ONLINE CHESS BOARD

Benjamin James 2015

Contents

Introduction 2
The specification 2
The design 2
The two problems 2
Moving the opponent's piece 2
Detecting my move 2
Solution 3
Multiplexing 3
Detection 3
LEDs 4
Arduino 5
Prototype 5
Different LEDs 5
The Pieces 6
Inside the Pieces 7
The mounting 8
The PCB 9
The price 9
Too many components 10
The Board 12
Testing and Assembly 12
PCB soldering 12
A problem 12
Prototyping ¼ 12
Testing all PCBs before assembly into board 13
Board assembly 13
Things I would have done differently 14
What I've learnt 14

Introduction

As a home educator who is looking to undertake a career in engineering, I was fortunate enough to secure work experience at Vitec, Bury St Edmunds. As part of my time at Vitec, I was to build a project of my own design, making use of their resources.

The specification

Online chess is unintuitive for me, the 2D diagram view doesn't feel natural. Unfortunately there's not always someone to play with in real life.

The solution: a real life physical chess board that interfaces with online chess so that I can play with online opponents on a real board.

The design

The two problems

There are two parts to the design problem:

1. How to move my opponent's piece on the board when they make their move
2. How to detect my move.

Moving the opponent's piece

At first, the main option I considered was having the opponent's piece moved robotically (perhaps using some kind of gantry above the board), but eventually I came to the conclusion that it would be beneficial in terms of cost and implementation to have it indicated to me in some way to move the opponent's piece myself.

I decided that the best way of indicating the opponent's piece to be moved was to have an LED inside each piece, and an LED on each square. Then the opponent's move could be shown by a flash on the piece to be moved, and a flash on the square to be moved to.

To supply power to the LED in the piece there were a number of solutions:

- A small coin cell or other battery inside each piece
- An induction loop under each square in the board with a corresponding one in each piece
- Some form of physical connection on the underside of the piece.

This was a decision to be influenced by the other problem – detecting my move.

Detecting my move

There were only two options I considered for this problem:

- A magnet in each piece with a reed switch in each square
- Some physical connection on the underside of the piece, so that the square could detect when the piece was sitting on it

Solution

It turned out that both problems could be solved by a physical connection on the bottom of the piece, so I decided that each piece would have a jack on the bottom, used for supplying power to the LED inside, and for detecting that the piece was present on a particular square. Each square would have a jack socket.

The jack would have 4 poles – two for power of the LED and two for detection of the piece. These detection connections would be shorted on the piece, so that when the piece was placed in any square's jack socket, the square would detect that a connection had been made.

Multiplexing

At this point it seemed that each square would have 5 lines running to it:

- A +5v detection output
- A detection input
- A +5v piece LED output
- A +5v square LED output
- LED ground

5 x 64 squares = 320 IOs

There was a way to reduce this significantly though.

Detection

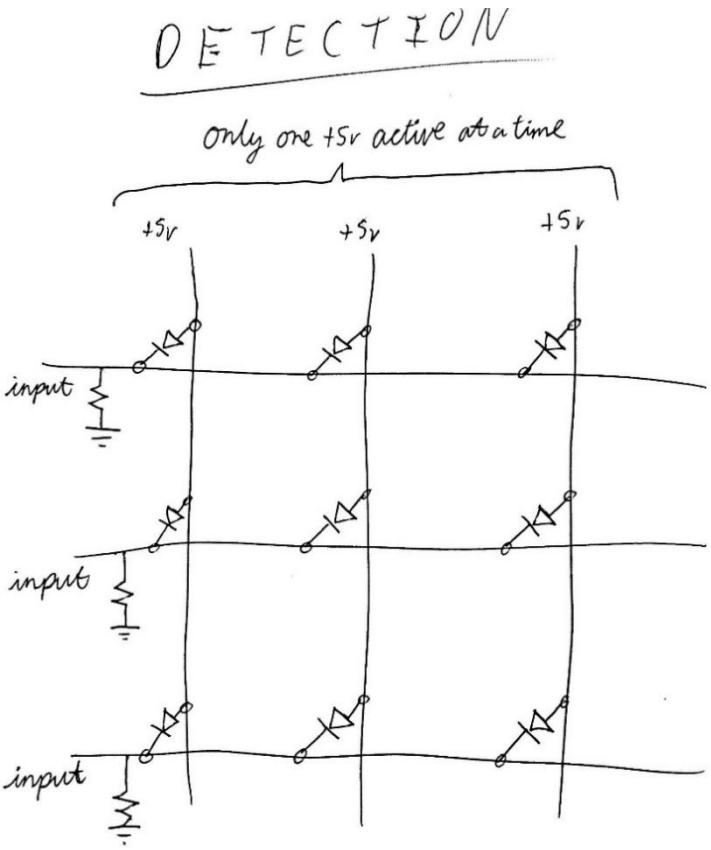


Figure 1: the detection circuit

Instead of having a detection output and a detection input going to each square, each column of squares would share a detection output and each row would share a detection input. Each column in turn would individually have its detection line turned on, and for each row on that column with a piece on, the detection input on that row would sense the piece.

Each connection that would be made by the chess piece would have to have a diode between it and the input, as otherwise the signal from one output line would flow through the connection, along an input line and end up electrifying all output lines, giving the impression of pieces where there are not.

The resistors on the input lines are pulldown resistors, to prevent those input lines floating when they are not being pulled HIGH by a connection.

LEDs

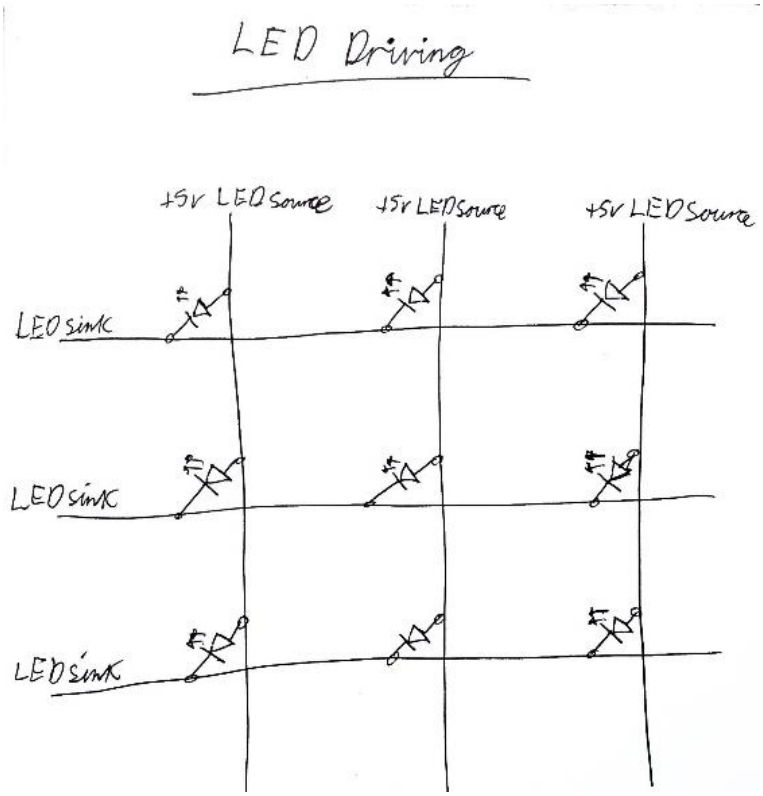


Figure 1: the LED circuit

A similar concept applies to the driving of the LEDs on a matrix.

By driving one output (the vertical columns) HIGH, and allowing one horizontal column to sink the current, each LED on the matrix can be targeted individually.

This webpage offers a clear explanation of the topic of multiplexing, and helped me to understand the concept.

Arduino

At this point, as I was to start prototyping and building physical elements of the board, I needed to make a decision on the microcontroller that I would use.

Options I considered included the Raspberry Pi, BeagleBone, mbed and Arduino.

Due to the number of IOs that would be required, and the fact that I was familiar with using it, I decided to opt for an Arduino Mega 2650.

Prototype

I decided that to fully grasp multiplexing, I would build a prototype of a 3x3 detection matrix, and a 3x3 LED matrix, as shown in this clip.

https://www.youtube.com/watch?v=DAw_sXPM9t0&list=UUOrvm1kAfi9ksAlnXbZeKbA

Different LEDs

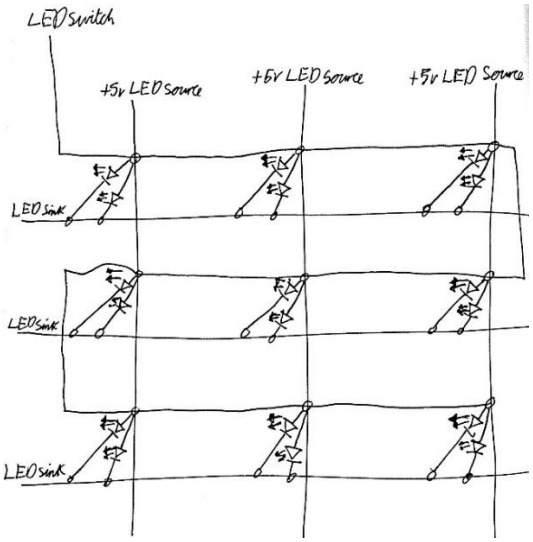


Figure 2: multiple LEDs solution 1

Each square on the board would need to control two LEDs: the one on the square and the one inside the piece. One option I considered to achieve this was to have one signal line that passed through every square on the board. When the signal line was being driven (ie, HIGH) and a particular square was targeted and supplied with power by the LED multiplexing, the LED of that square would light up. If the signal line was LOW and that square was supplied with power, the LED inside the piece sitting on that square would light up.

This seemed like a good solution to me, as only one extra IO line would be used to control double the number of LEDs. However, when I thought about how to implement this, I realised it might not be such a good idea. Either a relay, or some form of logic would have to be placed under each square to achieve this. In the end I decided against this option, as additional circuitry under each 64 squares was not only a financial cost but time cost in design and assembly as well. The solution I opted for was to have two LED driving lines running vertically through each square – one for the LED on the square and one for the LED inside the piece. This would increase the number of IOs needed by 8, but since I was going to use an Arduino mega, this would not be a problem as I would still have plenty available.

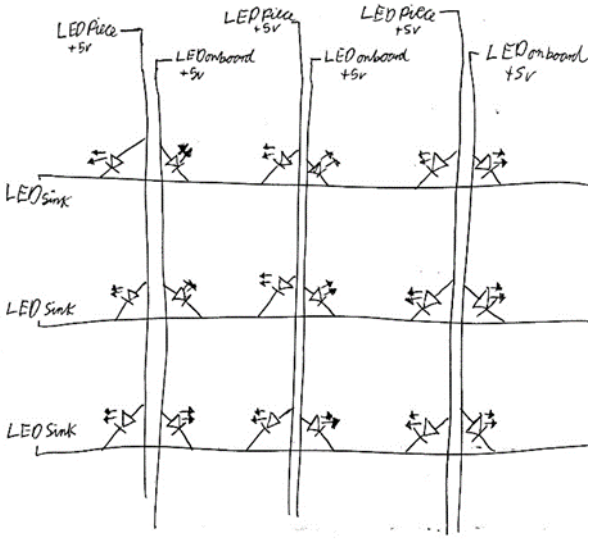
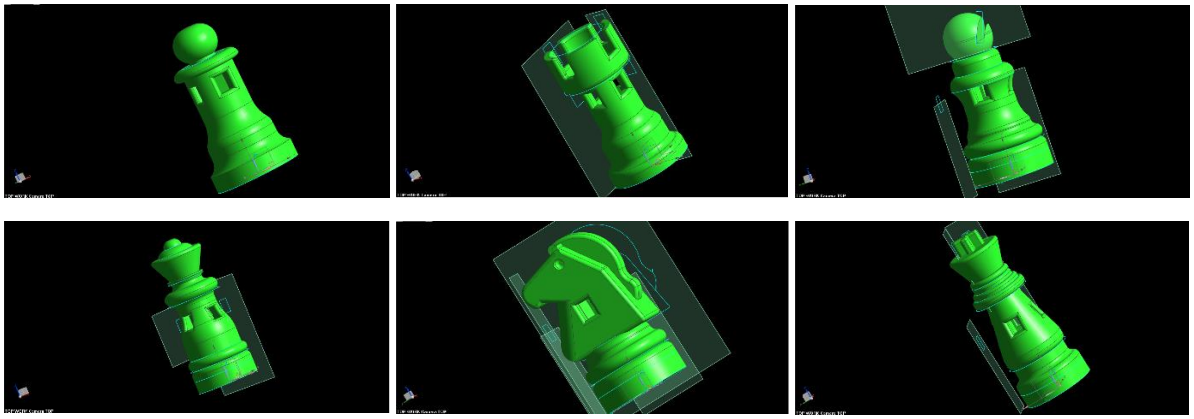


Figure 3: multiple LEDs solution 2

The Pieces



Figures 4.1: the computer models of the pieces

Since each piece was to have an LED inside, with a jack sticking out the bottom, it was clear that the pieces needed to be specially made.

The best option for manufacturing these pieces was to 3D print them, as the space to accommodate the jack and LED could be precisely produced. I designed the pieces using CAD (Computer Aided Design, as in Fig 4.1), then printed them out on the 3D printer. It was decided that it would be more cost effective to print all the pieces in black, then spray paint half of them white, as this way an additional printing filament wouldn't have to be used.

Figure 4.2: the completed pieces

Inside the Pieces

I decided that I would keep as little electronics inside the piece as possible. This meant having the bare minimum of a jack and LED inside the piece, with the LED's resistor and the signal diode (for the detection multiplexing) on the board itself.

Figure 5: the inside of each piece

Each jack needed to have two of its poles shorted (for the detection multiplexing), and an LED soldered to the other two poles. Since the two poles to be shorted were next to each other on the jack socket, I initially experimented with conductive glue to make the short. However, it wouldn't adhere to the contacts, so I decided to drop a small spot of solder over the gap between the contacts instead. I then soldered on the LED and repeated the process for each of the 32 pieces.

The mounting

My mentor and I discussed various ways of mounting the jack inside the piece. The main options we considered were:

1. Using the thread already present at the base of the jack (see above) to screw into the piece
2. Screwing a cover onto the jack contacts and glueing this inside the piece.

In the end we decided to opt for the second solution, as we weren't sure of the strength of the first option (bearing in mind the pieces will have to be repeatedly pulled and pushed into jacks), and it would have required the creation of a thread on the inside of the piece.

Figure 6: the completed jacks and jack sockets

The PCB

Initially, I planned to make the whole board one big PCB with a black and white chequer silkscreened on top. Since the board size was too big to manufacture, I decided to split it up into sixteen 2x2 squares, that would then be mounted on a frame. The squares would connect the horizontal and vertical signals to each other via header/receptacle pairs on each side and they would slot together and make one big PCB.

The price

It turned out that this would be too costly to manufacture, sixteen 10cmx10cm PCBs were just too expensive. The only option was to reduce the size, and as I didn't want to make the chessboard any smaller, I decided that instead of having a board made entirely of PCBs, I'd have a normal board with small PCBs inset in the centre of each 2x2 square. This way I could keep the same circuitry for a 2x2 square as before.

Of course having the PCBs not touching meant that connecting them with adjoining headers wouldn't work. Therefore, the connections between the PCBs would have to be wired.

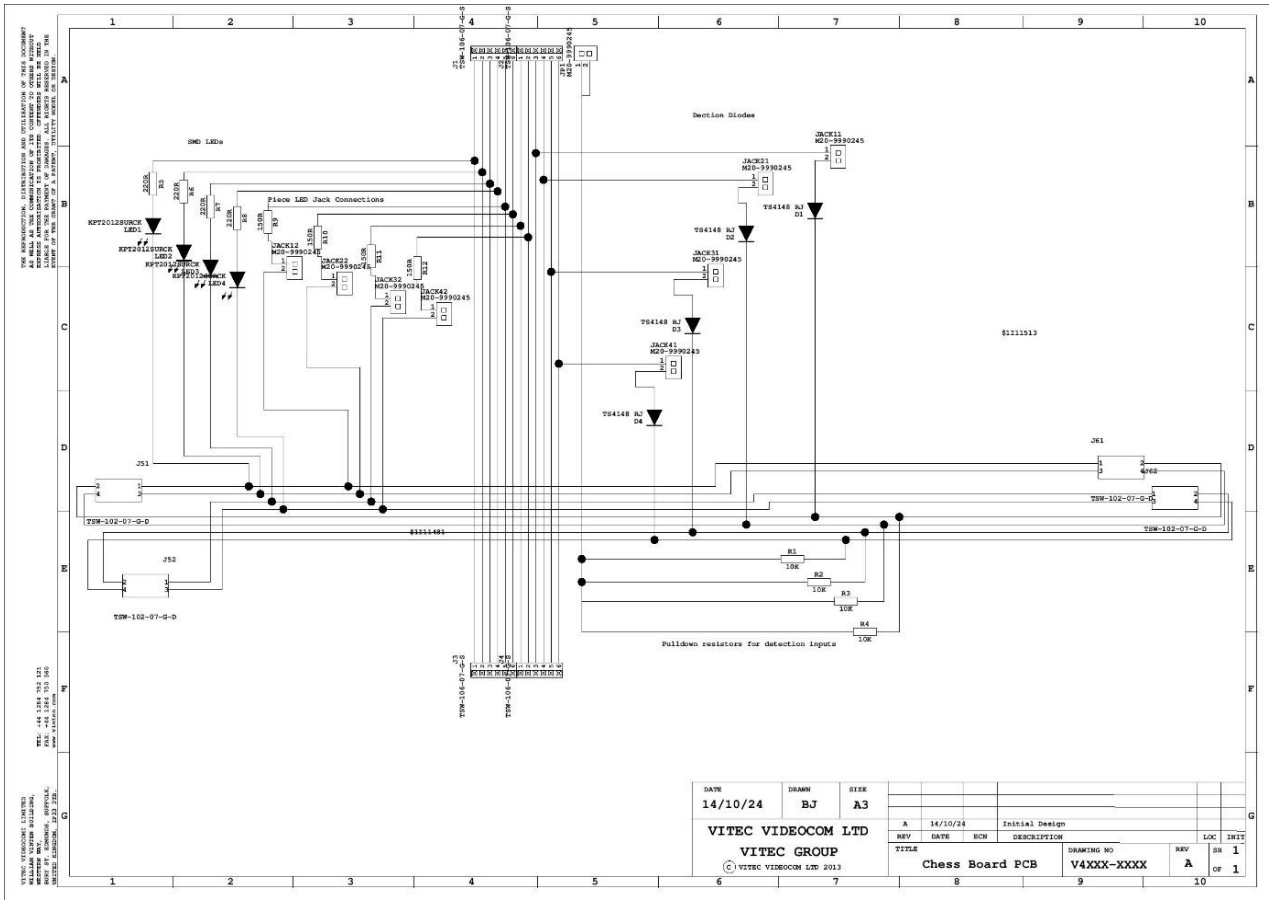


Figure 7: the first revision of the PCB circuitry

Firstly, I designed the circuitry for the PCB, and produced an electronic schematic, above. I hadn't used the circuit design software before, and my mentor greatly aided me in using it.

Next, I exported the circuitry into PCB design software, laid out the components and started to route the tracks, as seen in Fig 8.

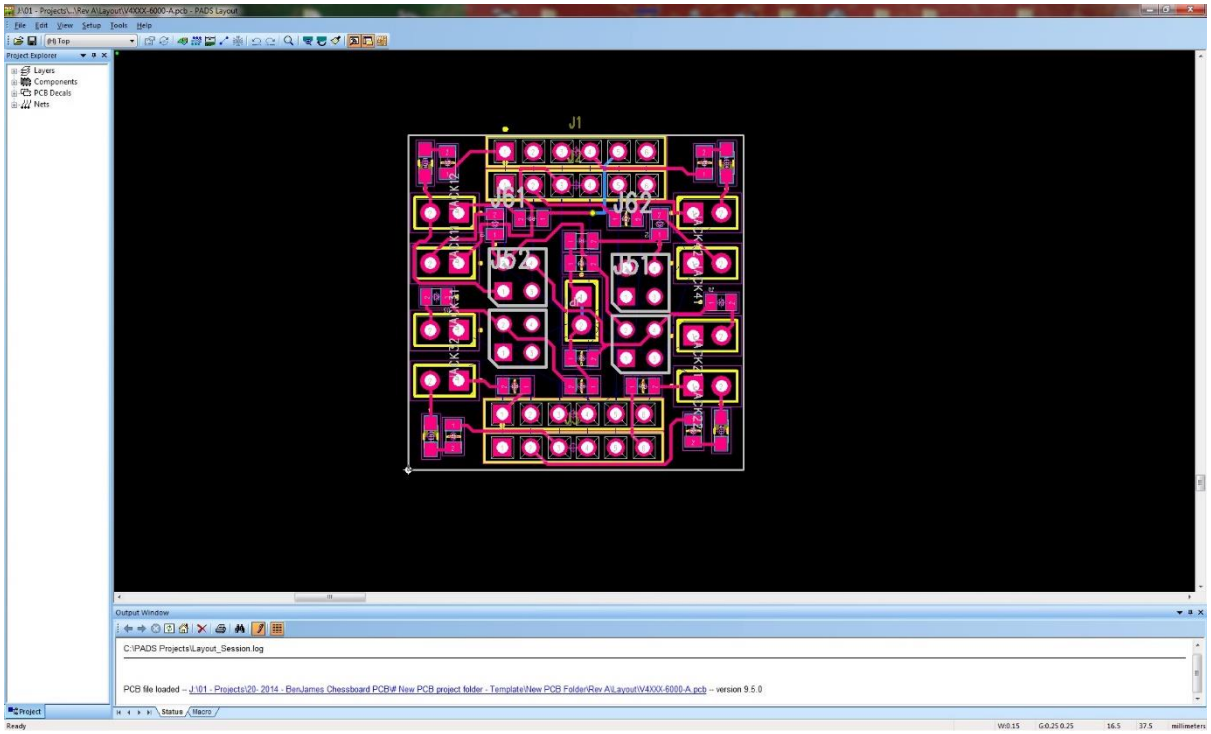


Figure 8: the first revision of the PCB layout

Too many components

However, I soon ran into difficulties, as the board was so small but highly populated with components. Routing the tracks soon became impossible with so many, and budget restrained the board to two layers.

I decided that instead of daisy chaining each board to the next, with a link between each to pass on the signal, I would split the wire below the PCB. This meant that half the number of sockets would be needed on the PCB. It would also make routing the tracks easier.

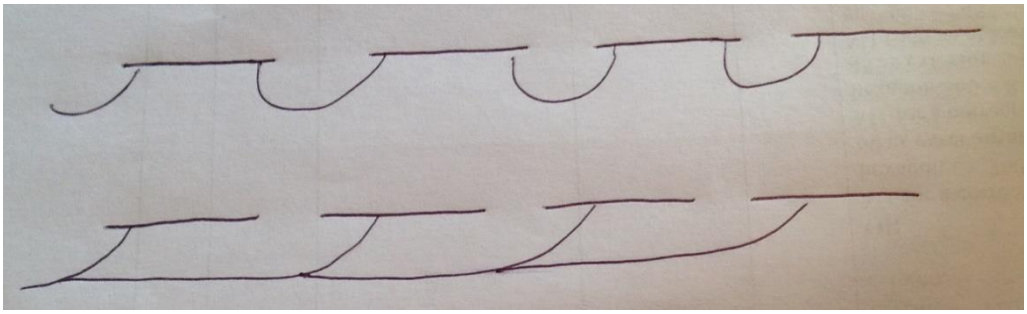


Figure 9: the two different designs

Unfortunately, what would be cut in the financial cost of the board would result in an additional time cost of producing and soldering the wire splices.

I went back and modified the circuit design (Fig 10), then exported this revision to the PCB design software (Fig 11). Since there were now less components, I was able to fully route all the PCB tracks.

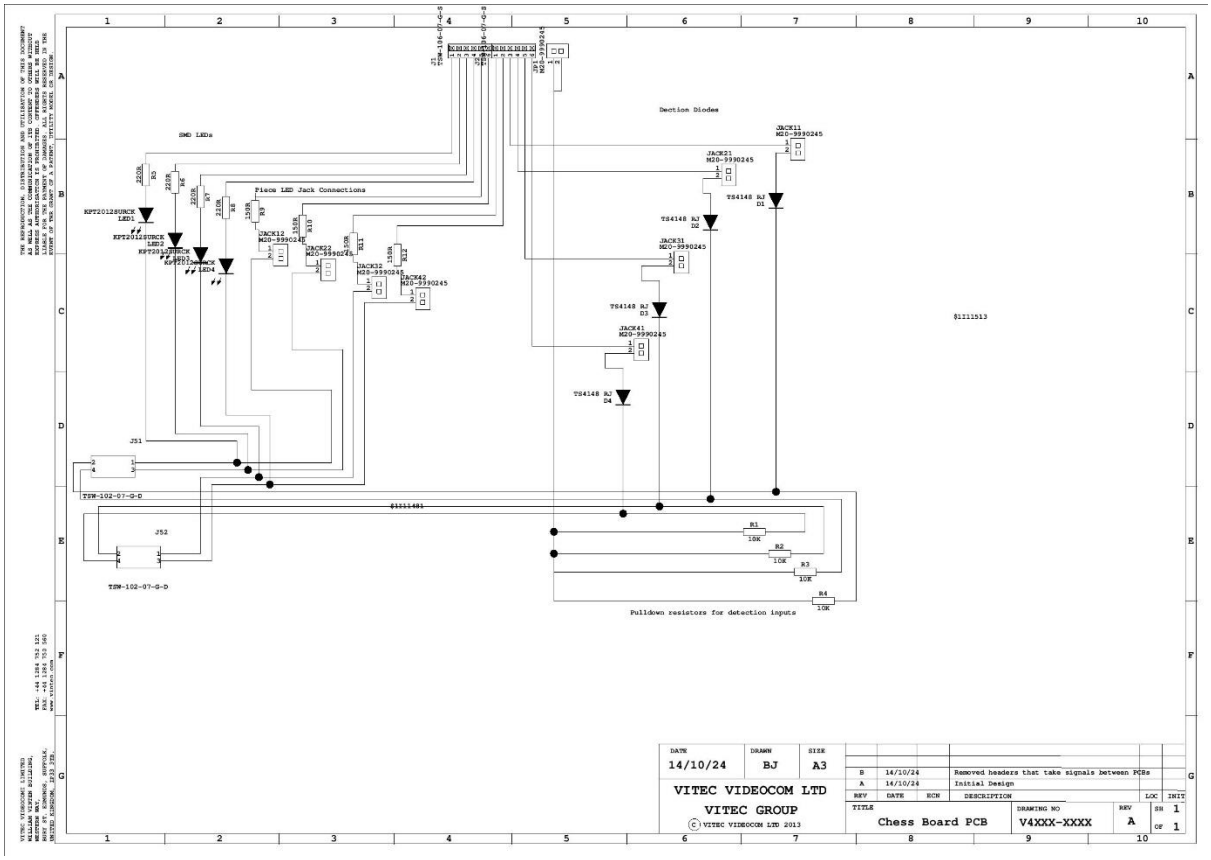


Figure 10: the second revision of the PCB circuitry

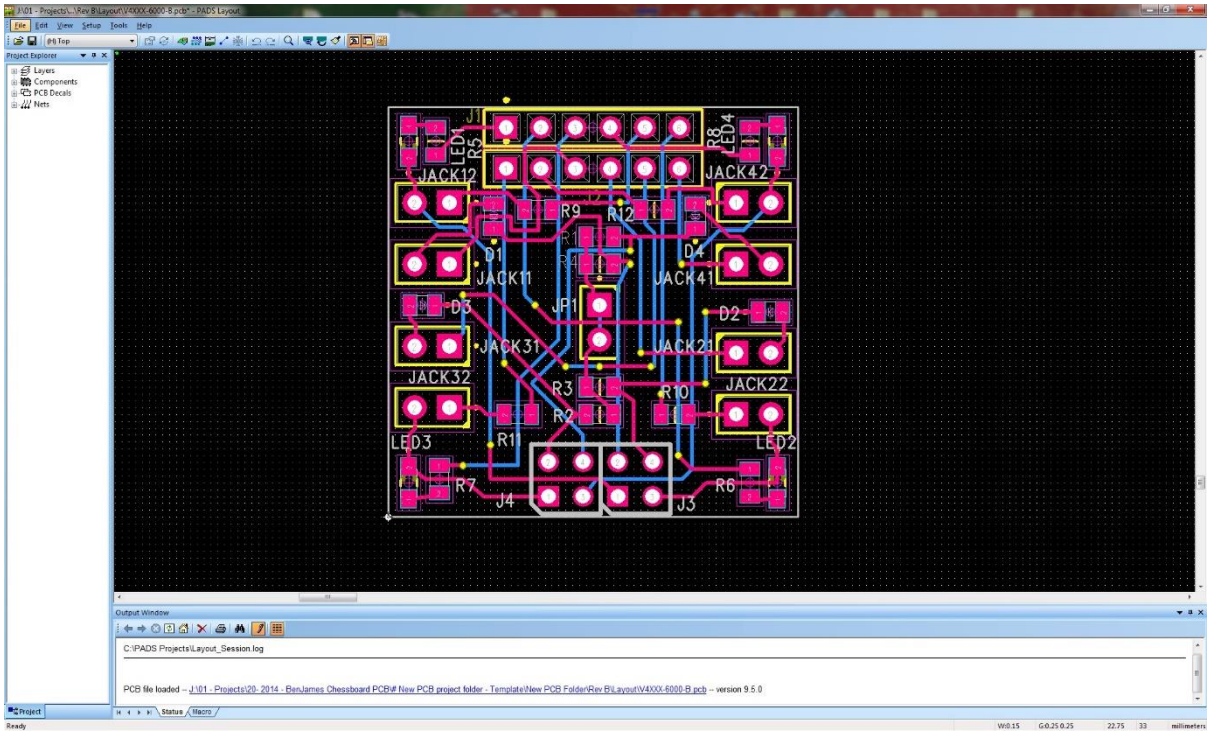


Figure 11: the second revision of the PCB layout

The Board

As mentioned above, the initial plan was to have the whole board as one PCB, but when this wasn't possible due to the cost of the PCBs, I decided to have a plastic square for the board, with holes milled for the PCBs and jack sockets. However, the milling machines which were due to be used to produce this were out of action, and therefore a board was designed that could be 3D printed in 4 quarters.

The quarters of the board would be glued together then mounted in a frame, which was to be machined from metal.

Testing and Assembly

PCB soldering

A problem

Once the PCBs had been manufactured and delivered, I set about soldering the SMD (Surface Mount Device) components onto them. However, as I consulted my schematics to check where each component went on the board, I realised that I'd made an error.

Each PCB controls a 2x2 matrix – 4 squares. I realised that when I'd designed the PCB I'd given each of the 4 squares its own connections when in fact, because of the multiplexing, each 1x2 column and each 2x1 row should share its connections.

This meant that for each duplicate connection I'd made on the PCB, I'd have to short the two connections together, to connect the appropriate row/column. Once I'd figured out which connections to short, I started to solder on the SMD components and the wire shorts. Since I had not done any surface mount soldering before, my mentor taught me how to do this.

Prototyping ¼

I decided that before I started linking all the PCBs together I would link just 4 of them together in one line (a quarter of the board), and test that this much worked. Since each PCB controls 4 squares I'd have an 8x2 matrix of squares.

To enable testing, I wrote some code for the Arduino that tested the detection, as well as the LEDs on each square and piece. Thankfully, after a few soldering joint fixes, everything worked as planned. This is a video which shows the LEDs in the pieces and the board working, and flashing in sequence (to prove that they are all controllable individually). http://youtu.be/HzorZ0hQMmQ

The next step was to solder the rest of the PCB columns together, then to link them horizontally. Once all the PCBs were linked together, I begun to solder the jack sockets onto each one. This soldering was the most time consuming part of the project; I completed over 2000 joints in total.

Testing all PCBs before assembly into board

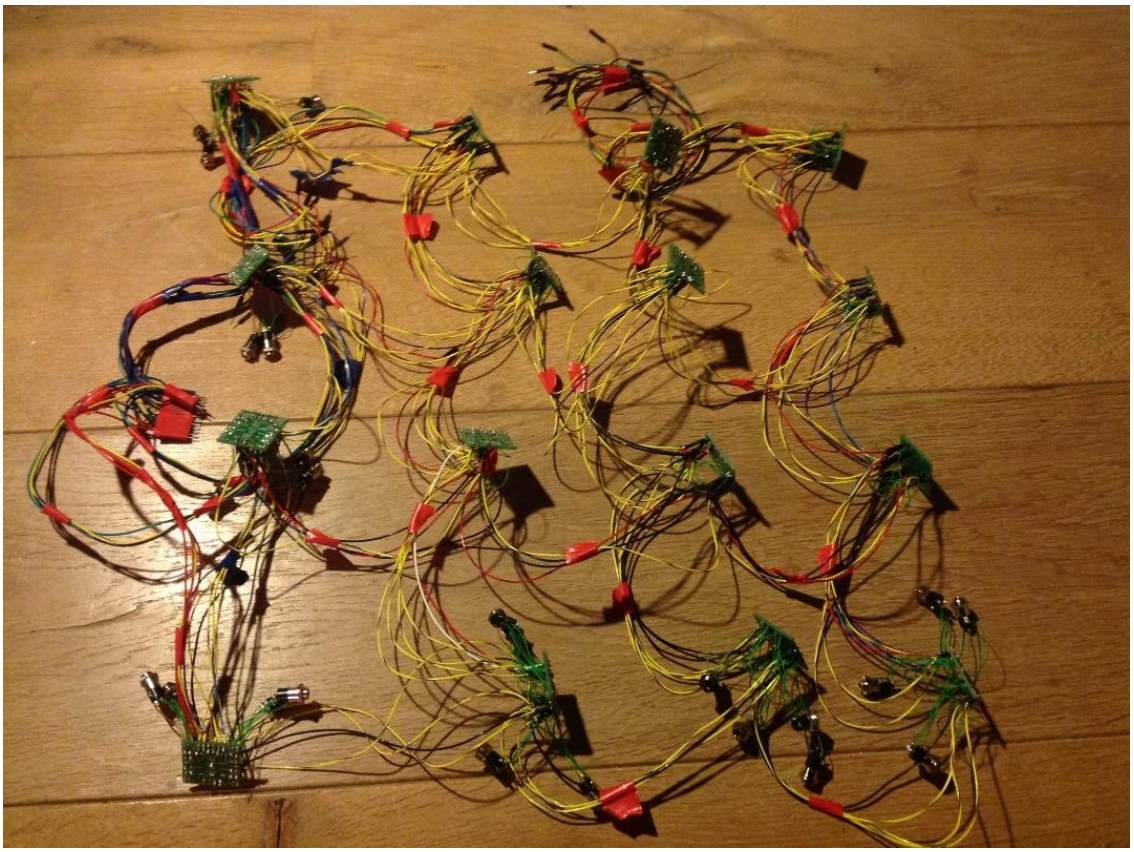


Figure 12: the wired PCBs before placement into the board

Before mounting the PCBs into the board, I needed to be sure they worked. I wrote more test code for the Arduino, this time for the entire board. The majority of the PCBs were working, though a few faulty connections were found, but were able to be fixed before assembly with the board.

Board assembly

Once all the quarters were 3D printed, I glued the board together. Unfortunately, I had forgotten the fact that there was an extra 1mm on two adjoining edges of each board, for mounting on a frame. Therefore, when I glued the quarters together some of them were misaligned, meaning one edge had to be filed down for the whole board to fit into the frame.

Things I would have done differently

- I would have used three pole jacks, with 4 pole sockets, as this would have meant that I wouldn't have had to short two of the poles on the jack, saving time on soldering.

- I would have reviewed the circuitry on the PCBs more closely before they were manufactured, and would have thought more about the connections between them. This would have meant that the error I made would have been avoided.

What I've learnt

- This project was the first time I'd used 3D printing and the CAD practices to facilitate it. I learnt how to create parts in CAD ready for production on a 3D printer.

- Designing the PCBs involved first creating the circuitry on an electronic schematic, then laying out the components as they would be on the PCB. I had to learn how to route the tracks between components on the board, and the techniques and factors to consider when doing this.

- Assembling the PCBs required me to solder the SMD components onto them. I hadn't done any soldering of this kind before, so I acquired a new skill learning how to do this.

- I feel like I've learnt a lot about the whole design-to-manufacture process, from an initial idea through to a finished product. I learnt how the first stages of specification, problem-solving and design are vital as they lay the foundations for everything ahead.

- Other skills I've acquired include: time management, report writing, planning and documenting stages of the build process.