ANSWERS

A12.1 Communication methods

Networks and the web · A level · OCR H446 1.3.3, AQA 7517 4.9.1.1, Eduqas A500QS 2.2 · about 25 min

What this lesson is about

Serial and parallel, synchronous and asynchronous, baud rate, bit rate, bandwidth, latency and protocols.

- Serial and parallel
- Synchronous and asynchronous
- Baud rate and bit rate
- Bandwidth and latency
- Protocols

Questions

5 marks in all

1. Why is serial transmission preferred to parallel over long distances? [1 mark]

- ☐ **A** Parallel wires suffer skew and crosstalk, which corrupt data at high speeds
- ☐ **B** Serial sends several bits at the same moment
- ☐ **C** Parallel needs start and stop bits
- ☐ **D** Serial does not need a protocol

> **Answer:** A. In parallel, bits on different wires arrive at slightly different times (skew) and interfere with each other (crosstalk); serial has one data line so avoids both.

2. In asynchronous serial transmission, what is the purpose of the start bit? [1 mark]

- ☐ **A** It changes the line from its idle state so the receiver can synchronise its clock for the character
- ☐ **B** It tells the receiver how many data bits follow
- ☐ **C** It is a parity bit for error checking
- ☐ **D** It returns the line to its idle state

> **Answer:** A. The line idles at 1; the start bit (0) makes a change the receiver detects and uses to time the bits that follow. The stop bit returns the line to idle.

3. A link uses 16 different signal levels and a baud rate of 2400. What is its bit rate in bits per second? [1 mark]

> **Answer:** 9600. 16 levels encode 4 bits per signal change, so the bit rate is 2400 x 4 = 9600 bps.

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

```
baud = 4800
bits_per_frame = 1 + 8 + 1
print(baud // bits_per_frame, 'characters per second')
```

> **Answer:**
>
> ```
> 480 characters per second
> ```
>
> Each character takes a start bit, 8 data bits and a stop bit, 10 bits in all, so 4800 bits per second carries 480 characters.

5. Which statement about bandwidth and bit rate is correct? [1 mark]

- ☐ **A** Bit rate is directly proportional to bandwidth
- ☐ **B** Bit rate is always equal to baud rate
- ☐ **C** Increasing bandwidth increases latency
- ☐ **D** Bandwidth is the delay before data arrives

> **Answer:** A. The wider the range of frequencies a channel carries, the more data it can carry per second. Latency is the time delay, a separate idea.

The task: frame a byte

The LED can be a one-wire serial line: white for 1, off for 0. Write `frame(ch)`. Its input `ch` is a single character with a code from 0 to 255. It returns a string of exactly 10 characters, each `0` or `1`: a start bit `0`, the 8 data bits of the character's code least significant bit first, then a stop bit `1`. For each character of `Hi!`, in order: 1. print the character, a space and its frame, for example `H 0000100101`; 2. put each of the 10 bits on the LED in turn, `led("white")` for 1 and `led("off")` for 0, with `wait(0.05)` after each. Then print `bits sent: <n>`, where n is the number of bits you put on the LED, counted by your loop. Last, print how long 1200 characters take at 9600 baud with a 2-level signal, using the length of one frame: `time for 1200 characters: <seconds> s`, where seconds is the result of the division printed as Python prints it.

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

BAUD = 9600

def frame(ch):
    data = format(ord(ch), "08b")
    return data

for ch in "Hi!":
    print(ch, frame(ch))
```

The hint students can ask for: Get the character's code, write it as eight bits, then reverse them so the least significant bit goes first. Put the start bit in front and the stop bit after. Count bits as you put them on the LED, and use the count of bits per frame for the timing line.

A solution

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

BAUD = 9600

def frame(ch):
    data = format(ord(ch), "08b")[::-1]
    return "0" + data + "1"

sent = 0
for ch in "Hi!":
    bits = frame(ch)
    print(ch, bits)
    for bit in bits:
        if bit == "1":
            led("white")
        else:
            led("off")
        wait(0.05)
        sent = sent + 1
led("off")
print("bits sent:", sent)
seconds = 1200 * len(frame("A")) / BAUD
print(f"time for 1200 characters: {seconds} s")
```

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