ANSWERS

A7.8 Sound and MIDI

Data representation · A level · AQA 7517 4.5.6.7, Eduqas A500QS 2.3 · about 30 min

BugBotLab

What this lesson is about

Sample rate, sample resolution, the Nyquist theorem and aliasing, and MIDI events played on the robot's piezo.

- Sampled sound
- The Nyquist theorem
- MIDI

Questions

6 marks in all

1. How many bytes does a 30 second mono recording take at a sampling rate of 8,000 Hz and a sample resolution of 8 bits? [1 mark]

> **Answer:** 240000. 8,000 x 8 x 30 = 1,920,000 bits, which is 240,000 bytes.

2. A recording must keep frequencies up to 5,000 Hz. What is the lowest sampling rate, in Hz, the Nyquist theorem allows? [1 mark]

> **Answer:** 10000. The sampling rate must be at least twice the highest frequency: 2 x 5,000 = 10,000 Hz.

3. A 440 Hz tone is sampled at 600 Hz. What happens? [1 mark]

- ☐ **A** It is recorded as a lower, wrong frequency: aliasing
- ☐ **B** It is recorded correctly
- ☐ **C** It is recorded as silence
- ☐ **D** It is recorded at twice the frequency

> **Answer:** A. 600 Hz is below the 880 Hz the Nyquist theorem needs, so the samples fit a lower frequency wave, 160 Hz.

4. Which of these are MIDI event messages or data carried in them? [1 mark]

Tick every answer that is true.

- ☐ **A** Note on
- ☐ **B** Note off
- ☐ **C** Velocity
- ☐ **D** The sampled sound wave

> **Answer:** A, B, C. MIDI stores instructions for an instrument, not the sound itself.

5. Which is a disadvantage of MIDI compared with sampled sound? [1 mark]

- ☐ **A** It cannot store speech or singing
- ☐ **B** Its files are larger
- ☐ **C** It is hard to change the tempo
- ☐ **D** It cannot change instrument

> **Answer:** A. MIDI describes notes, so real recordings such as voices cannot be stored. Its files are small and easy to edit.

6. What does this program print? [1 mark]

```
for note in [57, 69, 81]:
    print(note, round(440 * 2 ** ((note - 69) / 12)))
```

> **Answer:**
>
> ```
> 57 220
> 69 440
> 81 880
> ```
>
> Twelve note numbers make an octave, which halves or doubles the frequency.

The task: play the MIDI events

`events` is a list of `(note, seconds)` pairs: `note` is a MIDI note number (0 to 127) and `seconds` is how long to play it. 1. Write `frequency(note)`, which returns 440 × 2^((note - 69) / 12) rounded to the nearest whole number with `round`, as an `int`. 2. For each event in order, print `note <note> = <frequency> Hz` and play it on the piezo with `tone(frequency, seconds)`. 3. After the last note, print `minimum sample rate: <n> Hz`, where `n` is twice the highest frequency you played: the Nyquist rate for recording those notes' fundamental frequencies. Calculate every frequency; do not type them in.

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

events = [(60, 0.25), (64, 0.25), (67, 0.25), (72, 0.5)]

def frequency(note):
    return 440
```

The hint students can ask for: Every 12 note numbers is an octave, and an octave doubles the frequency, so the frequency is 440 times 2 to the power of how many twelfths the note is above note 69. Play each note for its own duration. The Nyquist rate is twice the highest frequency you need to keep.

A solution

```
# the two lines every program starts with: the commands, then the robot
from bugbot import *
connect()

events = [(60, 0.25), (64, 0.25), (67, 0.25), (72, 0.5)]

def frequency(note):
    return round(440 * 2 ** ((note - 69) / 12))

highest = 0
for note, seconds in events:
    f = frequency(note)
    print("note", note, "=", f, "Hz")
    tone(f, seconds)
    if f > highest:
        highest = f
print("minimum sample rate:", 2 * highest, "Hz")
```

Any program that meets the task's checks is marked correct in the simulator; this is one way, not the only way.