F28HS Hardware-Software Interface

Coursework 2

Systems Programming Report

By Callum Murray (cm2103) & Tommaso Gambro (tg2016)

Problem Specification

This assignment was based on the popular board game "MasterMind", where one player sets a combination of colours in a certain order and the other player tries to guess the number and order of the colours. In our implementation of this game, the player that sets the combination is the Raspberry Pi and the guesser is you (the user). Our program is a combination of C and in-line Assembly code which can be accessed through the command line using the following code to compile the file:

```
gcc -o cw2 master-mind.c
```

And this line to run the application:

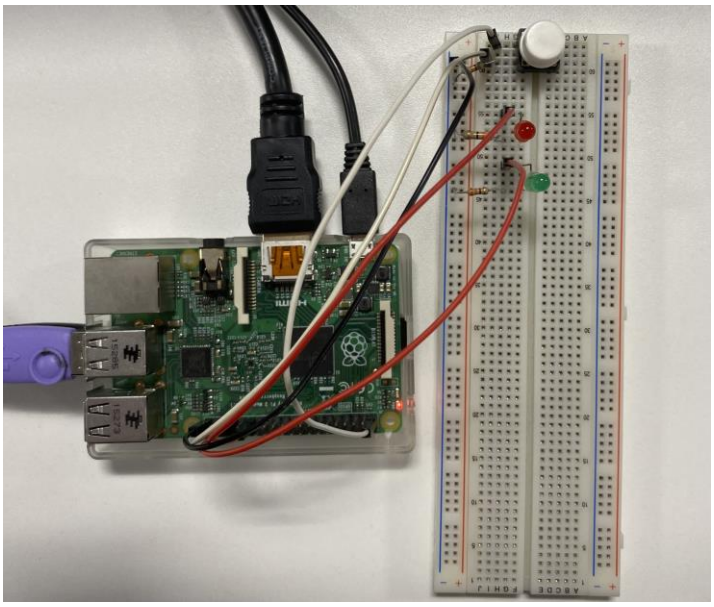
```
sudo ./cw2 [-v] [-d] [-u <seq1> <seq2>] [-s <secret sequence>]
```

Hardware Specification

We have used a Raspberry Pi 2 Model B for the processing and computation, which was wired to a breadboard containing one button, one red LED, one green LED and three 330Ω resistors each with a 5% tolerance.

The button is connected to the Pi's GPIO pin 19, the green LED to pin 13 and the red LED to pin 5 respectively.

Please see the photo below to see the exact wiring of each component.



Code Functionality Of The Main Functions

digitalWrite() - Used to send a value (1 or 0) to a pin number using the gpio address.

pinMode() - Used to set the mode of a pin to either Input or Output by calculating the fSel and shift first and then using the 'mode' variable being passed in as the mode it is set to.

waitForButton() - Used to wait for user input through the button. A timer is implemented here so that the program only waits for the user input for a set amount of time instead of indefinitely.

initSeq() - Used to dynamically allocate memory for the random sequence as well as randomly generating the colours and order of the sequence to be guessed.

countMatches() - Used to count the number of exact and approximate matches, making sure that the general matches don't include the exact ones as well. The number of matches is then stored as an array where the first element (index 0) is the exact matches, and the second element (index 1) is the approximate matches

Performance Related Design Decisions

When implementing the function pinMode() in Assembly code, the code was split into 2 sections for better readability and maintainability if anyone else wanted to modify the code. Instead of one big block of code, it was split so that one of the sections handled the shift calculation in the while loop, and the other handles the actual setting of the mode.

List Of Functions Directly Accessing The Hardware and which parts use c or asm

digitalWrite() - Accesses the hardware using C (ASM implemented but unsuccessful)

pinMode() - Accesses the hardware using C (ASM implemented but unsuccessful)

writeLED() - Accesses the hardware using C, using digitalWrite()

readButton() - Accesses the hardware using C (ASM implemented but unsuccessful)

Example Output When Running The Program

The LEDs will flash in line with the required behaviour in the specification.

Summary

In this coursework we managed to achieve a working and playable master-mind game in C which does implement the use of the breadboard and external components. However, the use of the LCD or the implementation of in-line assembly code was not managed, although it was attempted, and so the assembly code is in the comments.

This coursework has helped to further our understanding of low level assembly coding and gpio usage using a Raspberry Pi 2/3. It has also helped us to understand and use a Linux operating system (Debian), which will probably be helpful in the future.