

**Hardware Software Interface:**

Coursework 2,

*Mastermind*

F28HS

By

Jordan Walker

Wai Teng Chong

# Overview

For this assignment, we were required to implement a simple version of the MasterMind board-game using C and Assembly. The application needed to be run 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, using the RPi2 kit that was handed out early in the course.

The main aspects of the program are as follows;

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.

# Hardware Specification

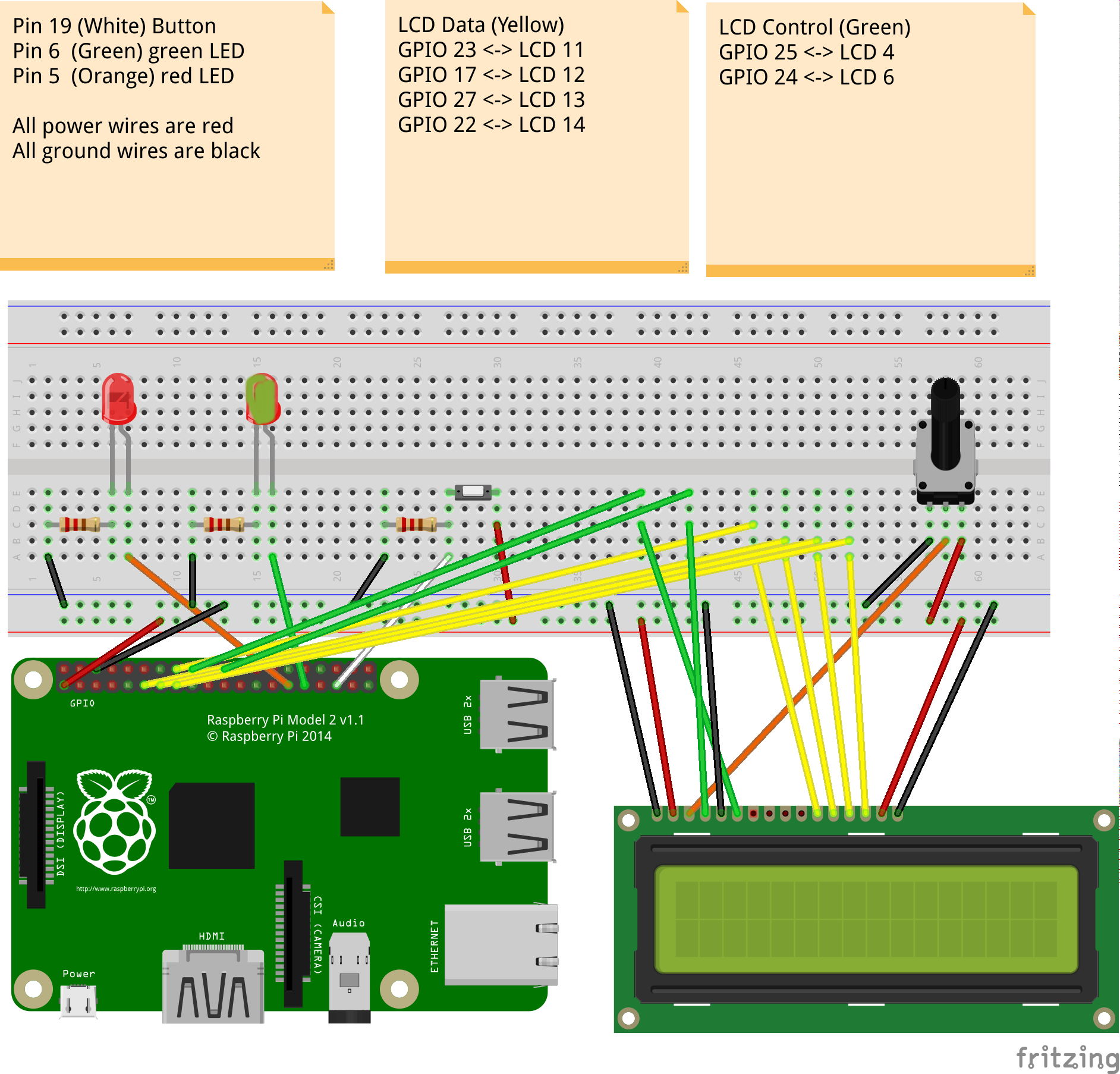
For the Hardware specification we were given very specific instructions in how to implement the program. This would consist of Two LEDs as output devices, one green for the data and one red for the control information so to separate parts of the game such as new round or indicating when it is showing black or white pegs. There will need to be a button so the user can input data values in accordance with the game and finally and LCD screen that shows the output in text form in accordance with the LEDs.

The Specification labels that;

1. The green data LED (right) should be connected to the RPi2 using GPIO pin 6.
2. The red control LED (left) should be connected to the RPi2 using GPIO pin 5.
3. A button should be used as input device. It should be connected to GPIO pin 19.
4. An LCD should be used as an additional output device. It should be connected as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| LCD | GPIO | LCD | GPIO |
| 1 | (GRND) | 9 | (unused) |
| 2 | (3v Power) | 10 | (unused) |
| 3 | (Potentiometer) | 11 (DATA4) | 23 |
| 4 (RS) | 25 | 12 (DATA5) | 17 |
| 5 (RW) | (GRND) | 13 (DATA6) | 27 |
| 6 (EN) | 24 | 14 (DATA7) | 22 |
| 7 | (unused) | 15 (LED+) | (3v Power) |
| 8 | (unused) | 16 (LED-) | (GRND) |
|  |  |  |  |
|  |  |  |  |

An overview of the entire project setup is as follows;



# Code Structure

The program structure flows downwards starting with adding the needed libaries for the program then the declarations. From then there is the setup of the LCD screen, setting of the out in assembly for inputs of 0 and 1. There are then functions to easily access the needed pieces of hardware such as ‘TurnOnLED1();’ and ‘TurnOffLED1();’ after that there is the functions that provide the flow through the game. Lastly is the main function that checks for the arguments from the user upon doing so either uses the debug or main game functions within the game.

# Functions

We only managed to put one function into Assembly which was however is called numerous times to access hardware;

void setoutput (int pinACT,int off, uint32\_t res)

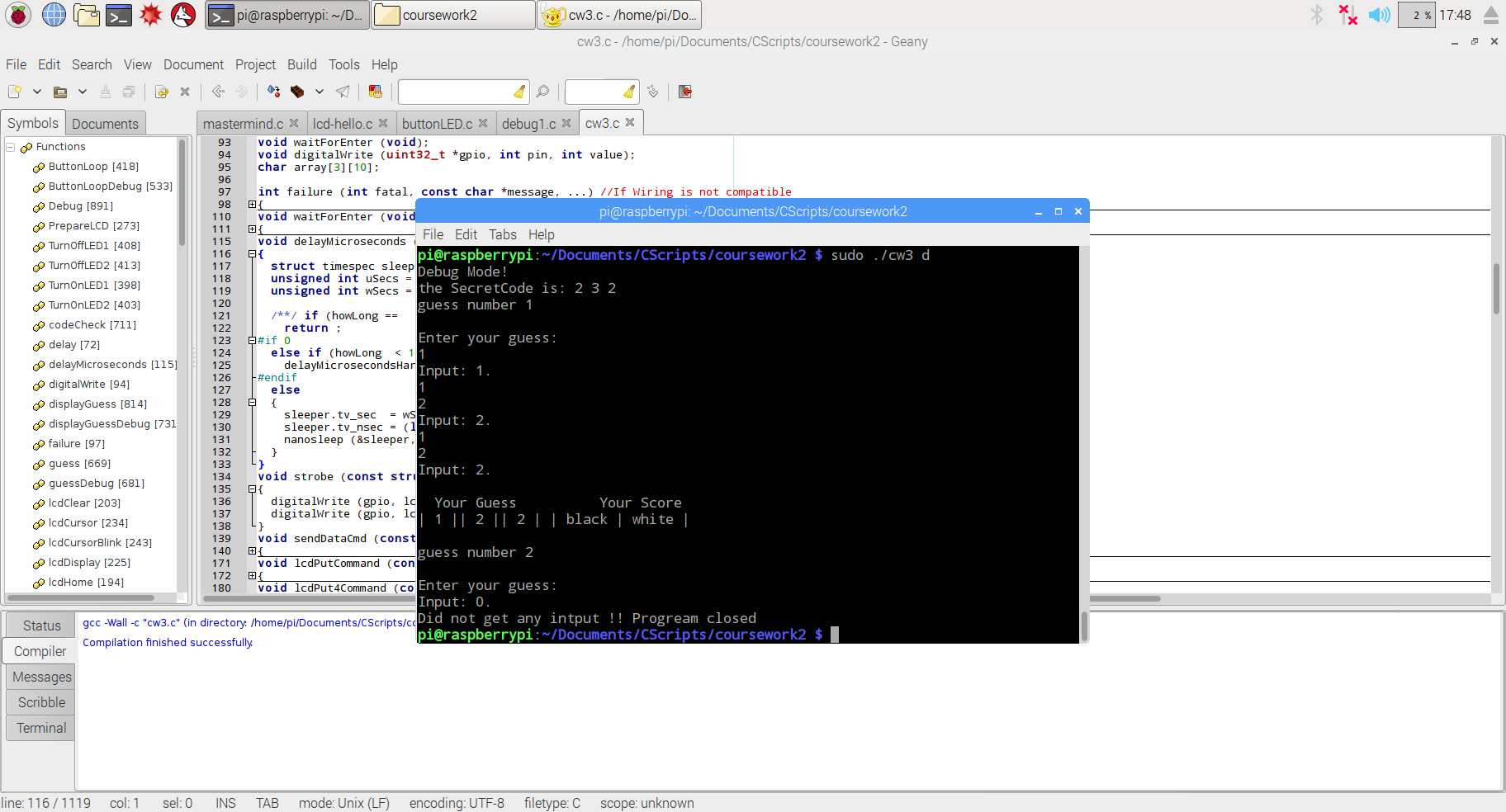
This was due to our lack of time and knowledge of assembly and is summed up in our summary. The rest of the program uses C to setup communcication between the Software and Hardware.

These functions are as followed;

* void setLED1()
* void setLED2()
* void setButton()
* void TurnOnLED1()
* void TurnOnLED2()
* void TurnOffLED1()
* void TurnOffLED2()
* PrepareLCD();

# A sample execution of debug mode

As follows are two screenshots of our program in Debug mode which can be accessed when you run the program using the sudo ./cw3 d command which allows the user to see every part of the transaction of data.



This image shows what happens when the user has finished a sequence but has got the wrong sequence with the one correct value in the right place and another correct value in the wrong place. It also allows the user to carry on too inputting the next sequence.

# C:\Users\jord\AppData\Local\Microsoft\Windows\INetCache\Content.Word\2016-12-04-174914_1824x984_scrot.png

This screenshot shows what happens when the user guesses the secret code and ultimately wins the game.

# Summary

Between us we have discussed that this piece of coursework was one of the most stressful yet fun and useful coursework’s we have completed during our time at Heriot Watt. During this coursework we have learnt how to create programs that can access the hardware and display a result to the user whom can interact with the program without the need of using the keyboard.

We have both learnt a great deal during this coursework and feel accomplishment from what we have created and by doing so has increased our knowledge of both the C and Assembly languages. We have both agreed the best feature of our program is the LCD display to give the user information. We followed on from the specification given and implemented the ability for the user to see the input they are currently on and then the selection that they have made. Although not a challenging feat now we have accomplished the task converting the integer to a char to then create a string to display on the LCD was a huge accomplishment for us. It also provides useful commands such as showing when to input the next value and so on.

There was two main struggles within our project however which included getting the input value from the button and also converting hardware features into assembly. This was a huge hindrance and didn’t allow us to fully meet the potential we think we could achieve.

The issue with the button input was our main problem however. When beginning the coursework we were quickly able to use the button however we were faced with a challenge that it sometimes it read the input twice, three - ten times until we realised it was cycling over the ‘for’ loop too quickly and realised we had to implement a way of only generating one input from a press. Once we figured the solution the problem seemed trivial however it was a useful learning process and resulted in us always getting an accurate input from the user.

We would however both of liked to create more assembly code in the program and believe that is our main set back in terms of meeting the requirements outlined. We however managed to create some inline assembly for setting the value of input i.e. whether it is 0 or 1 to send signals to the button and LED. We attempted numerous times to enable more to no avail however it is correct to say we still both learnt a lot from the basic implementation.