Objectives: This lab test assesses all material covered in the module, but particularly, functions, arrays, pointers, strings, and advanced types.

Lab test instructions

1. This lab test is for individuals working independently and must be completed during the lab session
2. Create new sketches as requested in the lab test.
3. Before you leave the lab, you must demonstrate the final status of your programme.
4. Create a single plain text submission file (.txt) for the lab. Use a plain text editor (e.g. NotePad++) to edit the .txt file and do not use Microsoft word or similar. Copy all the sketches you write and any other answers required for the lab into the submission text file. *Name the file "**108_LabTest2_firstname_surname.txt**". Include your name and lab number at the top of the file and clearly label everything in the file. Submissions without names or unclear sections/sketches/answers will be marked down or (in the worst case) not marked at all.*

Marking for LabTest2

All practical parts of the test will be marked during the lab session. It is essential that you get the demonstrator to confirm your progress during or at the end of the lab test.

The code will be reviewed and graded based on the submission document.

For all code sections, marks will be deducted for bad readability (e.g. missing or mismatching comments, poor variable names, bad indentation, unrecognised coding style, etc.), inappropriate use of global variables, unnecessary code repetition, incorrect behaviour, or failure to follow requirements of the lab question.

General marks are also lost if the submission document instructions are not followed.

Marks will be awarded for good design decisions and using the appropriate features of C when required (e.g. functions, pointers, arrays, structures, enums, etc.) to achieve the programme goals.

1 Button click to morse code — save sketch as “LT2_ClickToMorse”

Background: the basic objective of this programme is to convert clicks on the daughterboard switches to morse symbols and to decode sequences of morse symbols to alphanumeric characters.

To do this, the basic strategy is first to detect and measure the duration of clicks. Based on their duration, each click can be classified as signalling a morse code “dot” or a “dash”. Each classified morse code symbol (dot or dash) should be stored in a buffer, so that we can examine symbol sequences. Finally, we compare the symbol sequence in the buffer with known sequences and if we find a match, then we have successfully decoded an alphanumeric character which we can display.

The advanced part of the lab test focuses on making the programme more robust to different users with different click timings and to correctly detect letter and word breaks (based on timing) so that a morse-code operator could transmit messages quickly.

In general you should try and build up the programme incrementally. First get the minimum possible functionality working (usually serial printing, click detections and click durations). Then gradually add in all requirement functionality until the programme is done.

Most people will not be able to complete all requirements, so work to complete as many as you can.

LT2_ClickToMorse sketch basic requirements:

- Download the starter sketch from moodle, extract to your sketchbook and rename as required.
    - The starter sketch contains an array/lookup table with corresponding alphanumeric characters and morse code symbols (encoded as text string).
    - It also contains a function, writeBarLeds, to switch on/off a pattern of Bar LEDs based on a single bit mask parameter. (You don’t have to use this function, but if you do it will save you time.)
    - The superloop duration in the starter sketch is initially set to 50 ms. In general, we can only measure durations to an accuracy of +/- one superloop duration, so you might decide you need to change the superloop duration. If you do, ensure you pick a superloop duration between 10-100ms in duration. If you go outside this range, click detection using `readSwitchEvent` or `readSwitchEventTimes` from the ee108 library will not work at all.
- Detect clicks on SW1 and measure the duration of each click. Use the click duration (the duration from when button is first pressed until it is released) to classify the click as a morse dot (short click) or dash (longer click).
    - *NOTE: marks are deducted if you don’t use the ee108 library functions to detect button clicks.*

- ○ All morse code symbol times are expressed in multiples of a fundamental time unit, T_{MORSE}. Choose a duration to use as T_{MORSE} in your sketch from the acceptable range 100-400 ms. (Your chosen superloop duration should allow you to measure this duration with sufficient accuracy.)
- ○ A click duration of approximately 1 x T_{MORSE} should be classified as a morse dot symbol and a press of approximately 3 x T_{MORSE} should be classified as a morse dash symbol. Durations around 2 x T_{MORSE} are indeterminate and cannot be recognised as a dot or a dash.

- Each time SW1 is clicked, print the following to serial: the time in ms at which the button was released (i.e. when the click was detected), the duration of the click (from press to release time) and the classified morse symbol. The serial output should resemble the following:

```
LabTest2_clickToMorse starting...
914 ms: Click, duration 160 ms, symbol DOT
1647 ms: Click, duration 200 ms, symbol DOT
3388 ms: Click, duration 220 ms, symbol UNRECOGNISED
4473 ms: Click, duration 500 ms, symbol DASH
```

LT2_ClickToMorse sketch <u>intermediate</u> requirements:

- Each time SW1 is clicked, indicate the classified morse symbol on the Bar LEDs as follows:
  - ○ A dot is represented by the following pattern of LEDs (where a filled circle is on): ○○○○○○○○○●
  - ○ A dash is represented by the following pattern of LEDs: ○○○○○●●●●●
  - ○ Unrecognised is represented by the following pattern of LEDs: ●○○●○○●○○●
- Each time SW1 is clicked, add a representation of the classified morse symbol to a buffer. The buffer should be able to hold a sequence of up to 10 morse symbols represented as '.' for dot and '-' for dash.
- When the buffer is full, or when the buffer has some symbols but no new symbol has been classified (i.e. no click) for more than 2 seconds, or when a click is classified as unrecognised (neither a dot nor a dash), process the buffer:
  - ○ Ensure the buffer contents are usable as a C-string (i.e. nul terminated)
  - ○ Don't do a look up if the most recently classified symbol was unrecognised. Otherwise look up the decoded character that matches the sequence of classified symbols in the buffer in the morse code lookup table.
  - ○ Print the result of decoding to serial output such that it resembles the following:

```
1000 ms: Click, duration 160 ms, symbol DOT (.)
1800 ms: Click, duration 580 ms, symbol DASH (-)
3800 ms: Morse symbol string ".-" recognised as A

5000 ms: Click, duration 160 ms, symbol DOT (.)
5500 ms: Click, duration 300 ms, symbol UNRECOGNISED (?)
5502 ms: Morse symbol string ".?" not recognised
```

LT2_ClickToMorse sketch <u>advanced</u> requirements:

- The time gaps between clicks (i.e. from the button release of one click to the button press of the next) in morse code are as follows:
  - Inter-symbol gap within a character: 1 x T_{MORSE}
  - Inter-symbol gap between characters: 3 x T_{MORSE}
  - Inter-symbol gap between words: at least 7 x T_{MORSE}
- Modify your sketch to measure and print the gap between all symbols, as follows

```
1000 ms: Click, gap before 1000 ms, duration 160 ms, symbol DOT (.)
1800 ms: Click, gap before 220 ms, duration 580 ms, symbol DASH (-)
2140 ms: Click, gap before 180 ms, duration 160 ms, symbol DOT (.)
```

- Modify your sketch so that it does not have to wait for a full buffer or 2 second delay before decoding the morse sequence, but can instead recognise character and word gaps and print these to serial as follows:

```
1000 ms: Click, gap before 1000 ms, duration 160 ms, symbol DOT (.)
1800 ms: Click, gap before 220 ms, duration 580 ms, symbol DASH (-)
2140 ms: Click, gap before 180 ms, duration 160 ms, symbol DOT (.)
2360 ms: CHAR BREAK DETECTED, Morse ".-." recognised as R

2360 ms: Click, gap before 160 ms, duration 160 ms, symbol DOT (.)
2810 ms: CHAR BREAK DETECTED, Morse "." recognised as E

3410 ms: WORD BREAK DETECTED
```

LT2_ClickToMorse sketch <u>even more advanced</u> requirements:

- Until now we have assumed that you decide the duration of T_{MORSE} in advance and that all users must fit with the number you pick.
- Now you should instead measure the duration of the clicks and gaps that the user inputs and try to identify the appropriate duration for T_{MORSE} specific to this particular user. This should allow two different users, one who makes clicks using relatively short T_{MORSE} and another who uses a relatively long T_{MORSE} to both use your system successfully without having to recompile the sketch or change any constants. Instead the sketch should adapt to the users.
- If possible the sketch should allow for sloppy users who are not able to maintain a consistent duration for T_{MORSE}. The sketch can still distinguish between dots and dashes provided that consecutive click durations are sufficiently similar (when the same symbol is intended) and sufficiently different (when different symbols are intended).

Hint: you may find it useful to have an array of structures that can remember the gap and click duration for some recent history of symbols in order to adapt T_{MORSE} and classify symbols.

✏ Copy the final sketch into your answer document.