**Build Of Materials**

1 Arduino Uno

1 USB A cable

2 Breadboards

1 Yellow LED

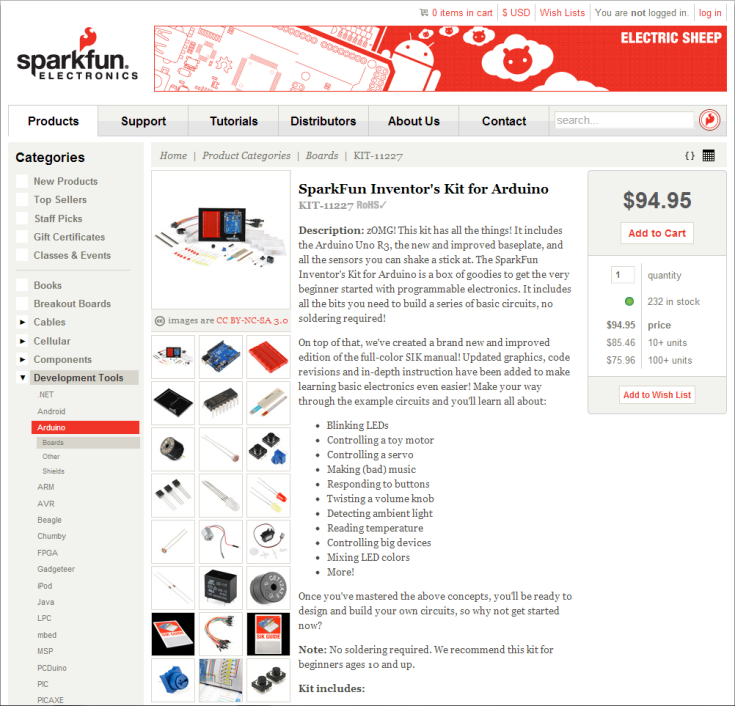
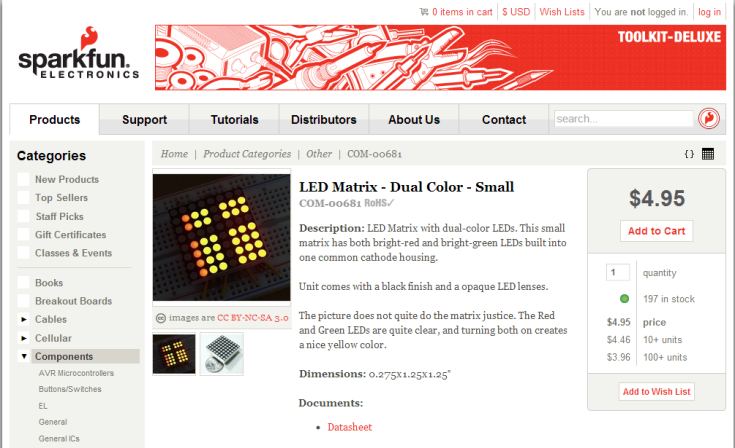
1 Red LED

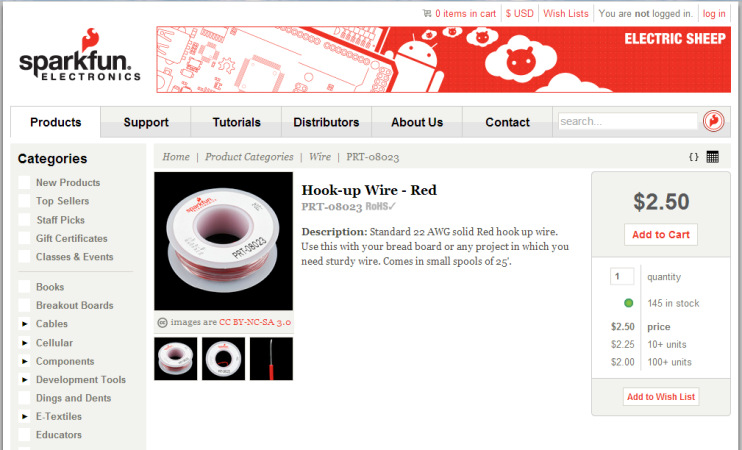
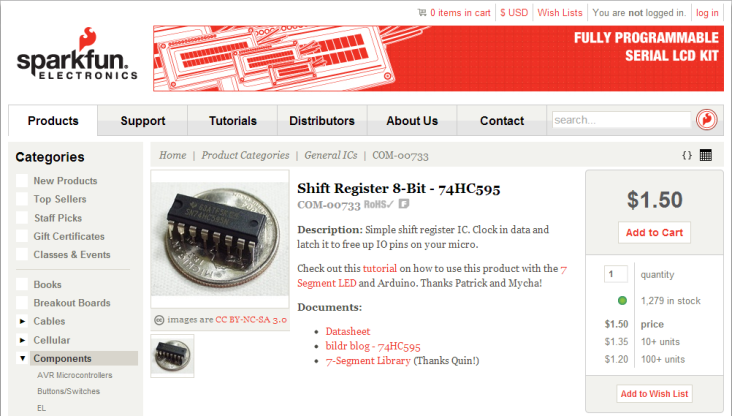
16 330 ohm resistors

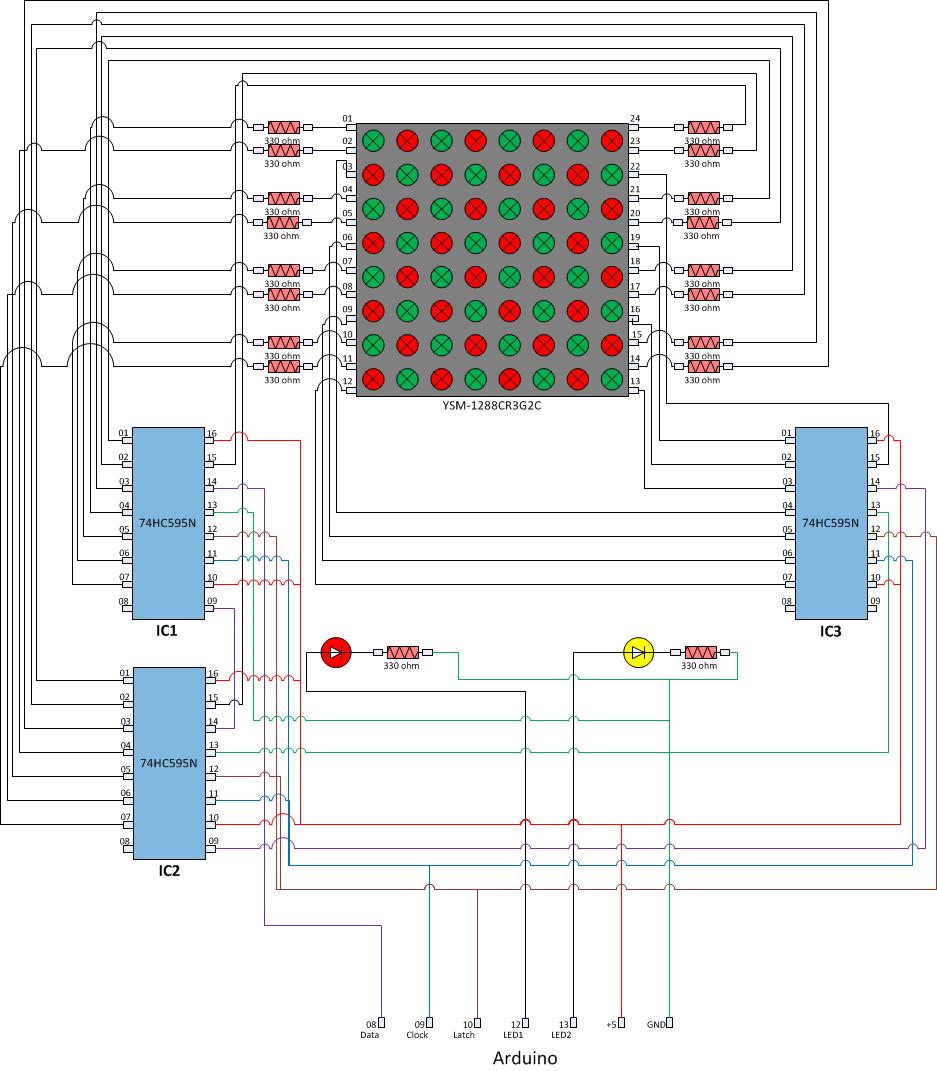
3 75HC595 Shift Registers (only one is included in the SparkFun kit, see data sheet)

1 Dual Color 8x8 LED Matrix (not included in the SparkFun kit, see data sheed)

5 Feet 22-24 gage wire (not included in the SparkFun kit)



Schematic

****

The code is designed to run on both the PC and the Arduino board by making a slight modification to it. If the “#define arduino” line is commented out in the C code, and then that code is copied and pasted into the Arduino IDE it becomes a valid, self-contained sketch (.ino file). Thus, the PC version and the Arduino version share the same logic and differ only in the output display mechanism (ASCII characters sent to the console on a PC versus pin signals to the LED matrix circuit on the Arduino) and related setup.

A partial solution is given which uses serial communication to run the checkers engine on the PC while displaying the game progress on the Arduino board LED matrix circuit. This solution is a step toward the final solution, given only for your reference. You may choose to use the serial communication code provided to create a similar initial implementation, but the final solution is able to run independently on the Arduino board with no communication

**Competition**

It is recommended that you instruct the students to isolate their checkers engine code, as well as their display code, by following the baseline code example. The Project03\_Solution\_Serial\_PC demonstrates how these files might be organized. Then, both teams’ checkers engines are compiled with Main.c and run as a C program on the PC. The method evaluateGames() will display the win/loss record at the end of a series of games.

**Stage 1 – Speed**

Run each team’s solution on the PC.

Running the solution on the PC, the user can run a series of hundreds or thousands of games in order to get an average execution time for a game by calling Main.evaluateGames(). Have one team’s checkers engine play itself (have it called from both takeTurnPlayer1() and takeTurnPlayer2). The result will reflect the speed performance on this engine only. This is the mechanism that can be used to determine the execution speed of a checkers engine.

Note that:

* You can change the number of games played by changing the value of numGames in Main.evaluateGames(). This value bounds the loop which plays games repeatedly and then summarizes the results.
* There is a global variable named “verbose” which determines whether a game displays each move as it is made, or whether only a summary of the game result is displayed. The increased I/O of verbose execution increases the execution time. Therefore, it is recommended that all rounds of the competition are run in non-verbose mode (verbose=0) to make execution more efficient and make the results focus more on checkers engine performance.

**Stage 2 – Memory Use**

Compile each team’s solution in the Arduino IDE.

When you compile the Arduino sketch in the Arduino , the size of the binary executable which will be uploaded is displayed in the bottom console in a message like:

Binary sketch size: 8,622 bytes (of 32,256 byte maximum)

Take one team’s sketch and compile it in the Arduino IDE. This is the mechanism that can be used to determine the executable size of a checkers engine (and the rest of the code).

Note that:

* If one implementation has a great deal of display code, the overall size of the executable will be larger, even if the checkers engine portion of the code is more compact.

**Stage 3 – Display**

Upload each team’s solution on to their Arduino board circuit and run.

The final game should be self-contained and run independently on the Arduino board. Take the compiled sketch and upload it to the Arduino board. Then disconnect the USB connector and connect the power adapter to power the board. The game should play itself repeatedly. The evaluation of the display is subjective, but criteria will include a recognizable game board, the ability to distringuish one player’s checkers from the others, and the ability to distinguish crowned checkers (kings) from un-crowned checkers. This may be judged by the instructor or by a popular vote of the class.

**Stage 4 – Win/Loss Record**

Merge code from two teams and run a series of games on the PC.

The win/loss record of the competition will be determined by running a series of hundreds or thousands of games pitting one checkers engine against another. This is done in Main.takeTurnPlayer1() and Main.takeTurnPlayer2() by replacing the call to takeAutoTurn() with the call to each team’s checkers engine takeTurn() method. Then run as in Stage 1 to determine the win/loss record of each team.

Note that:

* To be fair, when two checkers engines are pitted against each other after the series of games is complete, another series should be played with the playing order reversed. To do this, move one team’s takeTurn() method from takeTurnPlayer1() to takeTurnPlayer2() and the other’s from takeTurnPlayer2() to takeTurnPlayer1(). In this way, each team’s engine will have the opportunity to play first.
* By changing key method calls like takeTurn() and display(), teams can create a game with the two best checkers engines and the best display, even if all three elements come from different teams.
* Even an engine playing itself will not have exactly 50/50 in win/loss. Out of 1000 games, you might see 275 wins, 325 losses and 400 games that end in a draw. Repeating the evaluation might result in 350 wins, 325 losses and 275 draws. Results will vary from run to run. However, the Statistical results of 50,000 will be more precise and result in less variation than the results of 50 games.
* There is a global variable named “maxMoves” which determines the maximum number of moves in a game before it is declared a tie. As maxMoves increases, there are fewer draws but execution times increase because more moves are potentially taken in a game. You may find that some checkers engines do better in longer game and others do worse. Again, this value may be adjusted as desired, but should be consistent between all rounds of the competition.