edison:~# aplay -l
**** List of PLAYBACK Hardware Devices ****
card 0: Loopback [Loopback], device 0: Loopback PCM [Loopback PCM]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 0: Loopback [Loopback], device 1: Loopback PCM [Loopback PCM]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 1: dummyaudio [dummy-audio], device 0: Edison Audio (*) []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: dummyaudio [dummy-audio], device 1: ((null)) []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: dummyaudio [dummy-audio], device 2: ((null)) []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: Device [C-Media USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Figure 4 List of Playback Devices

Note that the sound card used in this example is C-Media USB Audio Device, which is recognized as card 2, device 0.

3. **$ arecord -l**

This command lists capture (recording) devices seen by ALSA.

```
root@chris_edison:~# arecord -l
**** List of CAPTURE Hardware Devices ****
card 0: Loopback [Loopback], device 0: Loopback PCM [Loopback PCM]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 0: Loopback [Loopback], device 1: Loopback PCM [Loopback PCM]
  Subdevices: 8/8
  Subdevice #0: subdevice #0
  Subdevice #1: subdevice #1
  Subdevice #2: subdevice #2
  Subdevice #3: subdevice #3
  Subdevice #4: subdevice #4
  Subdevice #5: subdevice #5
  Subdevice #6: subdevice #6
  Subdevice #7: subdevice #7
card 1: dummyaudio [dummy-audio], device 0: Edison Audio (*) []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: dummyaudio [dummy-audio], device 1: ((null)) []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 1: dummyaudio [dummy-audio], device 2: ((null)) []
  Subdevices: 1/1
  Subdevice #0: subdevice #0
card 2: Device [C-Media USB Audio Device], device 0: USB Audio [USB Audio]
  Subdevices: 1/1
  Subdevice #0: subdevice #0
```

Figure 5 List of Capture Devices

4. **$ vi ~/.asoundrc**

   In most cases, your Linux system should be able to record and play audio without any configuration. However, the Edison may not work without a configuration file. The **.asoundrc** file and **/etc/asound.conf** are the configuration files for ALSA drivers. We will write the **.asoundrc** file for our application.

5. Enter the following and save. Replace the card and the device numbers with what you found from Step 2 and 3.

```
defaults.ctl.card 2
defaults.pcm.card 2
defaults.pcm.device 0
```

Basic Operation: Record and Play

arecord is a command-line audio recorder for the ALSA soundcard driver. It captures audio from a sound card and saves it as a file. It supports several file formats. **aplay** is a command-line audio player for the ALSA soundcard driver. It reads from a file and plays the audio. This section demonstrates how to use these commands.

1. **$ arecord test.wav**

   The Edison begins to record audio as soon as it prints *"Recording WAVE 'test.wav' : Unsigned 8 bit, Rate 8000 Hz, Mono"*.

```
root@chris_edison:~# arecord test.wav
Recording WAVE 'test.wav' : Unsigned 8 bit, Rate 8000 Hz, Mono
```

   Figure 6 arecord Output

   This command records audio as a wave file in unsigned 8-bit sample format with sample rate at 8 kHz.

2. Press **Ctr-C** to stop recording.

3. **$ aplay test.wav**

   This command plays the wave file. Notice that the audio quality is very low. The audio quality heavily depends on the sample format and the sample rate since they determine the amount of audio information to be stored. The more the stored information, the better the audio interface can approximate the original signal. Let's try a different sample format.

4. **$ arecord -f S16_LE test.wav**

   -f option lets you choose the sample format. The default value is **U8** (unsigned 8-bit). **S16_LE** is 16 bit little endian.

```
root@chris_edison:~# arecord -f S16_LE test.wav
Recording WAVE 'test.wav' : Signed 16 bit Little Endian, Rate 8000 Hz, Mono
```

   Figure 7 arecord Output

5. **$ aplay test.wav**

   The audio quality is improved since more bits are used to represent each sample. Now, let's try to further improve the quality by increasing the sample rate.

6. **$ arecord -f S16_LE -r44100 test.wav**

   -r option lets you choose the sample rate. The default value is 8 kHz. Here, we chose 44.1 kHz. (note: 44.1 kHz/16 bit is the audio CD quality)

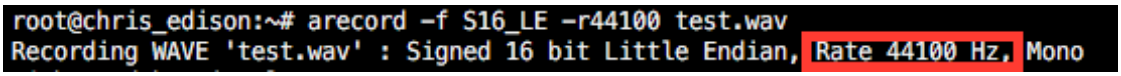


Figure 8 arecord Output

7. **$ aplay test.wav**

8. You can record audio as stereo by choosing two channels. Enter **"arecord -f S16_LE -r44100 -c2 test.wav"**

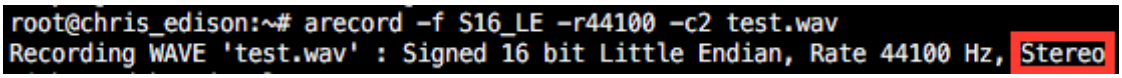
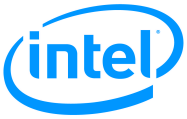
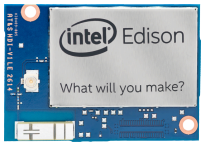


Figure 9 arecord Output

9. There are format shortcuts. The command in Step 8 is equivalent to "**arecord -f cd test.wav**" (record with CD quality).

10. If you want to see all available options, enter "**arecord -h**" and similarly, "**aplay -h**".



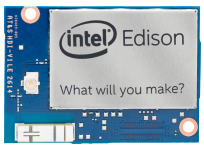


Human Interaction: Speech Recognition and Voice Synthesis

The audio interface on the Edison enables development of numerous applications. As an example, we will build a human interaction system using speech recognition and voice synthesis. Using this human interaction system, we can talk to the Edison and the Edison responds back to us. We will use a Python package called **SpeechRecognition**, which supports several engines and APIs including Google Speech Recognition, CMU Sphinx, Wit.ai, Microsoft Bing Voice Recognition, api.ai, and IBM Speech to Text. Google Speech Recognition will be used in this demo.

We will use an open source software voice synthesizer called **eSpeak**, which uses "formant synthesis" method and hence the size of the software is small.

1. First, we need to install **setuptools**, which lets us easily install/uninstall Python packages.
2. **$ wget --no-check-certificate https://bootstrap.pypa.io/ez_setup.py**
3. **$ python ez_setup.py --insecure**
4. Now, we will install **SpeechRecognition** package.
5. **$ wget --no-check-certificate https://pypi.python.org/packages/source/S/SpeechRecognition/SpeechRecognition-3.3.1.tar.gz**
6. **$ tar -xvzf SpeechRecognition-3.3.1.tar.gz**
7. **$ cd SpeechRecognition-3.3.1**
8. **$ python setup.py install**
9. **$ cd ..**
10. Google Speech Recognition only accepts audio as a FLAC format. Thus, we need to install FLAC codec.
11. **$ wget http://downloads.xiph.org/releases/flac/flac-1.3.1.tar.xz**
12. **$ tar -xvf flac-1.3.1.tar.xz**
13. **$ cd flac-1.3.1**
14. **$ ./configure --prefix=/usr --disable-thorough-tests && make**
15. **$ make install**
16. **$ cd ..**



17. Speech recognition system is ready. Let's try it. Enter "**arecord -f cd speech.wav**" and say anything in English (e.g. "Hello") to the microphone. Then, press **Ctr-C** when you done.

18. **$ vi speech2text.py**

19. Type the following Python code and save.

```
from os import path
import speech_recognition as sr

WAV_FILE = path.join(path.dirname(path.realpath(__file__)), "speech.wav")

r = sr.Recognizer()
with sr.WavFile(WAV_FILE) as source:
    audio = r.record(source)
try:
    print r.recognize_google(audio)
except sr.UnknownValueError:
    print("Google Speech Recognition could not understand audio")
except sr.RequestError as e:
    print("Could not request results from Google Speech Recognition service;
{0}".format(e))
```

20. **$ python speech2text.py**

21. Finally, let's install **eSpeak**.

22. **$ vi /etc/opkg/base-feeds.conf**

23. Add the following lines to the configuration file.

```
src/gz all http://repo.opkg.net/edison/repo/all
src/gz edison http://repo.opkg.net/edison/repo/edison
src/gz core2-32 http://repo.opkg.net/edison/repo/core2-32
```

24. **$ opkg update**

25. **$ opkg install espeak**

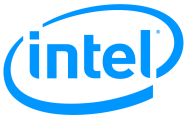
26. Let's try voice synthesis. Enter "**espeak hello**".

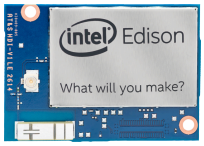
27. Now, let's implement the human interaction system.

28. Insert a Grove base shield and connect a button to D2.

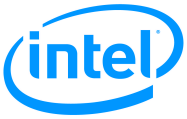
29. **$ vi human_interaction.py**

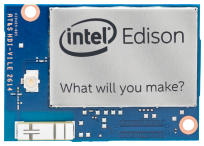
30. Type the following Python code and save.






```
import mraa
import os
import speech_recognition as sr
from os import path
import socket

REMOTE_SERVER = "www.google.com"

record = 'arecord -f cd STT.wav' # SST "Speech-to-Text"
stop_record = 'killall arecord'
play = 'play TTS.wav' # TTS "Text-to-Speech"
stop_play = 'killall aplay'

WAV_FILE = path.join(path.dirname(path.realpath(__file__)), "STT.wav")

# This handler runs when the button is released.
def handler(args):
        print 'Ok.'
        os.system(stop_record)

# Set up a button
button = mraa.Gpio(2)
button.dir(mraa.DIR_IN)
button.isr(mraa.EDGE_FALLING, handler, handler)

# Check the Internet connection
def is_connected():
        try:
                host = socket.gethostbyname(REMOTE_SERVER)
                s = socket.create_connection((host, 80), 2)
                return True
        except:
                pass
        return False

# Convert speech to text
def speech2text():
        r = sr.Recognizer()
        with sr.WavFile(WAV_FILE) as source:
                audio = r.record(source)
        try:
                return r.recognize_google(audio)
        except sr.UnknownValueError:
                print("Google Speech Recognition could not understand audio")
        except sr.RequestError as e:
                print("Could not request results from Google Speech Recognition
service; {0}".format(e))

# Simple conversation
def react(text):
       if (text == "hello"):
              return "Hello."
       elif (text == "what is your name") or (text == "what's your name"):
              return "My name is Edison."
       else:
              return "Sorry. I do not understand."

# Use voice synthesizer to generate speech
def speak(text):
        if text is None:
```






```
                print 'no text to be converted.'
        else:
                command = 'espeak \'' + text + '\''
                os.system(command)

def control():
        text = None
        if (button.read() == 1):
                print 'Listening...'
                os.system(record)
                print 'Processing...'

                text = speech2text()
                print "\t%s" % text
                response = react(text)
                speak(response)

if (is_connected() == True):
       print "We have the Internet connection!"
       while (1):
              control()
else:
       speak('Sorry. I need the Internet to understand you.')
```

31. **$ python human_interaction.py**

32. You may add more elif statements to react() function to expand the system.