Hardware Software Interface

Coursework 2 – Mastermind on Raspberry Pi using C and ARM Assembly

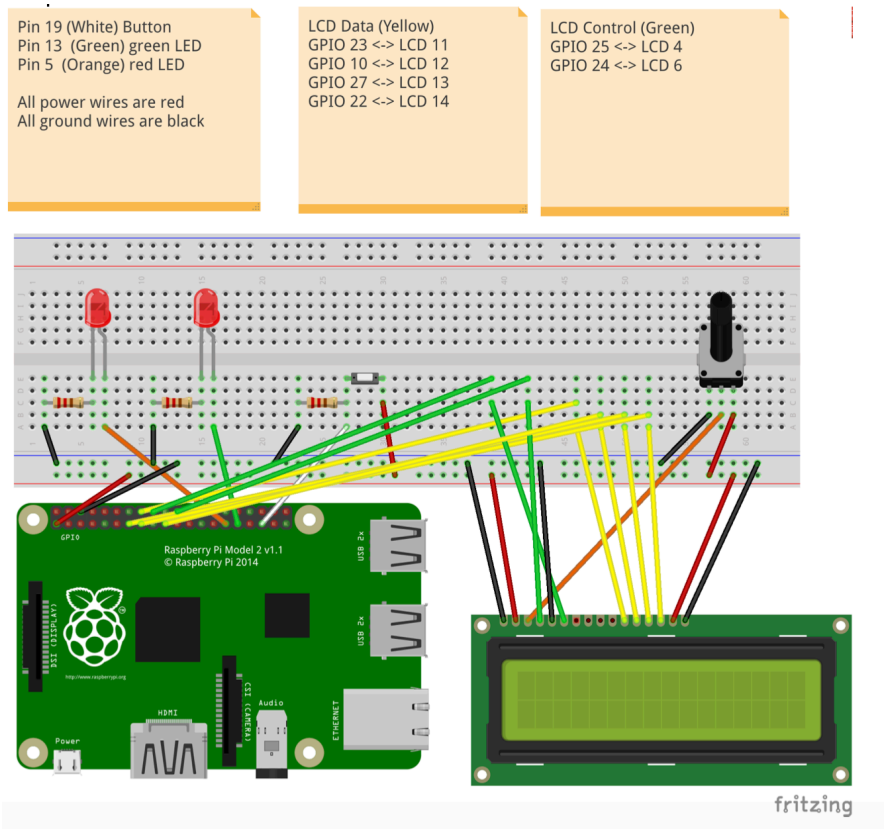
Ashwin Rajendran (H00151779) and Chi-Wai Kong (H00278458)

Specification

The aim of this coursework is to develop the Mastermind application in C and assembler, running on a Raspberry Pi2 with the following attached devices: two LEDs, a button, and an LCD (with attached potentiometer). The devices should be connected to the RPi2 via a breadboard.

The explanation of how the mastermind game works can be found here: https://en.wikipedia.org/wiki/Mastermind_%28board_game%29

The same wiring as specified in the coursework (see fritzing diagram below) was used:



What was achieved

All of the functionality that was specified in the coursework was achieved, including the functionalities 13 and 14 which were crossed out due to the LCD labs being missed due to strikes. A summary of the all the application's functionality (taken from coursework) is given below. Including additional features, we thought would be nice additions (see 15 onwards).

Functionality:

The application must provide the following functionality (see the sample sequence above):

1. The application proceeds in rounds of guesses and answers, as in the sample for the board game.
2. In each round the player enters a sequence of numbers.
3. A number is entered using the button as an input device. Each number is entered as the number of button presses, i.e. press twice for an input of two etc.
4. A fixed time-out should be used to separate the input of successive numbers. Use timers (either on C or Assembler level), as introduced in the lectures.
5. The red control LED should blink once to acknowledge input of a number.
6. Then the green data LED should repeat the input number by blinking as many times as the button has been pressed.
7. Repeat this sequence of button-input and LED-echo for each element of the input sequence.
8. Once all values have been entered and echoed, the red control LED should blink two times to indicate the end of the input.
9. As an answer to the guess sequence, the application has to calculate the numbers of exact matches (same colour and location) and approximate matches (same colour but different location).
10. To communicate the answer, the green data LED should first blink the number of exact matches. Then, as a separator, the red control LED should blink once. Finally, the green data LED should blink the number of approximate matches.
11. Finally, the red control LED should blink three times to indicate the start of a new round.
12. If the hidden sequence has been guessed successfully, the green LED should blink three times while the red LED is turned on, otherwise the application should enter the next turn.
13. When an LCD is connected, the output of exact and approximate matches should additionally be displayed as two separate numbers on an 16x2 LCD display.
14. On successful finish, a message “SUCCESS” should be printed on the LCD, followed by the number of attempts required.

Additional Features:

15. An intro message on LCD showing the game name and our student numbers
16. A message on LCD while in game showing “Make a guess”
17. An outro message on LCD, after the success (functionality 14), a message saying, “See you later” and the program terminates with exit code (0)

Explanation of Code

The code is split into two files:

- Mastermind.c - contains the main logic of the program, using functions defined in Func_Imp
- Func_Imp.c – defines majority of the functions used in the program

To compile the program use:

```
gcc Mastermind.c
```

To run the program use:

```
sudo ./a.out
```

To run the program in debug mode:

```
sudo ./a.out -d
```

The code was split this way to make it easier to understand the program with just Mastermind.c and hide the implementation details of the functions in the other file. The rest of this section will describe/explain the contents of the two files in depth:

In the following two sections I will describe the code in the order it appears in the file, so it would be more clear if you read the code as you read the explanations.

Func_Imp.c

Functions using ARM Assembly

The following functions use ARM assembly (their corresponding C versions are also included above each of these functions, but they are commented out as only the assembly versions are used):

- setMode(x,y,z) implements the mode set for the GPIO pins, (INPUT = 0 and OUTPUT = 1), it calculates the correct GPSEL Register and the required shift. This function was implemented using ARM assembly language
- writePin(x,y,z) implements the function for writing either 1 or 0 to the GPIO pin, using GPSET register for 1 output and GPSET clear register for 0 output.
- readPin(x,y) returns 0 or 1 depending in the input value from the GPIO pin

Non-ARM Assembly Functions

- mapgpio() creates/sets up the memory mapping between the gpio pins and main memory, using address 0x3f200000 as the base address
- now we define two structs LCDData and LCD, which holds data to be displayed on the LCD (from the pins) and information on how the data is displayed on the LCD.
- Lcdstrobe(x,y) this reads the pins of the lcd and loads the data into the LCDData struct
- The next two functions lcd4bits(x,y,z) and lcd8bits(x,y,z) are used to load the specified pins of the lcd with a data
-

The following functions are self-explanatory and have comments describing them in code so I will just write a quick summary about them:

- Lines 244 – 431 contains several functions which adjusts contents the LCD display., these functions are used by larger functions later on in the code to display data in the right format.
- Lines 434 – 476 contains several functions to show the results on the LED
- getButtonInput() receives the input from the button
- initialise is the function we use to set where the data is coming from for the LCD and LEDS
- initialise() sets up the necessary lcd and led pins, according to the coursework specification
- start() shows the start message
- byeMessage() shows the end message()
- UsersAttempt(x,y) checks if the users answer is correct and returns the exact and approximate values for this round
- getGuess() takes a guess from the button
- show Results(x) shows the result of the round in the LCD
- generateCode(x) generates a secret code based on the number of colours (x)

Mastermind.c

This contains the main logic of the program; this file just uses functions defined in Func_Imp. This was done so that it would be easy to change/edit the logic or flow of the game, rather than having to go through a lot of functions every time. If a function needs to be changed then they can edit the Func_Imp.c.

The pseudocode for this file is:

1. Set the colours and code length to 3 (can be edited)
2. Initialises the set-up of the led and lcd pins, connecting to the GPIO pins
3. Check if the user has entered the program using debug mode
4. Generate a secret code
5. Display the intro message on starting the game
6. Loop in the mastermind game until user guesses right answer
7. On success show the good-bye messages and end the game

Pictures of Set-up and Execution

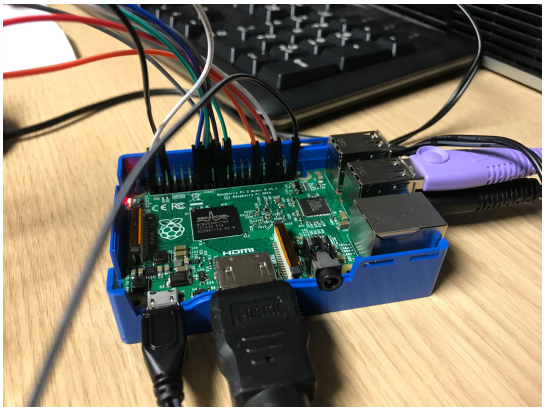


Figure 1: Raspberry Pi Wiring

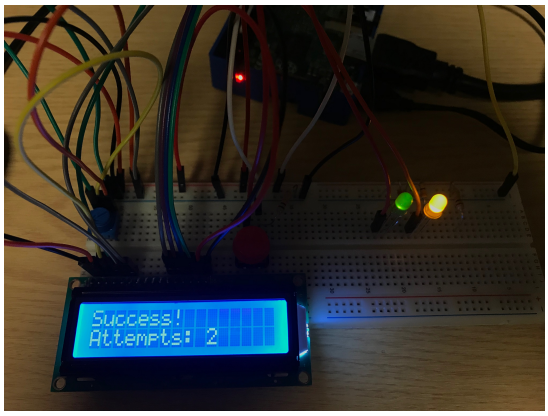


Figure 2: Success Message

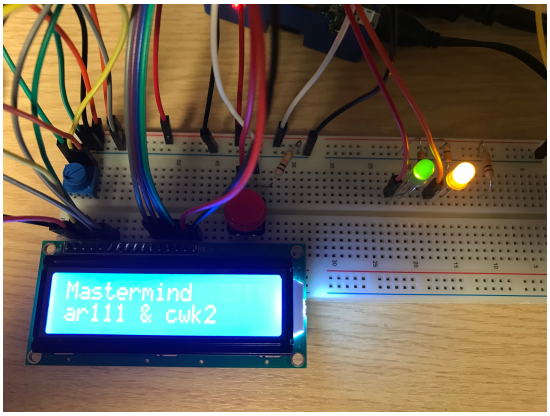


Figure 3: Intro Message

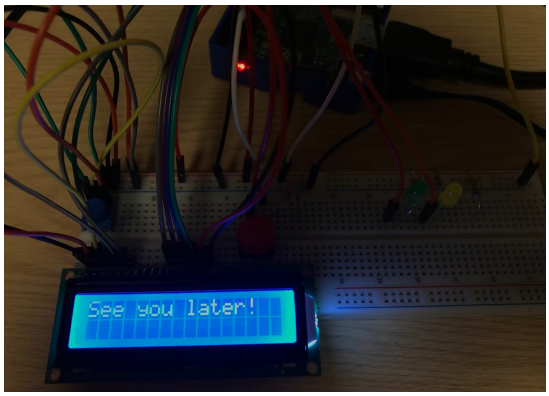


Figure 4: Goodbye Message

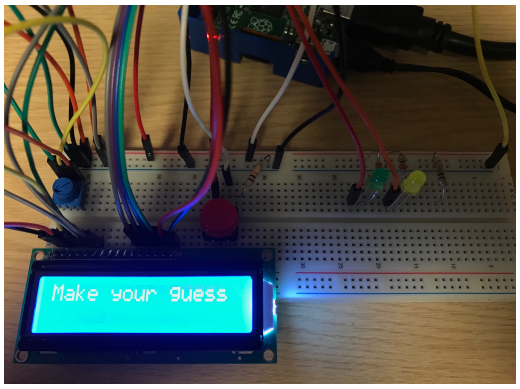


Figure 5: In game message

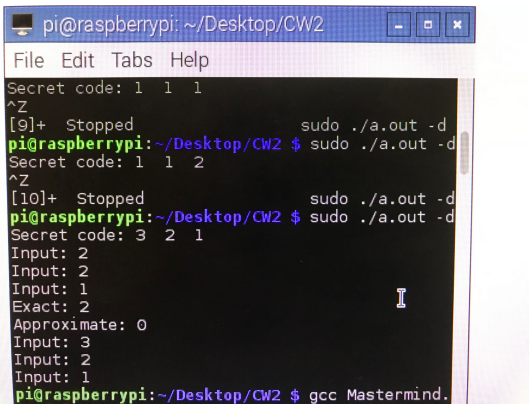


Figure 6: Debug Mode

Conclusion

Overall doing this course work taught us a lot about how hardware handles code, and how low-level languages work. Learning assembly was particularly challenging and taught us a how hardware in controlled. Being computing and electronics students, this coursework was invaluable to us as it taught us about the interface between the hardware and software.

I think we have achieved all the functionality that was asked of us, and we added in a few more additional features (message screens on the LCD) for fun. In summary, we really enjoyed doing this coursework and have learnt a lot, we will carry this knowledge and experience that we gained from this coursework throughout the rest of our education and career as this course was very relevant to our degree (Computing and Electronics).