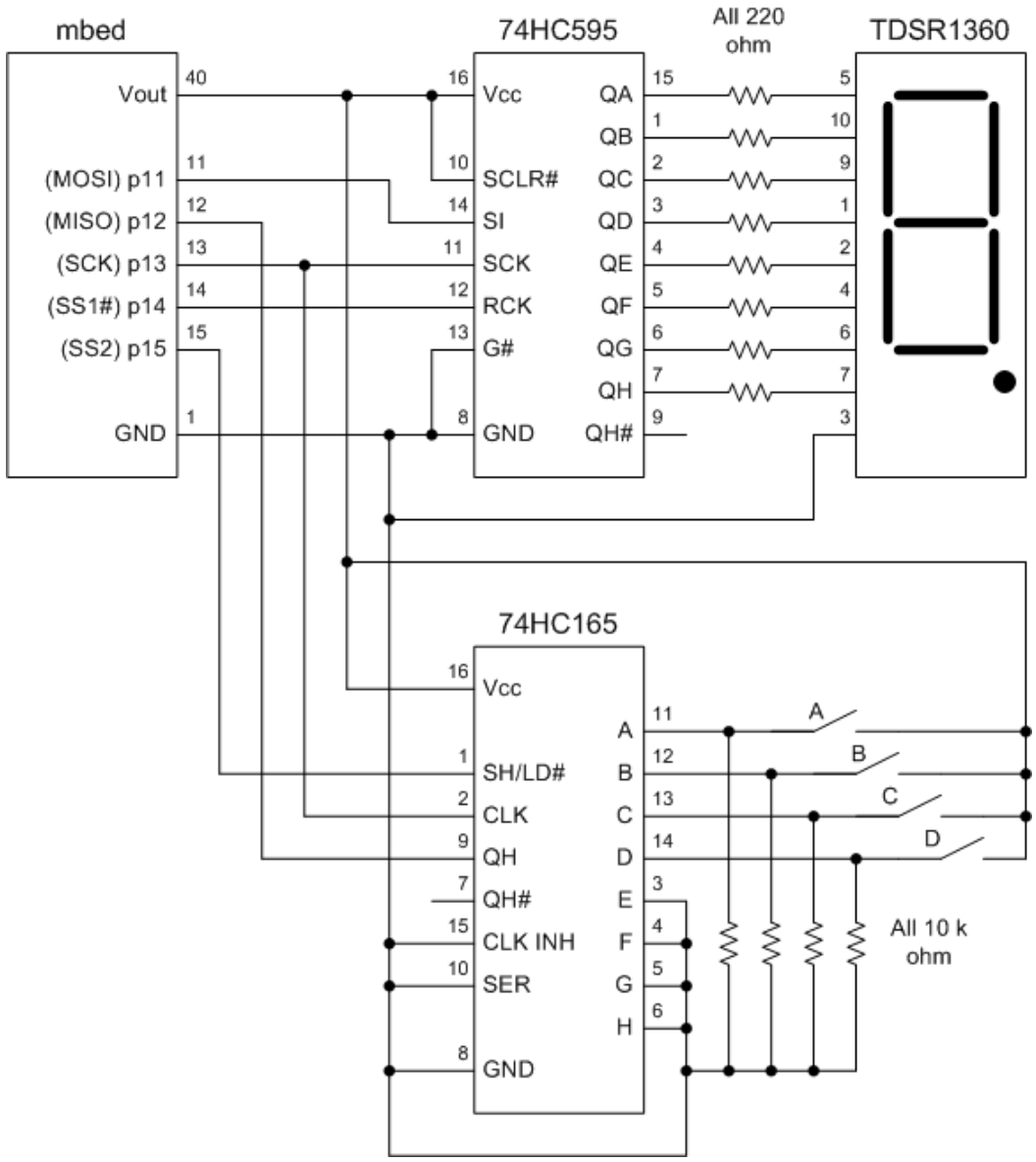
Assignment #4 – SPI reaction time game

In this assignment, you will use the DigitalOut, SPI, and Timer objects on the mbed microcontroller board to implement a simple game that challenges your reaction time.

Build the circuit shown below:



In this circuit, the mbed operates as an SPI controller and the 74HC595 and 74HC165 chips operate as SPI peripheral devices. The 74HC595 uses SPI mode 0 and an active low chip select on p14. The 74HC165 uses SPI mode 2 and an active high chip select on p15.

The 74HC595 expects an 8-bit message from the SPI controller and uses this message to control the 7-segment display. Bit 7 of the message controls segment A of the display, bit 6 of the message controls segment B of the display, and so on. Bit 0 of the message controls the decimal point on

the display. The 74HC595 does not send any data to the SPI controller. If more than 8 bits are transferred, only the last 8 bits will be used. The display is only updated at the end of the message.

The 74HC165 sends an 8-bit message to the SPI controller, reporting the state of the 4 switches. Bit 0 reports the state of switch A, bit 1 reports the state of switch B, and so on. If more than 8 bits are transferred, all of the remaining bits will be 0. The 74HC165 ignores any data sent by the SPI controller.

The game itself consists of an indefinite number of rounds, with each round slightly more difficult than the last, with the challenge to the human player to last as many rounds as possible before eventually making a mistake. For each round, the mbed should randomly select a letter A, B, C, or D and display this letter on the 7-segment display. After the letter is displayed, the human has a limited amount of time to press the switch corresponding to this letter. If an incorrect switch was pressed, or the no switch was pressed withing the time limit, the game is over and the 7-segment display should show “L” for lose. If the correct switch was pressed, a new round should begin with a shorter time limit.

When the game starts, the time limit is 1 second. For each subsequent round, the time limit should be 80% of the previous time limit.

The C function rand() will return a pseudo-random integer. It actually returns the next number in a very long sequence, so it’s actually only random in that it’s hard for a human to predict. However, the mbed always starts in a well-defined state, so you will always see the same sequence every time you start your program if you don’t do anything else. If you want to increase the randomness, you can start a timer at the beginning of the game and use the elapsed time at the start of the current round to “seed” the random number generator using the srand() function like this: srand(timer.read_us()). Note that this just moves the random number generator to a new point in the sequence; you still use the rand() function to get the random numbers.

Submit your “main.cpp” to the appropriate dropbox on Canvas by the end of April 8th.