
Make:
Sensors
Projects and Experiments to Measure the World with Arduino and Raspberry Pi
Tero Karvinen, Kimmo Karvinen & Ville Valtokari

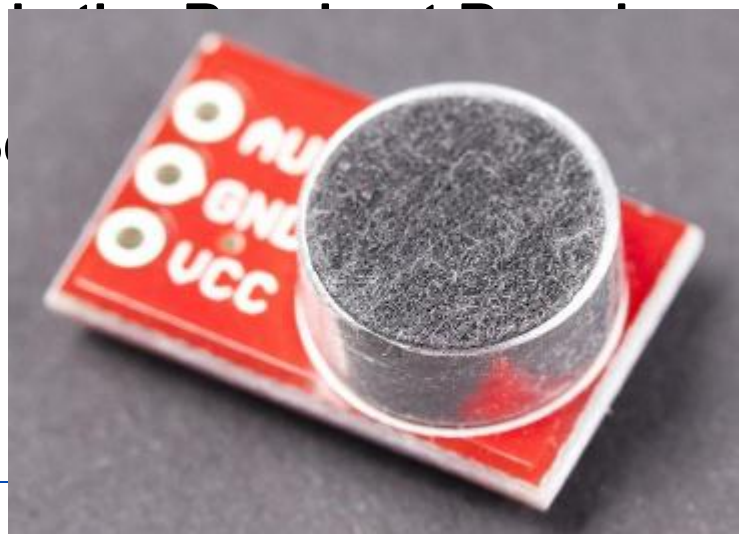
Sound

Sound

- Sound waves in the air are a compression and decompression of the medium. This movement can be detected and analyzed.
- Raspberry Pi can record and analyze sounds. In one project, you'll use a fast Fourier transform (FFT) to split sound into frequencies. This calculation can extract frequencies from any wave. With Arduino, you'll use a microphone to measure volume.

Experiment: Hearing Voices/Volume Level

- A microphone can detect sound level. This first experiment simply reads and prints values from a microphone.
- For many projects, you probably want to manipulate the numbers read from the microphone. You could set a threshold, apply a moving average, or detect minimum and maximum values. Later examples in this chapter let you try setting a threshold for sound. The component we used [illegible] for Electret Microphone (BOB-09964) from Sparkfun. S[illegible]



Microphone Breakout Code and Connection for Arduino

- Figure 11-2 shows the Arduino circuit for the microphone. Wire it up as shown, and then run the code shown in Example 11-1.

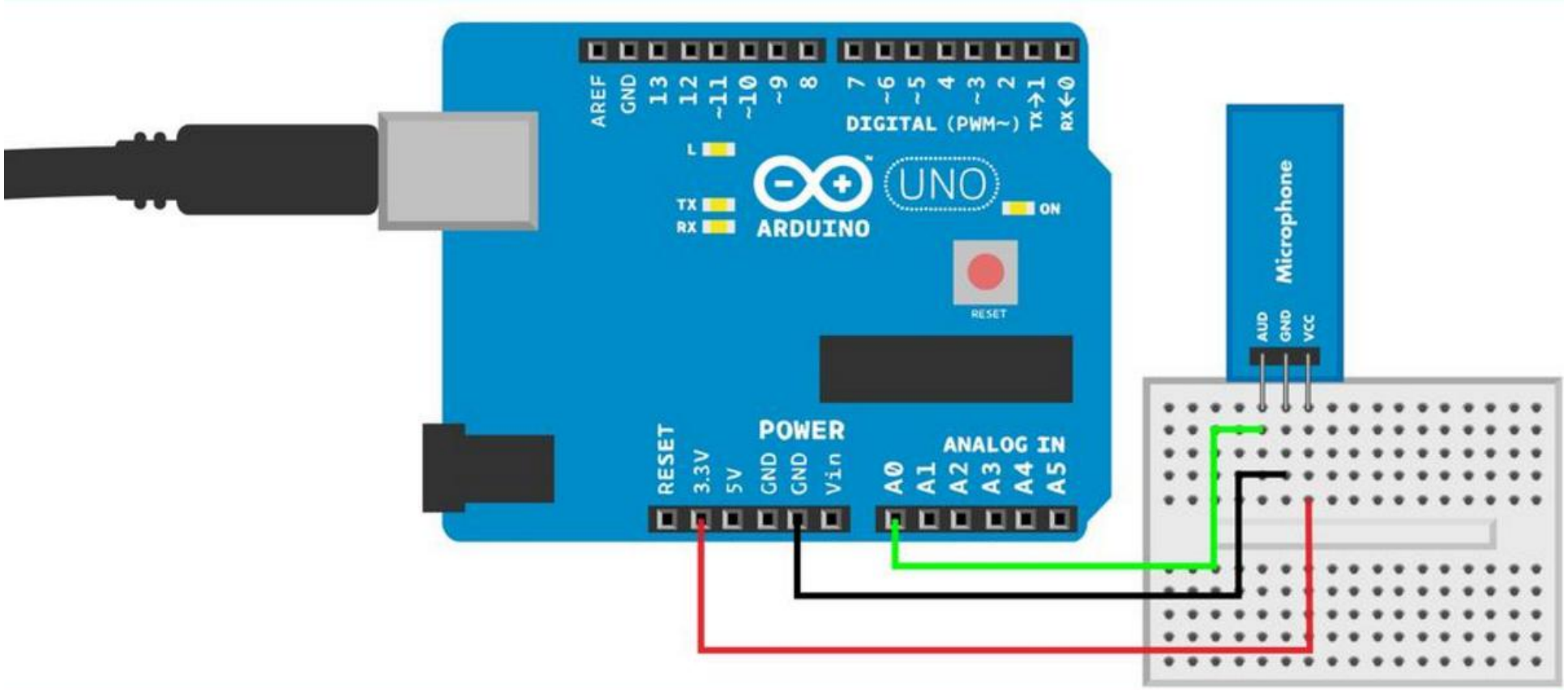


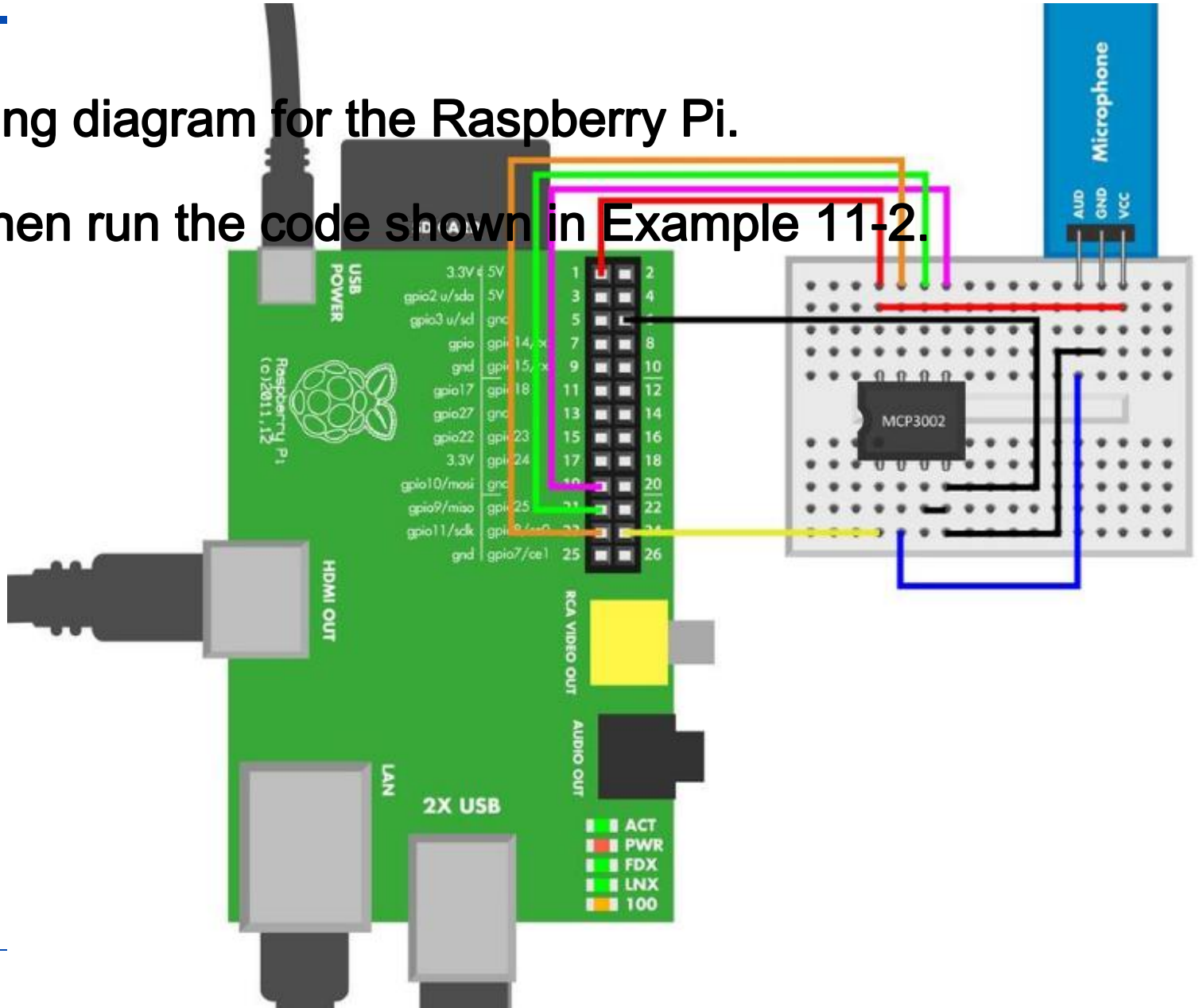
Figure 11-2. Arduino circuit for microphone

Example 11-1. microphone.ino

```
// microphone.ino - print audio volume level to serial. Print "Sound" on loud sound.
// (c) BotBook.com - Karvinen, Karvinen, Valtokari

const int audioPin = A0;
const int sensitivity = 850;

void setup() {
        Serial.begin(115200);
}

void loop() {
        int soundWave = analogRead(audioPin);          //0-1023之间的整数
        Serial.println(soundWave);
        if (soundWave>sensitivity) {                   //声音足够大时进行处理
                Serial.println("Sound!");
                delay(500);
        }
        delay(10);
}
```

Microphone Breakout Code and Connection for Raspberry Pi

- Figure 11-3 shows the wiring diagram for the Raspberry Pi.
- Hook everything up, and then run the code shown in Example 11-2.



Example 11-2. microphone.py

```
# microphone.py - read sound from analog and print it
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import time
import botbook_mcp3002 as mcp   #

def readSound(samples):
        buff = []       #
        for i in range(samples):        #
                buff.append(mcp.readAnalog())   #
        return buff

def main():
        while True:
                sound = readSound(1024) #
                print(sound)
                time.sleep(1)   # s

if __name__ == "__main__":
        main()
```

Environment Experiment: Can You Hear a Pin Drop?

- Can you hear a pin drop? Let's solve this question once and for all with a sound sensor. Connect the sensor to Arduino as you did in Microphone Breakout Code and Connection for Arduino and upload the code. Place the sensor on a solid plane such as a wooden table or a floor so that the microphone part points in the direction where you are going to drop the pin. Do all you can to minimize any background noise. Change the sensitivity value in the code so that the "sound" message is not triggered when you don't make any sound. Carefully find a value that is just on the edge of reacting to sound without responding to silence.

Environment Experiment: Can You Hear a Pin Drop?

- Can you hear a pin drop? Let's solve this question once and for all with a sound sensor. Connect the sensor to Arduino as you did in Microphone Breakout Code and Connection for Arduino and upload the code. Place the sensor on a solid plane such as a wooden table or a floor so that the microphone part points in the direction where you are going to drop the pin. Do all you can to minimize any background noise. Change the sensitivity value in the code so that the "sound" message is not triggered when you don't make any sound. Carefully find a value that is just on the edge of reacting to sound without responding to silence.

Figure 11-4. Pin dropping on floor

Environment Experiment: Can You Hear a Pin Drop?

- Open the Serial Monitor in the Arduino IDE and drop a pin on the plane. Did you get the “sound” message in the serial monitor? If yes, then you really can hear a pin drop. If you didn’t get any reaction from the sensor, make sure that the room is as quiet as possible, drop the pin closer to the sensor, and make sure that you have adjusted the sensitivity properly.
- Sensitivity to quiet sounds depends on the background noise level. Try putting some loud music on and readjust the sensitivity before dropping the pin. Can the sensor still hear it?

Test Project: Visualize Sound over HDMI

- Have you always wanted a 50” graphical equalizer? In this project, you’ll analyze sound with Raspberry Pi and show the result on your television. Sound frequencies are shown as a colorful, animated graph (Figure 11-5).

Figure 11-5. Animated graph on a big screen

Test Project: Visualize Sound over HDMI

- WHAT YOU'LL LEARN
- In the *Visualize sound on HDMI* project, you'll learn how to:
  - Analyze sound numerically.
  - Do very fast calculations in Python, using SciPy.
  - Extract frequencies with fast Fourier transform (FFT).
  - You'll also refresh your skills on pyGame and drawing Full HD graphics to HDMI output, like your television or a video projector (see Test Project: Pong).

Enabling the Serial Port in Raspberry Pi

- To use the serial port in Raspberry Pi, you must release it first. Otherwise, it's used by a login shell that you can connect to over a serial cable:

  $ sudoedit /etc/inittab

- Comment out the last line that grabs the serial port. You can comment out a line by putting a hash in front of it, which causes the line to be ignored.

  # T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100

- Reboot the Raspberry Pi.

Visualizer Code and Connection for Raspberry Pi

- Install prerequisites. On your Raspberry Pi, open the terminal and run these commands:

  $ sudo apt-get update

  $ sudo apt-get -y install python-pygame python-numpy

- Figure 11-6 shows the circuit diagram for Raspberry Pi. Hook it up, and then run the code shown in Example 11-3.

Visualizer Code and Connection for Raspberry Pi

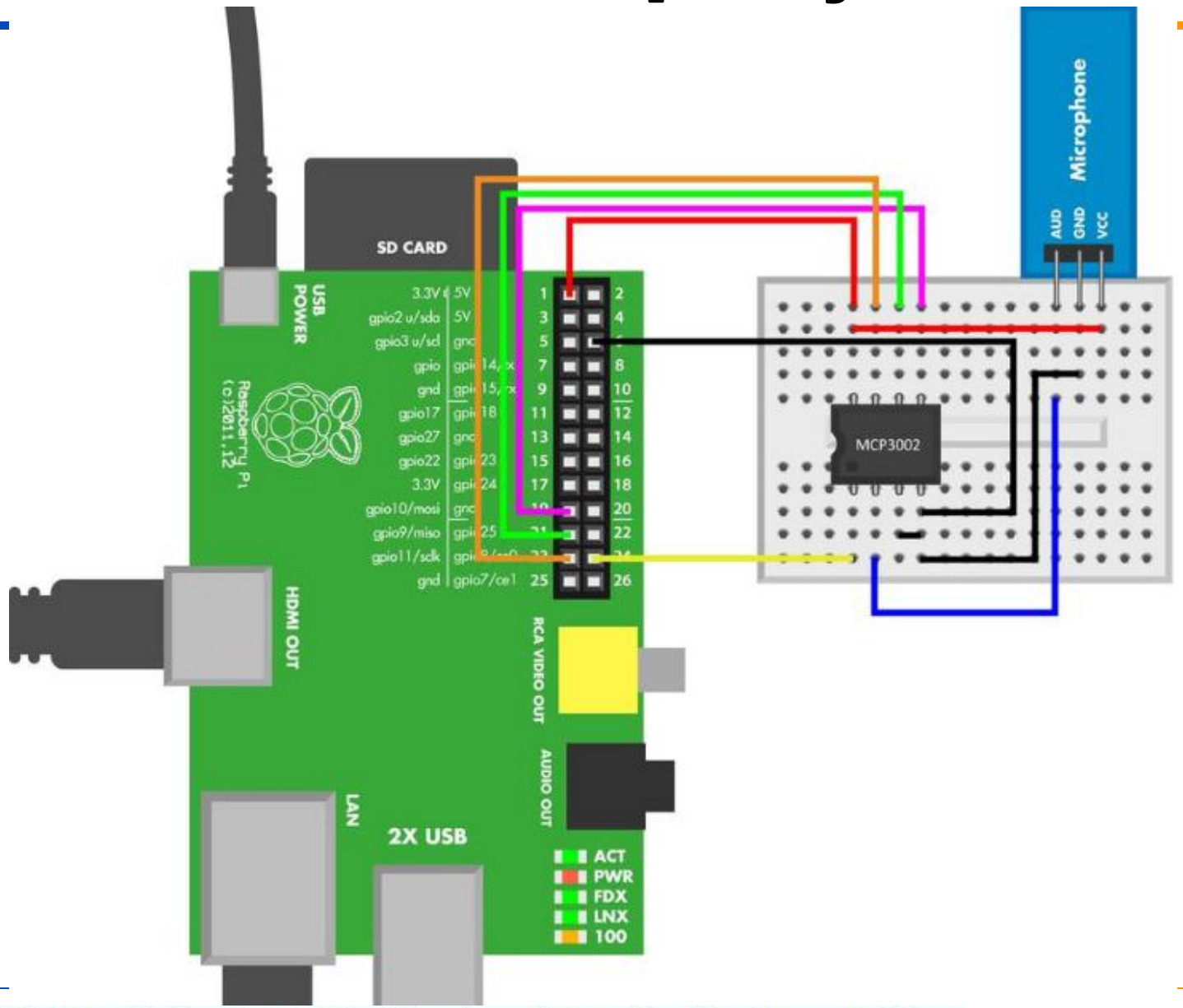


Figure 11-6. Raspberry Pi microphone circuit for equalizer

Example 11-3. equalizer.py

```
# equalizer.py - show equalizer based on microphone input
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

import pygame  # sudo apt-get -y install python-pygame
import math
import numpy  # sudo apt-get -y install python-numpy

import microphone    #
from pygame.locals import *

pygame.init()

width = 800
height = 640

size = width, height
background = 0, 0, 0
```

Example 11-3. equalizer.py

```
# equalizer.py - show equalizer based on microphone input
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

screen = pygame.display.set_mode(size, pygame.FULLSCREEN)
fullBar = pygame.image.load("equalizer-full-small4.jpg")    #
emptyBar = pygame.image.load("equalizer-empty-small4.jpg")
clock = pygame.time.Clock()
pygame.mouse.set_visible(False)
mainloop = True

barHeight = 36
barWidth = 80
barGraphHeight = 327
barPos = [55, 130]

sampleLength = 16
```

Example 11-3. equalizer.py

```
# equalizer.py - show equalizer based on microphone input
# (c) BotBook.com - Karvinen, Karvinen, Valtokari

def fftCalculations(data):    #
        data2 = numpy.array(data) / 4    #
        fourier = numpy.fft.rfft(data2)    #
        ffty = numpy.abs(fourier)    #
        ffty = ffty / 256.0        #
        return ffty

while mainloop:
        buff = microphone.readSound(sampleLength)    #
        barData = fftCalculations(buff)    #
        for event in pygame.event.get():
                if event.type == pygame.QUIT:
                        mainloop = False
                if (event.type == KEYUP) or (event.type == KEYDOWN):
                        if event.key == K_ESCAPE:
                                mainloop = False
        screen.fill(background)
```

Example 11-3. equalizer.py

```
# equalizer.py - show equalizer based on microphone input
# (c) BotBook.com - Karvinen, Karvinen, Valtokari
```

```
# Draw data to pillars
for i in range(8):    #
        bars = barData[i+1]    #
        bars = int(math.floor(bars*10))    #
        if bars > 10:
                bars = 10
        bars -= 1
        screen.blit(emptyBar, (barPos[0]+i*(barWidth+10), barPos[1]),
                                (0, 0, barWidth, barHeight*(10-bars)))
        if bars >= 0:
                barStartPos = (barPos[0] + i * (barWidth + 10),
                                barPos[1] + barGraphHeight - barHeight * bars + 6)
                barSourceBlit = (0, barGraphHeight - barHeight * bars+6,
                                barWidth, barHeight*bars)
                screen.blit(fullBar, barStartPos, barSourceBlit)    #
pygame.display.update()
```