CAB202 Microprocessors & Digital Systems

Assessment 2: Microcontroller project

Task summary

Your task is to implement the game of Simon on the QUTy. The classic Simon game is shown in Figure 1.

Figure 1: Simon game[1].

This game produces a sequence of lights and tones and requires a user to reproduce this sequence. If the user succeeds, the sequence becomes progressively longer.

On the QUTy, the four pushbuttons will perform the function of the four coloured buttons in the Simon game. Each pushbutton will be mapped to the vertical segments directly above it on the 7-segment display and for each press of a pushbutton, a tone will sound. This is shown in Figure 2 and Table 1.

[1] https://en.wikipedia.org/wiki/Simon_(game)

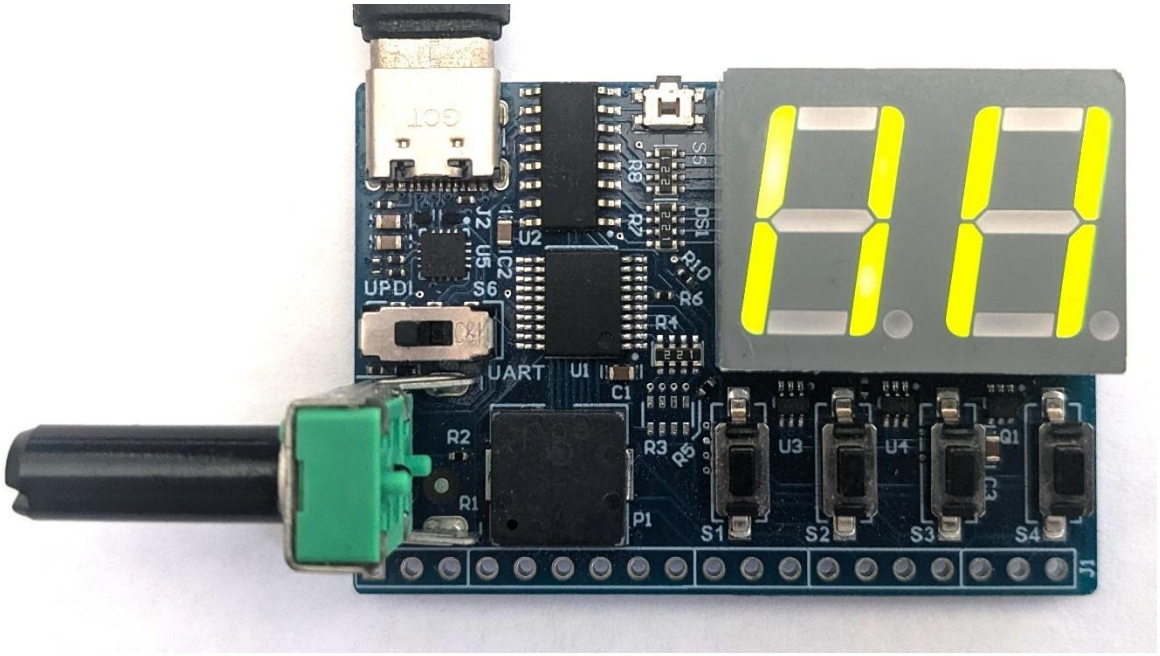
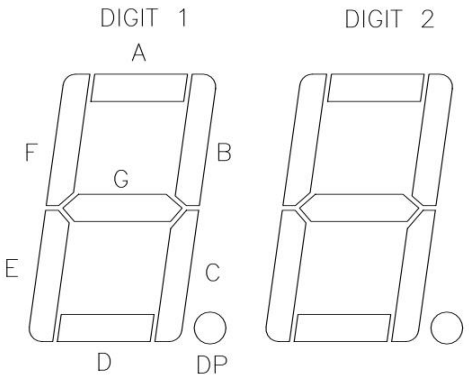


Figure 2: Pushbuttons and corresponding LED segments.



(a)

Pushbutton	Segments	Tone
S1	Digit 1: e, f	E(high)
S2	Digit 1: b, c	C♯
S3	Digit 2: e, f	A
S4	Digit 2: b, c	E(low)

(b)

Table 1:
(a) LED segment label mappings;
(b) Pushbuttons and their corresponding LED segments and tones.

The frequencies of the tones mapped to each pushbutton will depend on your student number. More information is provided in Table 2.

Functionality

Your programme is required to implement the following core functionality:

A. Gameplay

- Upon system reset, or at the start of a new game, the **sequence length** is set to **one**.
- On each turn of gameplay, "Simon" will sound a sequence of tones using the buzzer, and as each tone is sounded, the corresponding segments of the 7-segment display will be lit (see Table 1). The tone will play, and the display segments will be lit for 50% of the duration of the **playback delay** (set by the potentiometer—see section C). The tone will then stop, and all display segments will be turned OFF for the remainder of the **playback delay**. Following this, the next note in the sequence will be played.
- Once the entire sequence has played, the user must press the pushbuttons corresponding to the sequence. As each button is pressed, the corresponding segments of the 7-segment display are lit, and the corresponding tone is sounded. The segments of the 7-segment display are lit, and the corresponding tone is sounded, for either: the **playback delay** or the duration of the button press, whichever is longer.
- If the user's sequence matches Simon's, the SUCCESS pattern (all segments of the 7-segment display lit) is displayed for the **playback delay**, followed by the user's score, which is also displayed for the **playback delay**. Finally, the **sequence length** is increased by **one**. The user's score is equal to the **sequence length**.
- The next sequence will replicate the notes from the previous sequence, while appending an additional note at the end.
- The game repeats until the user fails to match Simon's sequence, or the maximum **sequence length** is reached.
- If the user fails to match Simon's sequence, the FAIL pattern (the "g" segment of each 7-segment display lit) is displayed for the **playback delay**, followed by the user's score which is also displayed for the **playback delay**. Additionally, the **sequence length** is set back to **one** and a new sequence will begin.
- The user's score is displayed as a decimal number, between 0 and 99. In the unlikely event that a score exceeds 99, the two least significant digits will be displayed. Leading zeroes will be shown only if the score exceeds 99. For example, if the user's score is 8, ' 8' will be shown. If the user's score is 108, '08' will be shown.

B. The Sequence

- The sequence displayed by Simon comes from a pseudo-random sequence seeded by the digits of your student number.
- Upon system reset, the pseudo-random sequence seed will be set back to its initial value, allowing the Simon game to be replayed.
- After the sequence is played by Simon, if the user succeeds in matching the sequence, the next sequence will contain the same steps as in the previous sequence, with one extra step added to the end.
- If the user fails to match Simon's sequence, the game will restart, and a new sequence will be generated. This sequence will commence from where the previous sequence finished.
  For example, if the sequence was 3212331 and the next digit in the sequence would have been 4, but the user enters a 2 instead of a 3 in the 5th step, the game will end and the next game will start from 4.
- The pseudo-random number generator is based on a linear-feedback shift register (LFSR) with MASK = 0xE2023CAB. The LFSR state must be initialised with your student number, encoded as a 32-bit hexadecimal literal.

  Example Student number: n12345678 → STATE_LFSR = 0x12345678
  Student number: n5556667 → STATE_LFSR = 0x05556667

- To produce the next step in the sequence, apply the following algorithm:

```
BIT ← lsbit(STATE_LFSR)
STATE_LSFR ← STATE_LFSR >> 1
if (BIT = 1)
     STATE_LFSR ← STATE_LFSR xor MASK
STEP ← STATE_LFSR and 0b11
```

- A STEP value of 00 means that the tone E(high) will be played, and the user must press pushbutton S1 to reproduce that step, a STEP value of 01 means C♯ is played, and so forth.

C. Playback Delay

- The **playback delay** will be read from the potentiometer and will range between 0.25 to 2.0 seconds. The **playback delay** is used to control several aspects of the game as discussed in Section A.
- The playback speed can be **increased** by turning the potentiometer **clockwise** (direction when facing the potentiometer).
- The playback speed can be **decreased** by turning the potentiometer **anti-clockwise** (direction when facing the potentiometer).
- The playback speed is calculated as a linear interpolation between 0.25 and 2.0 seconds, depending on the position of the potentiometer. In other words, if the potentiometer is exactly halfway between the most clockwise and most anti-clockwise positions, the playback speed will be exactly halfway between 0.25 and 2.0 seconds.

D. Playback Frequency

- The playback frequencies of the four tones will default to the values derived from your student number, as shown in Table 2.
- The octave of the playback frequencies can be **increased** by receiving an "INC FREQ" token through the UART.
- The octave of the playback frequencies can be **decreased** by receiving a "DEC FREQ" token through the UART.
- When the above tokens are received, playback frequencies will be incremented or decremented in steps of **one octave** (frequencies are multiplied or divided by 2).
- When the above tokens are received, **all four frequencies** will be incremented/decremented at once.
- The playback frequencies should remain within the range of human hearing (20Hz to 20kHz).

The four playback frequencies can be calculated using the following table:

E(high)	C♯	A	E(low)
$4\text{xx} \times 2^{\frac{-5}{12}}$	$4\text{xx} \times 2^{\frac{-8}{12}}$	4xx	$4\text{xx} \times 2^{\frac{-17}{12}}$

1. xx = last two digits of your student number.
2. Frequencies are in Hz and should be rounded to the nearest integer.

Table 2: Playback frequencies for sets of four Simon tones.

For example, if xx = 40, A = 440, and E(high) = 330 Hz, C♯ = 277 Hz, and E(low) = 165 Hz.

E. Gameplay through UART

Gameplay, INC/DEC FREQ, RESET and SEED:

- Instead of pressing the buttons on the QUTy board, the user may press certain keys in the serial monitor. The ASCII characters are sent to the QUTy over the UART, and perform the same actions as the pushbuttons.
- The UART interface will be configured to 9600—8-N-1.
- The INC FREQ and DEC FREQ keys can be used to **increment** or **decrement** the frequency of the tones as discussed in Appendix D. These keys can be pressed at any time during gameplay and are not considered part of the user entered sequence that is compared with Simon's. Changes to the frequency are retained until system reset (RESET).
- The RESET key immediately ends the current game, resets all frequencies back to the default, resets the pseudo-random sequence seed back to the initial seed, and resets the **sequence length** back to **one**.
- If the user's sequence matches Simon's, the string "SUCCESS" is displayed on the terminal, followed by a newline ("\n"). The user's score is then displayed in decimal (up to 65535), followed by a newline ("\n").
- If the user fails to match Simon's sequence, the string "GAME OVER" is displayed on the terminal, followed by a newline ("\n"). The user's score is then displayed in decimal (up to 65535), followed by a newline ("\n").
- The SEED key followed by 8 hexadecimal digits (in lowercase) will result in the LFSR being re-initialised with the 8 digits encoded as a 32-bit value. The new seed will apply to the next sequence generated (whether that is due to current game being won, or lost, or if the RESET key is sent over serial) If any of the next 8 characters is not a hexadecimal digit, the LFSR will not be updated.
- The keys that can be received through the UART, and their corresponding actions are summarised in Table 3.

Key	Function	Action taken
'1' or 'q'	S1	S1 during game play
'2' or 'w'	S2	S2 during game play
'3' or 'e'	S3	S3 during game play
'4' or 'r'	S4	S4 during game play
',' or 'k'	INC FREQ	Increase frequency of tones
'.' or 'l'	DEC FREQ	Decrease frequency of tones
'0' or 'p'	RESET	Reset frequencies to default and sequence index to 0
'9' or 'o'	SEED	Load new seed for pseudo-random sequence

Table 3: ASCII characters received through the UART and actions taken. In each case two keys are supported.

High Scores:

- A high score table of the 5 highest scores is retained, where each entry is formatted as <name, score>. The name consists of a string that can contain up to 20 characters. Scores of up to 65535 can be retained.
    - At the conclusion of a game, if the user's score is within the top 5 highest scores, the prompt "Enter name:" will be displayed on the terminal. The user may then enter their name. If user input is detected before 5 seconds have elapsed, the programme will wait for the user to conclude typing and press Enter.
    - If no user input is received after 5 seconds, the name is set to any empty string.
    - If there is no input from the user within 5 seconds after the last keypress, the programme will stop waiting, and the name will be set to the characters entered so far.

  A new entry <name, score> is then inserted on a new line in the high score table. The high score table is then displayed on the serial terminal, in order of highest score to lowest, as in the following example:

```
Alice 20
Bob 18
Charlie 14
David 9
Erin 7
```

- At the conclusion of a game, if the user's score is not within the top 5 highest scores, the current high score table is displayed.
- If there are fewer than 5 entries in the high score table, only the entries that exist will be shown.
- The high score table will be stored in SRAM and will therefore be reset when the QUTy is reset (however, the RESET serial command must not affect the high score table).

Note: Your implementation must not use bit banging (manual driving of the SPI or UART interfaces without using the associated peripheral). It also cannot use the functions provided in qutyio.h.

At the time of assignment release, some content relevant for the assignment is still to be covered in class. Appropriate implementation methods for the functionality of the assignment will be covered in class during the semester. In your tutorials, you will be writing code for many of the blocks required for the assignment.

Deliverables

Source code and microcontroller firmware (40%) **11:59pm Friday 2nd June**