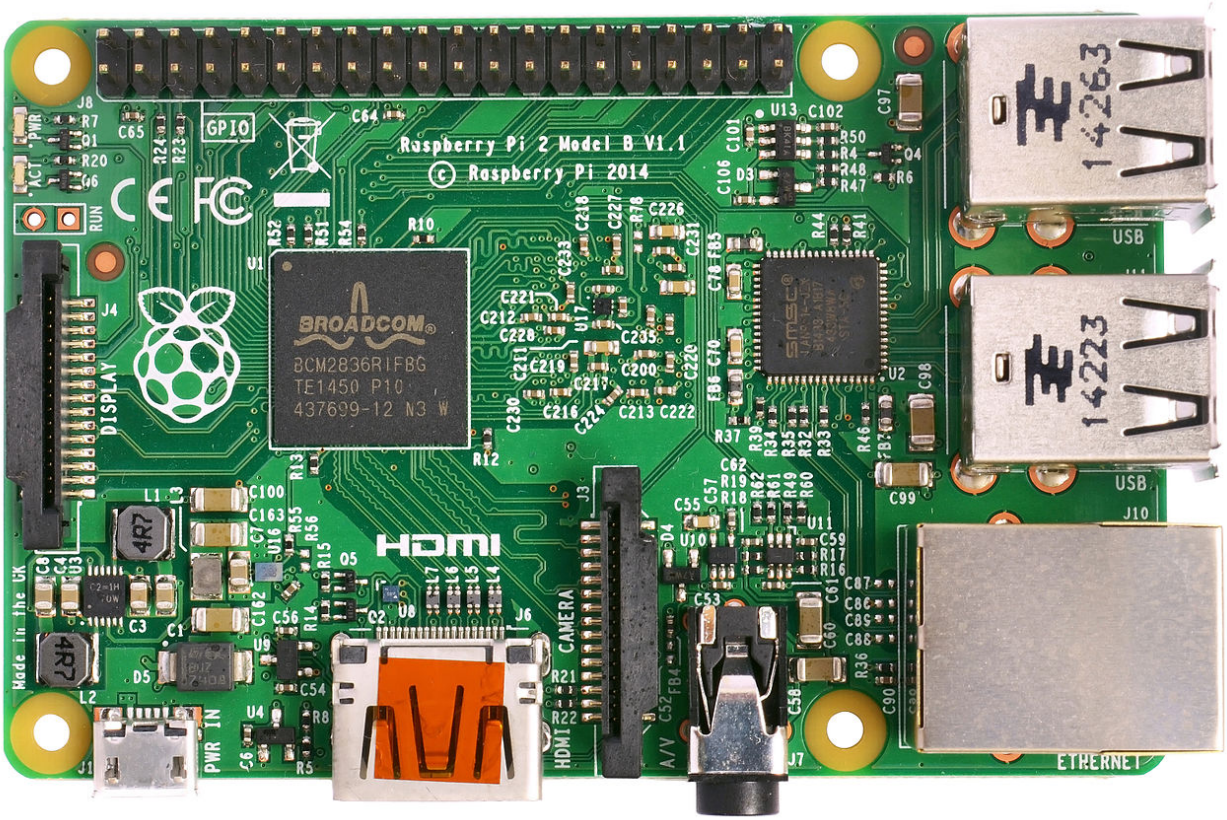
# Hardware-Software Interface

## **Coursework 2:** Systems Programming

27-03-2017

Giacomo Benso (GB11)



## Problem Specification:

We have been asked to develop a simple instance of the mastermind game on a raspberry PI 2 connected with some LEDs, a BUTTON and an LCD.
The programming languages being used are C and some ARM assembler embedded in the C code, two languages close to the hardware for a better understanding of how everything works at this level.

## Hardware Specification and Platform:

The hardware used for this coursework is the Raspberry PI 2, a breadboard, an LCD display, 2 LEDs and a button, everything connected with appropriate wires and few resistors.
Every connection with any external input/output device is created connecting the device to a RPI2 pin which is then mapped into memory to allow its use writing in the correct position in memory.
For this coursework every IO device is plugged into the breadboard and the wire connected with the corresponding pin is also inserted in the correct column of the breadboard to send the data to the device. All the circuits end with a resistor and another wire connected to the ground.
The 2 LEDs are connected to pin number 5 and 6 and are configured as output while the button is connected to pin number 19 and configured as input.
The LCD is a bit more complicated because it requires more wiring as more data has to be sent to it, precisely we connect it to 4 different gpio pins to send the bits in: 17, 22, 23 and 27.

## Code structure:

The code structure follows the usual C structure with some include and define statements in the beginning followed by the functions needed in the main function, the last part is the main method.
A lot of the functions declared are needed to allow the display to work, other functions needed are for example the 2 delay functions to insert some delay in the program flow.
I also created some function for the functionality of the game like the one finding the number of exact and partial matches or the one checking if a value exists in an array.

The program also includes the three functions to communicate with the hardware but I am going to discuss those in the next paragraph.
The main function starts with the declaration of the variable needed in the execution of the program, we then have the memory mapping to transfer the gpio pins in main memory allowing us to send bits writing in the correct location in memory.
After mapping the pins in memory we then set the modes for the IO devices connected to those pins.
We then set the display mode and make sure is ready to receive characters via the lcdPuts() function.
At this point we start the game execution which is going to be discussed in the "program execution" paragraph.

## Functions accessing hardware:

In my program I have three functions to communicate directly to the hardware:

- void setPinMode_ASM(int piNNum, int inOut)
- void sendBitSetClear_ASM(int pinN, int value)
- int getButtonValue_ASM(int butt)

**setPinMode_ASM**: This function is used to set the mode of the IO connected to the pin passed as a parameter to input (0) or output (1).
The function takes the pin number and calculates the GPFSEL register and the position in it where we need to write our bit (this part is done in C).
Then if the "inOut" parameter is set to 1 (output) we execute the inline ARM code to write a 1 in the correct position otherwise we execute the other ARM inline which set that bit to 0 (output).
I use this function to set the mode of the button, the 2 LEDs and the 4 pins connected to the display.

**sendBitSetClear_ASM**: This function is used to send the value of 1 in the correct location in the set or clear registers.
The first part of the function selects the correct register (GPSET0, GPSET1, GPCLR0 or GPCLR1) depending on the pin number passed as a parameter (this part is done in C).
However, the main part of the function is the inline ARM that sends the bit (1) in the correct location in the chosen register.

**getButtonValue_ASM**: This function is used to retrieve the value from the GPLEV0 register, the pin number for the button is passed as a parameter.
The function is almost all of it done in inline ARM apart from the declaration of the value to be return and the return statement.
In the ARM part we retrieve the value of the GPLEV0 register in the position corresponding to the button pin and we return that value (1 if the button is pressed and 0 is it is not).

## Program Execution:

The execution of the program in debug mode (adding the argument "-d" when running it) follows this sequence:
1. The welcome message is printed on the LCD.
2. the secret code is generated and displayed on the terminal.
3. we start the first round where the user has to insert the code using the button, for every input the value is shown on the screen and the LEDs blink once for every time the button was pressed.
4. For every code (3 inputs) inserted the full and partial correct guesses are displayed in both the LCD and the terminal.
5. the game finishes either if the secret is guessed (the LCD displays the number of attempts needed) or if the player has reached the maximum number of attempts defined (set to 10).

## Summary:

Working on this coursework I have learnt a lot about programming at low level especially the assembly part (which was the most challenging in my opinion) taught me a lot about this type of approach this close to the hardware.
I think and I hope I have achieved all the features I have been asked to reach for this piece of coursework including the functionality of the game which seems to follow the specification as well as the part covering inline assembler.
For the assembler part in the beginning I was creating different functions for every task but then I realized I just had 3 types of functions (set the mode for the pin, send a 1 to either the clear or the set register and reading a value from the GPLEV register) so I created this three functions taking some parameters to perform the action on the correct pin and location.
Overall I enjoyed working on this coursework and I think it has improved some of my programming skills.