When I originally built the unit, it sort of worked, but had issues – periodic lockouts and strange behavior. I could never finish a game. I got frustrated with the unit and almost threw it away, but after letting it sit for awhile got the resolve to try and figure out the problems (especially after spending all the time and $$$ building it!). The concept of the gameplay is great – just needed to tweak the electronics and programming.

I narrowed the main issues to 1) insufficient power supply, 2) insufficient memory in the Arduino Nano MPU, and 3) program code issues. I think 3) might be related to 2) – not too much can be done/added to the code without running out of memory in the Nano.

1) Power supply: You will NOT be able to run the game with 5V off a computer USB port. The LEDs at full brightness (32 white squares, plus the control panel) draws over 2A. This, plus the RPi, was overwhelming my 5V, 2A power supply, and at times the RPi would not even boot up. Also, the current draw is likely exceeding the design rating of the micro-USB connector. What I ended up doing was using a 12V, 36W supply I had lying around, and using two 12V-5V voltage regulators in the chessboard – one reg for the LED strips, and the other reg for the RPi/MPU. I also reduced the LED brightness levels (downsized the values of controlPanelLED.setBrightness and chessboardLEDS.setBrightness in the Chess\_Simp.ino setup area from their max 255 levels). Using 2 voltage regulators might be overkill – but once I did this, the power supply problems went away. You might be able to run the board with a USB-C type connector off an RPi4 type 5V, 3A power supply – I may try this in the future.

2) Arduino Nano memory: With the power supply issue worked out, the game would still lock up at times, and I could only sometimes finish a game. I thought this might have something to do with bugs in the code, so I started experimenting with revising the RPi python code and Nano sketches. In the process of doing this, I discovered that I could not add too much to the Arduino code without running out of memory space. I think the 2K of RAM is just not enough for the sketch + variables. I am not a very efficient programmer, so what I did was change the MPU to a ATSAMD21-based board (an Adafruit ItsyBitsy M0 express). This has 32 KB RAM – more than enough. I had to change some of the pin assignments, which was pretty straightforward. Additional advantages are the M0 board uses 3.3V logic (so no need for a logic-level converter board), and the TX/RX serial port is separate from the USB port, so programming/comm does not interfere with the RPi communication (unlike the Nano where TX/RX and the USB is shared). See the revised wiring diagram.

3) Code issues: Again, this might have had something to do with 2). Once I had more memory to play with, I could go ahead and resolve some of the issues I saw. These included hint handling, player pawn promotion, and checkmate detection. Also, I was able to add a “check board layout” function to allow the player to identify the pieces on the board if something happens to physically disrupt the piece positions. The code revisions I did were to the StartChessGame.py (for the Raspberry Pi), and Chess\_Simp\_forM0.ino (for the M0). I changed the name of the ino file to avoid confusion with the original Nano sketch.