eChess: magnetically actuated tokens with PCB coils

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor: Stefan Petkov Donkov

Directores: Asier Marzo, Josu Irisarri

Pamplona, fecha de defensa 6/9/2021

Acknowledgments

I want to thank all the members of UpnaLab that have shown interest, assisted and helped me push through the difficulties and hurdles encountered during the development of the project.

I especially want to thank my director, Asier Marzo, for his support to the project.

Summary

The project consists in the design, construction and programming of a device capable of electromagnetically controlling the pieces of a chess board. The device will consist of two parts, one consists of a matrix of magnetic field sensors in charged of detecting the movement made by the user playing with the board, the other part has the function of moving the pieces around the board by the means of a matrix of printed circuit board coils controlled by a microcontroller. With this device a user would be capable of playing chess, physically against a computer that would make its moves automatically on the board. It would be possible to switch the game board and program the computer to play any other board game against it.

Keywords

Chess, Automation, Electromagnetic Linear Actuation, Electromagnetic Sensors, PCB, PCB Coils, Magnets, Arduino, Python

Index

Objectives

The main objective of this project is to develop a working prototype of an automatic chess board that allows for the fully automated movement of the pieces on the board while at the same time, being able to detect the moves a human player could make. This would allow for a real game of chess to be played between a human and a chess engine on a physical board, without the need for an intermediary to make the moves for the engine.

In order to achieve the goal described above we have to, on one hand, design and test different methods of sensing the movement made by the human player. This includes considering different microcontrollers, different sensors and sensor arrangements in order to maximize speed, reliability, user experience and feel while keeping cost to a minimum.

On the other hand, we have to develop a magnetic linear actuation system that is able to move one or more pieces at the same time in a relatively fast manner while having the smallest footprint possible. We have to consider all the different ways of interacting and modifying the magnetic field.

The last aspect of the project we have to consider is the software. We must interpret the information received by the sensor system and translate it into a chess move. We also have to obtain the move a chess engine wants to make, and translate it into which series of actuators have to be powered in order for the chess piece to perform it, this information must then be sent to the microcontroller.

State of the Art

Currently there are several different types of automatic, electronic or intelligent chess boards, some like the DGT e-boards [1] or the MILLENIUM chess computers [2], focus on movement detection of the user and integration with popular online chess portals. The DGT e-boards detect the player's move without any extra action, you just play chess normally, but some of the MILLENIUM chess computers require the player to push down on the piece before and after they have made the move in order to detect it.

Figure 1 ChessGenius chess computer [2]

Figure 2 DGT Smart Board [1]

However when it comes to being also able to move the pieces on its own, we only found 1 board capable of it, Square Off [3].
It is able to detect a player's movement without forcing them to exert pressure against the board, and is able to move the pieces by means of an electromagnet.

Figure 3 Square Off chess board [3]

There is a kickstarter that was later revealed to be a scam [4] that promised to do all Square Off does but better and with a lower profile, Regium Chess. Their website no longer exists, but there are archives of it here [5] and here [6].

Figure 4 Regium chess board [5]

Regarding electromagnetic linear actuation of magnets, there are interesting videos on the topic like in this one [7], in which some theoretical and simulated analysis on the magnetic fields produced by PCB coils, as well as some practical demonstrations of linear actuation of magnets using the coils.

There are also videos about the use of arrays of electromagnets to move magnets around [8] [9] [10].

There is also a channel on Youtube, called Carl Bugeja [11], in which a lot of different PCB coil designs and applications are explored, like linear actuators, motors and robots.

System

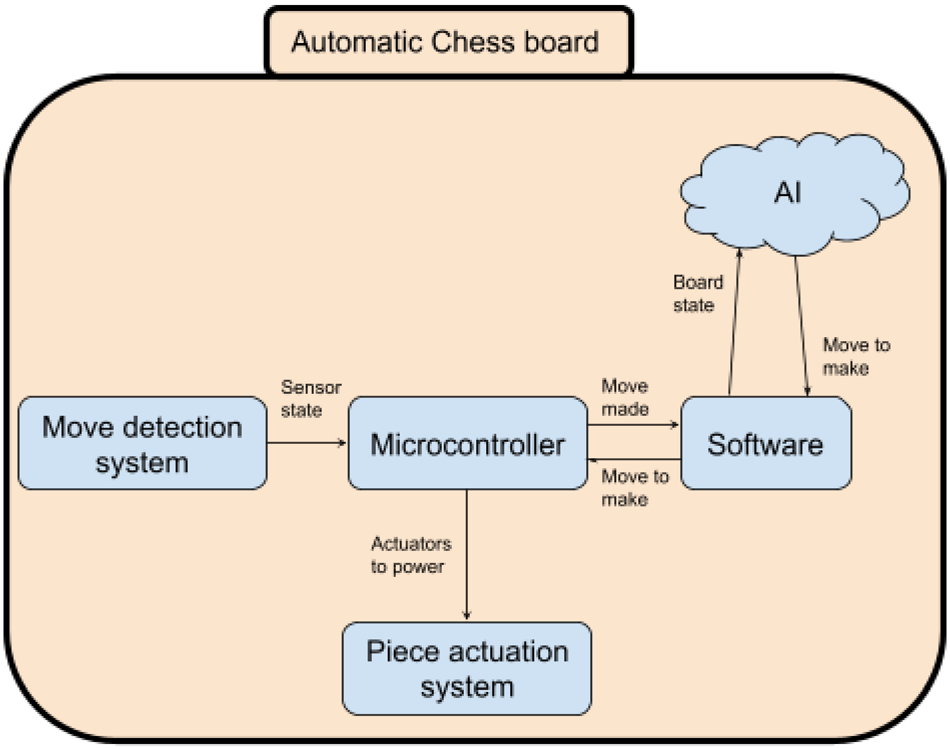


Figure 5 Diagram of components

In the diagram above, we can appreciate all the components, and the interactions between them, necessary in order to develop an automatic chess board.

The movement detection system must sense a change in a chess piece's position, and send this information to a microcontroller, which would translate this information into the chess move that the user has made. The move has to be communicated to the software in order to keep track of the board state and decide the next move to be made by the AI. The microcontroller must now receive the information of the move to be made and power the required actuators in order to carry the move out.

1. Movement Detection System

This system is in charge of detecting the chess move a human could play on the board.

For achieving this goal, the detection system had to meet several criteria:

- Must be reliable, meaning it should not fail to detect a move or stop working.
- Must be accurate, in the sense that it must not detect a move different than the one that has been made.
- Must be fast: the user should not have to wait for the system to do its job, it should be the other way around, the system should wait for the user.
- Should not interfere with other systems, it should not cause any problems in the actuation of the pieces or the software.
- Should be user friendly. The user should understand how the system works intuitively or with minimal instructions, and be able to easily remember it.

1.1. Methods of detecting pieces

Several methods of detecting the chess pieces were considered:

1.1.1. Magnetic Switches (Reed Switches):

This type of switches, for example the MDCG-4 [12], have several inconveniences for our purpose.

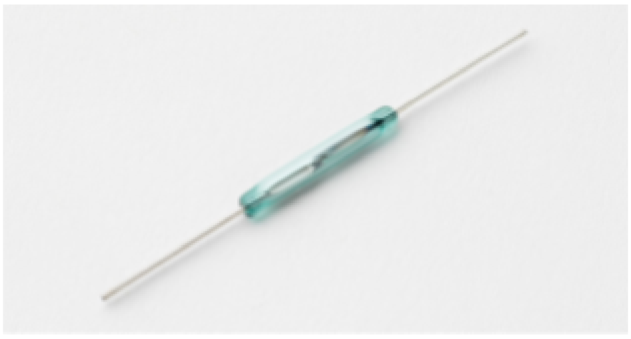


Figure 6 Reed Switch [12]

One of the inconveniences is their dimensions, with around 15mm of length and 2.3 mm in diameter, even while being on the small side, compared to other models, they still have a large footprint when compared to alternatives like hall effect sensors.

Another inconvenience is the fact that they are quite fragile since they are made out of glass.

There are two limitations that we deemed critical when discarding these switches as the goto option for the system. The first one has to do with the fact that their output is digital, meaning we can only detect if the switch is open or closed, this on itself isn't a problem, but combined with the fact that in order to actuate the switch, the magnetic field has to be close to perpendicular to the plane described by the internal contacts of the switch, it doesn't allow for any tuning of the system.

These factors directly interfere with the reliability, accuracy and user friendliness of the system, since the pieces would have to be very accurately placed in the middle of the square or risk actuating a switch of a contiguous square.

1.1.2. Hall effect sensors:

These kinds of sensors, like the DRV5053 [13], work by measuring the intensity of the magnetic field acting upon them and outputting an analog voltage relative to it.

These sensors address all the limitations and inconveniences of the magnetic switches. Their footprint is much smaller, 3.25x4.1x1.62mm, the output is analog and there is no need for the chess piece to be perfectly placed in the middle of its square.

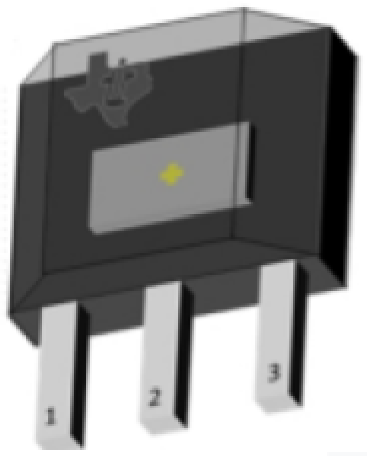


Figure 7 Hall effect sensor [13]

These features allow us to be able to set and adjust threshold voltages to detect when a piece is over the sensor. Even if an adjacent sensor could partially sense the magnetic field, its output would be significantly lower than that of the intended sensors output, meaning we can set a threshold voltage that would allow us to differentiate between a piece passing over the sensor while it is being moved, a contiguous sensor detecting the piece partially, and a piece being placed in its intended position.

Since the system is adjustable, we can allow the user to be less strict when using the system, we can allow less precision when placing the piece on the board and more freedom of movement, in the way of not forcing the user to have to lift the piece off the board to an unreasonable height.

All these features let us have an extremely reliable and accurate system while at the same time not hurting its user friendliness and keeping the footprint extremely small and compact. For these reasons we decided to use hall effect sensors for the design and construction of the movement detection system.

1.2. Sensor Matrix

We first decided to test the viability of the sensor matrix in a smaller scale than the one that would be used in the end, 3x3 instead of 8x8, since it would simplify the prototyping and still prove if the system would work or not.

We proposed the powering of the complete array of sensors and then the individual reading of their analog output values. This would work for the small 3x3 proof of concept, since we would only need 9 pins of a microcontroller like the arduino to measure the voltages, but if we go the 8x8 scale, we quickly realize that this method of powering and measuring the sensors is unviable, since there are no ratherly available microcontrollers with 64 analog pins, and adding a multiplexer would bring more complexity and decrease the speed of the system significantly.

For the above described problem we went for the following approach:

- Power the sensors by rows
- Read output values by columns

With this method we are able to read the analog values of 8 sensors at a time, bringing down the total number of pins necessary from 64 down to 16, 8 digital pins to power the 8 rows of sensors, and 8 analog pins to read the analog output voltages of the 8 columns of sensors.

A multiplexer could still be used to reduce the number of pins even further, but we have deemed it unnecessary since most microcontrollers already have much more than 16 pins and the implementation complexity and speed costs far outweigh the number of pins cost.

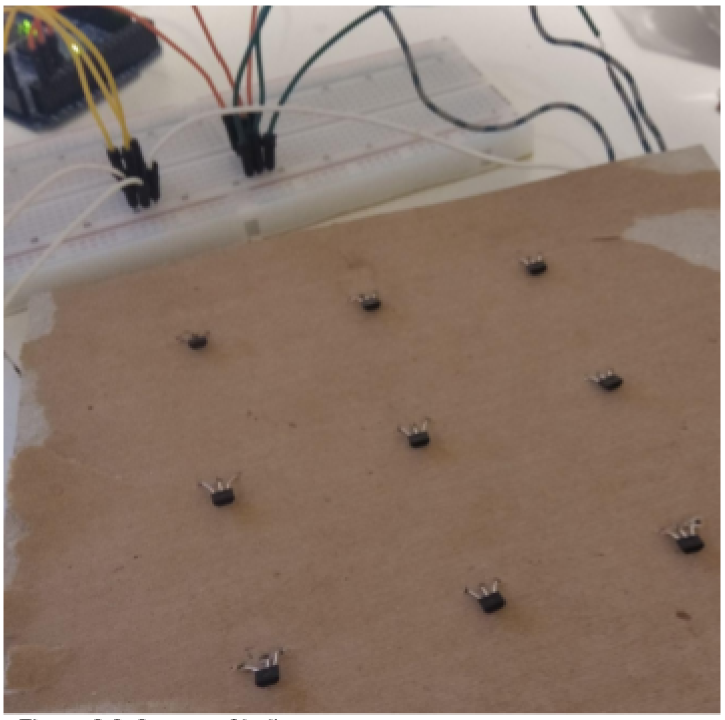


Figure 8 3x3 array of hall sensors

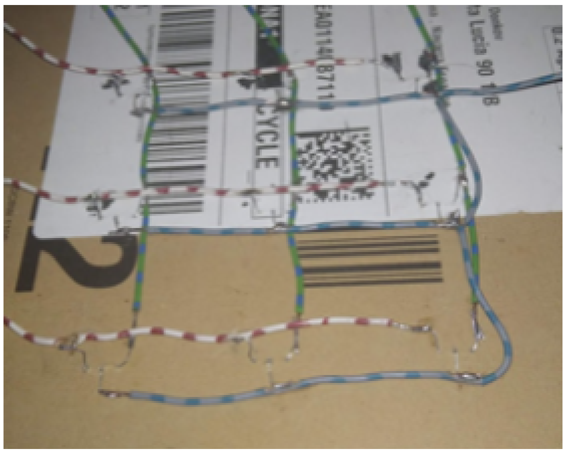


Figure 9 Wiring of the sensor matrix

In the figure 9 we can see how the sensors are wired, the grey blue striped wire is the common ground for all the sensors, the red striped wires have the Vcc pins of the the sensors of each row wired in parallel, and the green blue striped wire has the output pins of sensors of each column wired in parallel.

The complete assembly of the 8x8 matrix of sensors can be seen below, it uses a laser cut piece of wood in order to have the sensors equally spaced and secured to easy assembly when it comes to soldering.

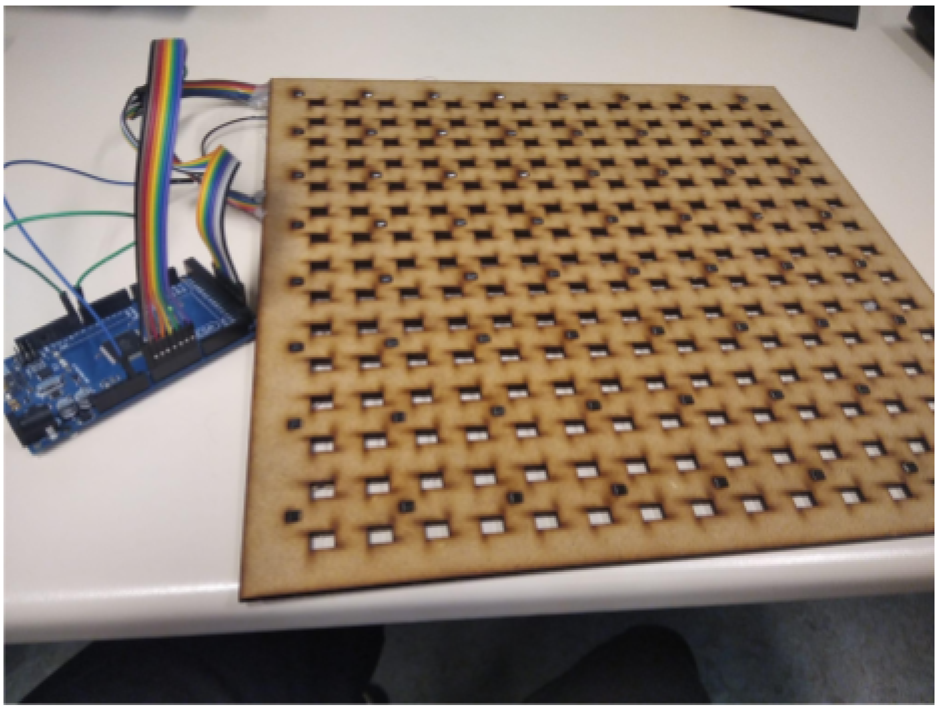


Figure 10 8x8 array of hall sensors

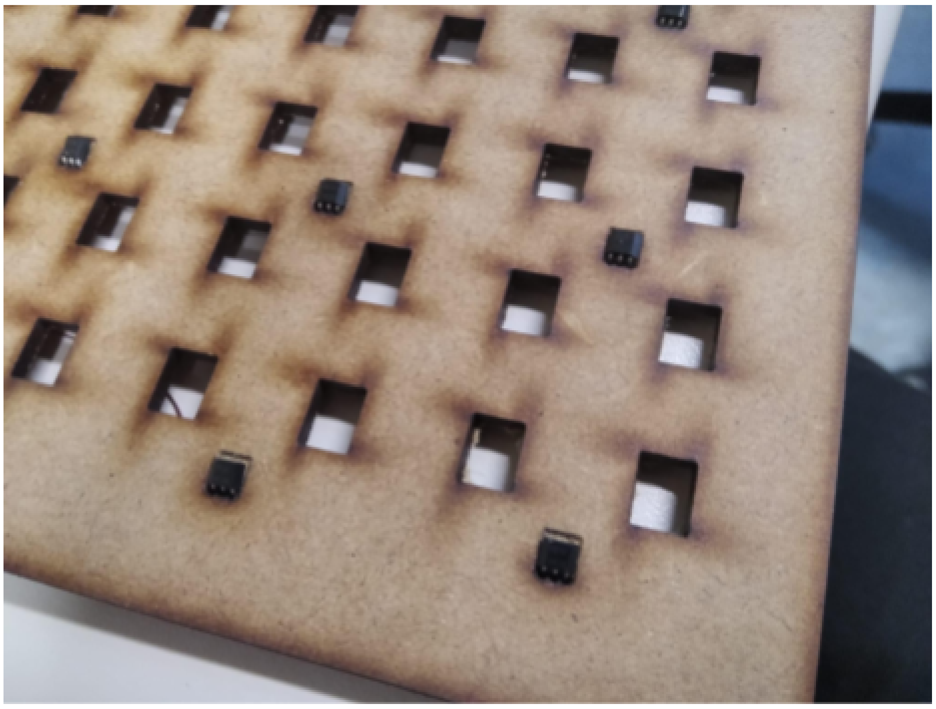


Figure 11 Close up picture of the 8x8 array of sensors

1.3. Microcontroller code

With the above described sensor array, we are able to detect whenever a piece is lifted off the board and when it is placed on the board. We can keep track of the state of the board before and after the move is made. However only with this information there is ambiguity when trying to find out the move that was made by the user, for example the following initial and final positions:

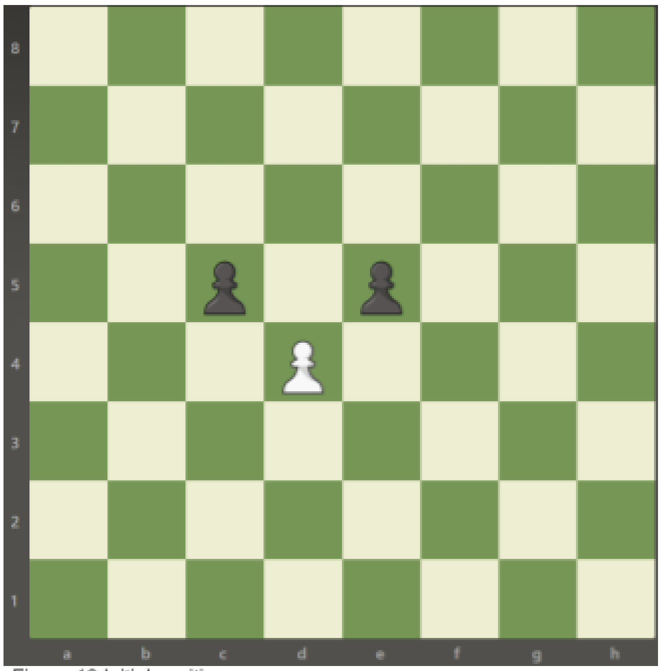


Figure 12 Initial position

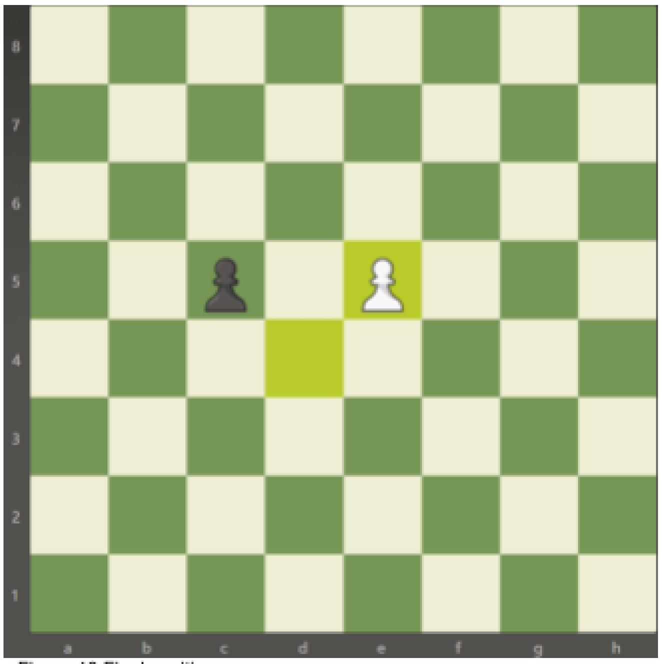


Figure 13 Final position

We would not know which of the pawns was taken by white, since the only difference we would find when comparing the initial and final state of the board is that there is no longer a piece in position d4.

In order to overcome this inconvenience we must also keep track of several more actions the user could do:

- Which position has changed at the start of the move
- If any which positions changed during the move

For example, for the figures above, we first have to detect when the d4 pawn is lifted off the board, after that we must detect when the e5 pawn is lifted off the board, and finally we must detect when the white pawn is placed back on e5. We can then wait for the user to press a button to signal that he has completed the move or wait an arbitrary amount of time.

Since this process is critical or else the reading of the move made would be erroneous, we must specify a protocol to be followed by the user whenever he/she is to make a move, the following steps have to be followed:

- When moving a piece normally:
    - Lift the piece off its square
    - Place the piece on the desired position
    - End turn
- When taking an opponent's piece:
    - Lift your piece off its square
    - Lift your opponent's piece off its square
    - Place your piece on the desired position (takes into account en passant [14])
    - End turn
- When castling [15]:
    - Lift the king off its square
    - Place the king on the desired square
    - Lift the rook off its square
    - Place the rook on the desired square
    - End turn

Since we can read the value of the sensors several times per second there is not any problem as it should be physically impossible for a user to move a piece in less time than what it takes to execute the code.

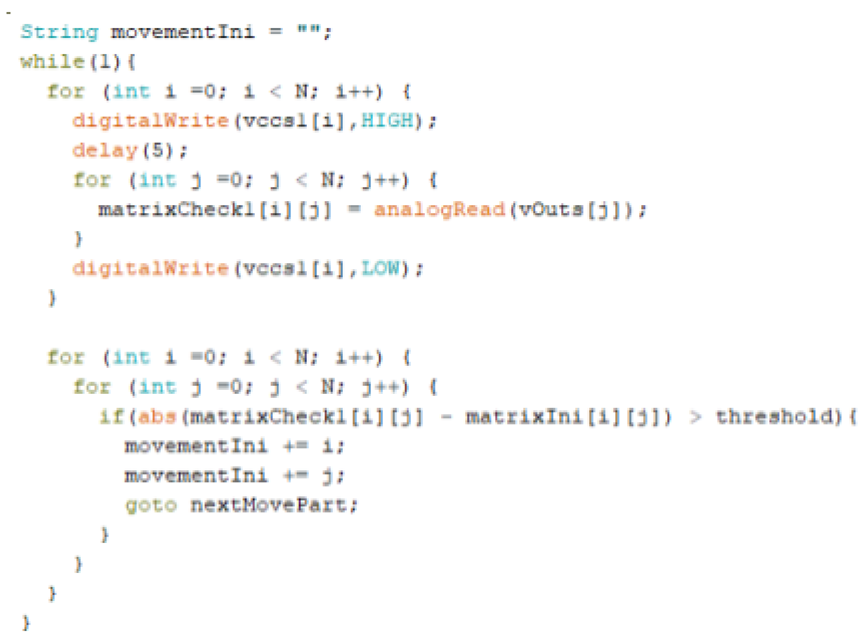
The code implementing the movement detection on an Arduino Mega microcontroller is made available here:

https://gitfront.io/r/Sugarb0y/6d426b9c6a30af4b2aa1fd28a4547670a4b38d61/eChess/blob/integrationTestMoveDetection-SerialCommunication.ino

A video demonstration of the movement detection system interacting with the software can be found here:

https://youtu.be/PWWIM00yKk8

The function in charged of the movement detection is getUserInput():

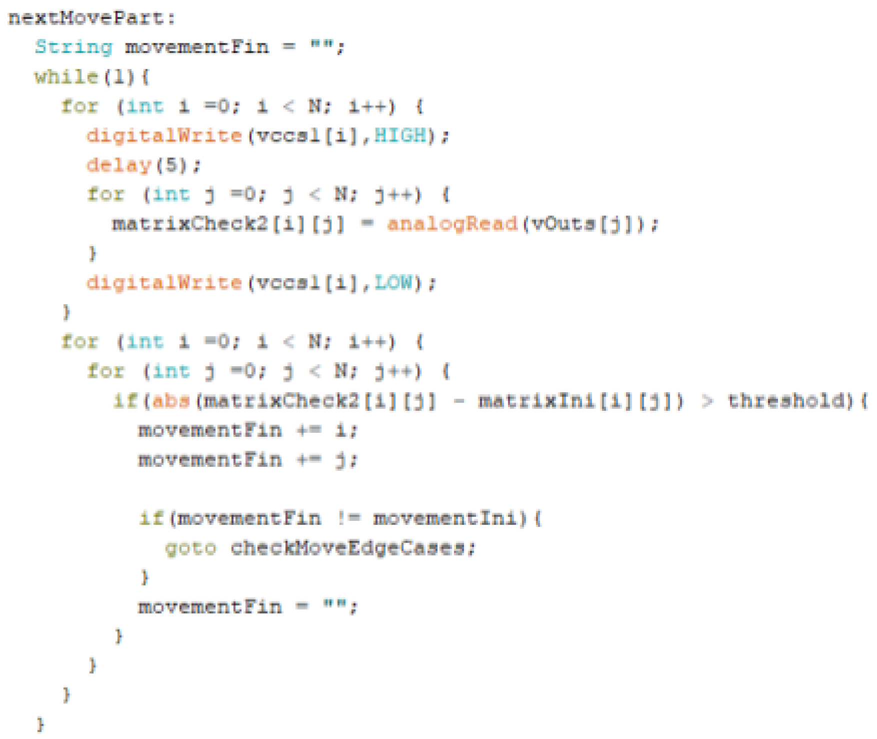


```
String movementIni = "";
while(1){
  for (int i =0; i < N; i++) {
    digitalWrite(vccs1[i],HIGH);
    delay(5);
    for (int j =0; j < N; j++) {
      matrixCheck1[i][j] = analogRead(vOuts[j]);
    }
    digitalWrite(vccs1[i],LOW);
  }

  for (int i =0; i < N; i++) {
    for (int j =0; j < N; j++) {
      if(abs(matrixCheck1[i][j] - matrixIni[i][j]) > threshold){
        movementIni += i;
        movementIni += j;
        goto nextMovePart;
      }
    }
  }
}
```

Figure 14 getUserInput() part1

It first loops infinitely until a piece is lifted off its square, stores the coordinates of that square and then jumps to the next label

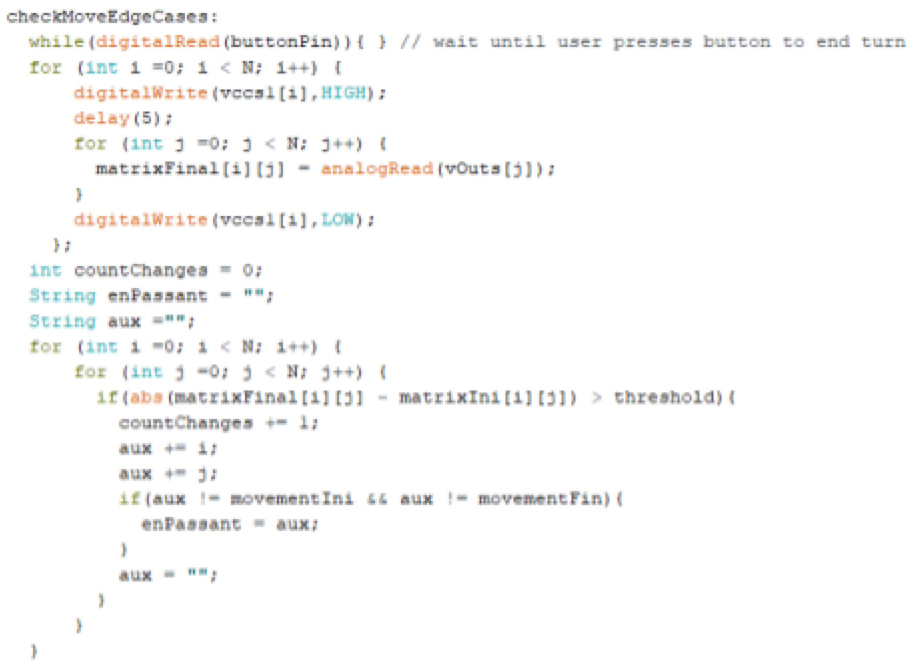


```
nextMovePart:
  String movementFin = "";
  while(1){
    for (int i =0; i < N; i++) {
      digitalWrite(vccs1[i],HIGH);
      delay(5);
      for (int j =0; j < N; j++) {
        matrixCheck2[i][j] = analogRead(vOuts[j]);
      }
      digitalWrite(vccs1[i],LOW);
    }
    for (int i =0; i < N; i++) {
      for (int j =0; j < N; j++) {
        if(abs(matrixCheck2[i][j] - matrixIni[i][j]) > threshold){
          movementFin += i;
          movementFin += j;

          if(movementFin != movementIni){
            goto checkMoveEdgeCases;
          }
          movementFin = "";
        }
      }
    }
  }
```

Figure 15 getUserInput() part2

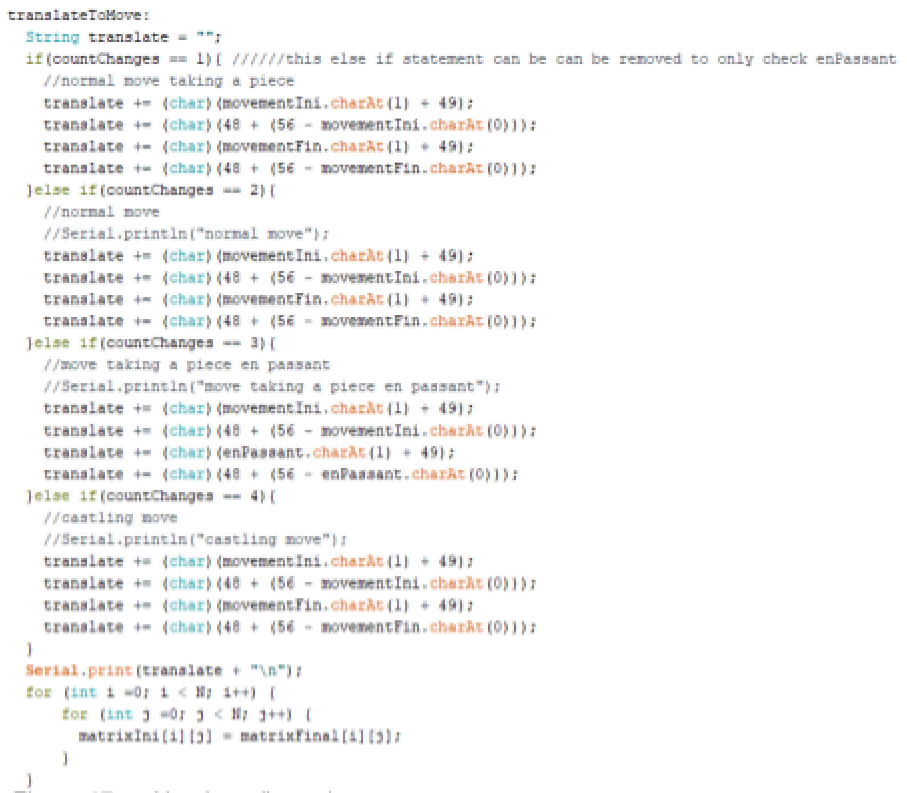
It then loops infinitely checking the values of the sensors until one of them changes over a threshold, meaning a piece was either placed on or removed from that square, it stores the coordinates of that square then jumps to the next label



```
checkMoveEdgeCases:
  while(digitalRead(buttonPin)){ } // wait until user presses button to end turn
  for (int i =0; i < N; i++) {
      digitalWrite(vccs1[i],HIGH);
      delay(5);
      for (int j =0; j < N; j++) {
        matrixFinal[i][j] = analogRead(vOuts[j]);
      }
      digitalWrite(vccs1[i],LOW);
    };
  int countChanges = 0;
  String enPassant = "";
  String aux ="";
  for (int i =0; i < N; i++) {
      for (int j =0; j < N; j++) {
        if(abs(matrixFinal[i][j] - matrixIni[i][j]) > threshold){
          countChanges += 1;
          aux += i;
          aux += j;
          if(aux != movementIni && aux != movementFin){
            enPassant = aux;
          }
          aux = "";
        }
      }
  }
```

Figure 16 getUserInput() part3

We now have to wait until the user presses the button indicating his move has finished. We then read the final position of the board and count the changes with respect to the initial position. If there are more than 2 changes it stores the third one to remove the need to calculate the final coordinate of an en passant move.



```
translateToMove:
  String translate = "";
  if(countChanges == 1){ //////this else if statement can be can be removed to only check enPassant
    //normal move taking a piece
    translate += (char)(movementIni.charAt(1) + 49);
    translate += (char)(48 + (56 - movementIni.charAt(0)));
    translate += (char)(movementFin.charAt(1) + 49);
    translate += (char)(48 + (56 - movementFin.charAt(0)));
  }else if(countChanges == 2){
    //normal move
    //Serial.println("normal move");
    translate += (char)(movementIni.charAt(1) + 49);
    translate += (char)(48 + (56 - movementIni.charAt(0)));
    translate += (char)(movementFin.charAt(1) + 49);
    translate += (char)(48 + (56 - movementFin.charAt(0)));
  }else if(countChanges == 3){
    //move taking a piece en passant
    //Serial.println("move taking a piece en passant");
    translate += (char)(movementIni.charAt(1) + 49);
    translate += (char)(48 + (56 - movementIni.charAt(0)));
    translate += (char)(enPassant.charAt(1) + 49);
    translate += (char)(48 + (56 - enPassant.charAt(0)));
  }else if(countChanges == 4){
    //castling move
    //Serial.println("castling move");
    translate += (char)(movementIni.charAt(1) + 49);
    translate += (char)(48 + (56 - movementIni.charAt(0)));
    translate += (char)(movementFin.charAt(1) + 49);
    translate += (char)(48 + (56 - movementFin.charAt(0)));
  }
  Serial.print(translate + "\n");
  for (int i =0; i < N; i++) {
      for (int j =0; j < N; j++) {
        matrixIni[i][j] = matrixFinal[i][j];
      }
  }
```

Figure 17 getUserInput() part4

We now translate the coordinates we have stored into a String with the move in the UCI chess notation [16].

This code could be simplified but has been left like this to facilitate changes in case the notation has to be modified to better represent the special moves like en passant and castling.

To finish we update the initial position to be the final one.

2. Actuation System

2.1. 2 Axes electromagnet

Our first approach to the actuation system was to move the pieces around the board by means of an electromagnet capable of moving with 2 degrees of freedom under the chess board. With this approach, we would be able to move all the pieces individually to any position on the 2 dimensional plane defined by the board.

We considered this system because of its simplicity when it comes to its fabrication and control by software, since it is essentially a 3D printer with one of its axes removed. Looking at the figure 18, you could imagine the system by removing the bed of the printer and substituting the extruder head with an electromagnet.

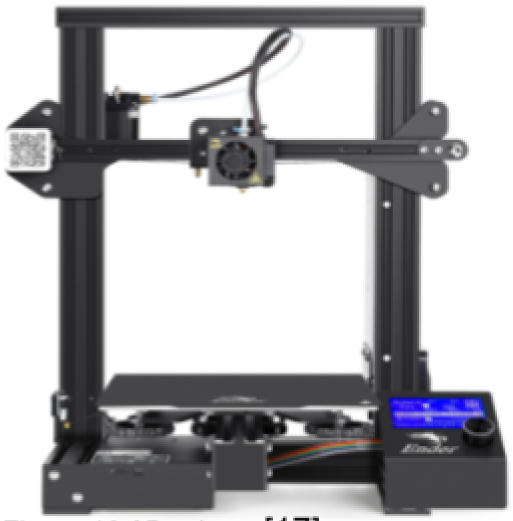


Figure 18 3D printer [17]

Although it is simple, this kind of system has drawbacks:

- The space needed to house the entire 3D printers frame is significant
- The weight would be too much, hurting portability and user experience
- When moving, the noise produced is quite loud
- We can not actuate more than 1 piece at a time
- The movement of the pieces is slow

For these drawbacks and the fact that there already exist commercial products that use the same approach [3] [18], we decided to first attempt other ways of actuating the pieces, and leave this method as a last resort in case we are not able to develop a more silent, compact and innovative solution.

Figure 19 Square Off chess board [3]

2.2. Printed Circuit Board Coils

The next way we considered to solve the problem of actuating the pieces is the use of planar coils to generate magnetic fields, by allowing current to flow through them, with which would be able to affect the pieces, either attracting them or repelling them, depending on the polarity of the magnetic field generated.

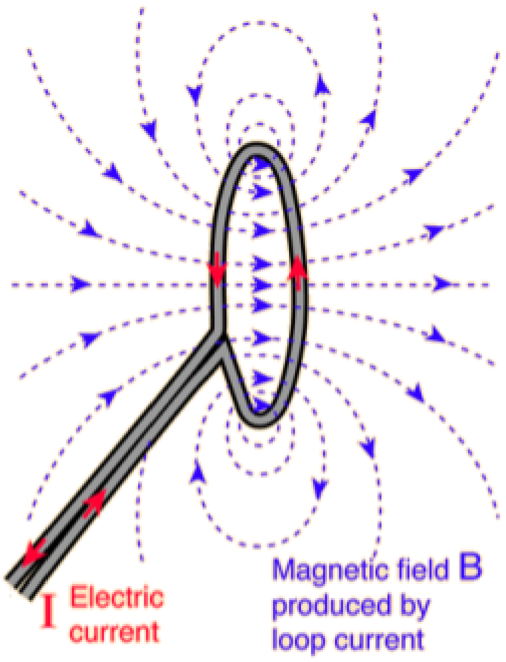


Figure 20 Magnetic field generated by a loop of wire [19]

With the use of PCB coils we can greatly reduce the size of the whole actuation system when compared to the 2 axes electromagnet actuation, this is due to the small thickness of the PCBs of just a few millimeters or even less than 1 millimeter (depending on the manufacturer [20] [21]) at an extra cost. This small size even allows us to stack PCBss on top of each other allowing for a stronger magnetic field and/or resolution.

This method of actuation also allows us to move multiple pieces at the same time as well as significantly reduce the amount of noise, since there are no moving parts like gears, belts or motors, the only thing that would move is the piece or pieces we desire.

All this advantages however do not come without any downside:

- Complexity in the design of the printed circuit boards
- High cost if the PCB is of large size
- The coils generate heat, we can think of them as resistors
- The magnetic field outside of the shape defined by the coil is weak

Although problematic, we are confident that the downsides can be resolved with a careful and calculated approach to the design and implementation of this kind of system of actuation of the chess pieces.

2.2.1. Initial proof of concept

After some searching on the internet we came across this video [22] where PCB coils are used to actuate magnets, fortunately they also included the gerber file of the PCB design [23]. We ordered the PCBs and after some soldering we got the below shown actuators.

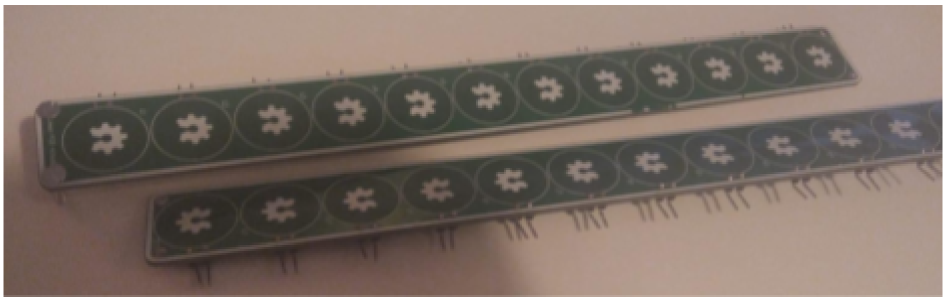


Figure 21 Initial proof of concept PCBs

We tried replicating the actuation of a 2cm in diameter magnet shown in the aforementioned video, but were unsuccessful. From our practical tests we came to the conclusion that the magnet should be partially over the coil we want to move it towards.

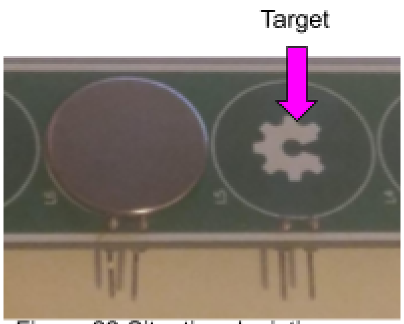


Figure 22 Situation depicting impossible actuation of magnet

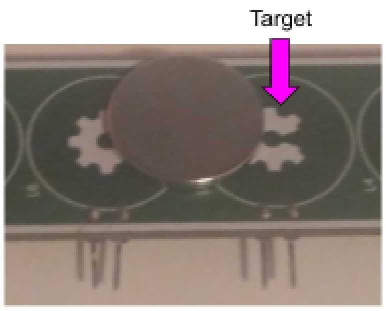


Figure 23 Situation depicting possible actuation of magnet

In figure 22 we would not be able to move the magnet towards the target coil since there is no overlap between the magnet and the coil. The reason being that the field produced by the coil is much weaker, outside the circle described by the coil than the inside. We could drive the coils at a higher voltage, to augment the strength of the field produced, but then we increase the heat they produce and risk overheating and damaging them.

In figure 23 the magnet would be easily able to move to its target since it would be affected by a greater magnetic field, since it is partially inside the circle described by the coil.

In order to overcome this limitation, we decided to try stacking the PCBs so that there is overlap between coils, like in the figures shown below:

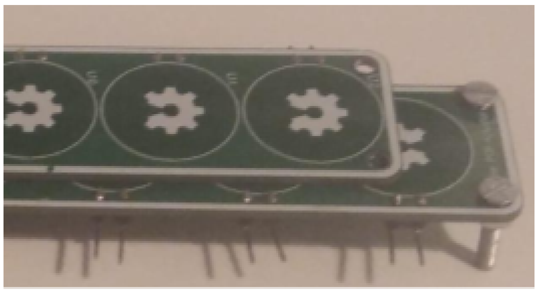


Figure 24 Stacked PCBs

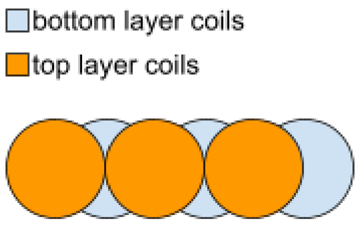


Figure 25 Depiction of the coil arrangement

With this arrangement we were easily able to move the magnet linearly along the track described by the PCBs. However, this arrangement only allows 1 degree of freedom, therefore we must design an arrangement that enables 2 degrees of freedom.

2.2.2. PCB designs

2.2.2.1. Initial design

We decided to test the behaviour of different sizes of magnets and coils. In order to achieve this, we had to design our own PCB with different coil diameters and number of PCB layers per coil. We decided to design 4 different types of coils:

- 2 centimeters in diameter and 4 layers
- 2 centimeters in diameter and 2 layers
- 1 centimeter in diameter and 4 layers
- 1 centimeter in diameter and 2 layers

The coils consist of 5 mil in width tracks, arranged in such a way so that they approximate a spiral, and the different layers of tracks are connected in series by means of vias, in order to avoid creating opposing magnetic fields between layers, see figures below.

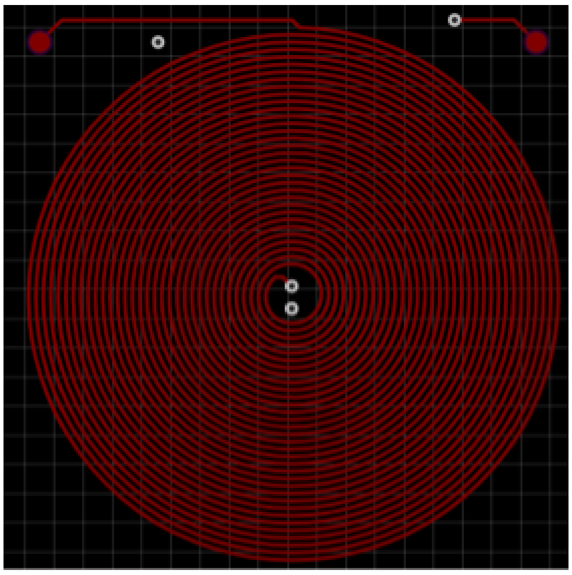


Figure 26 Top layer of a 4 layer PCB coil

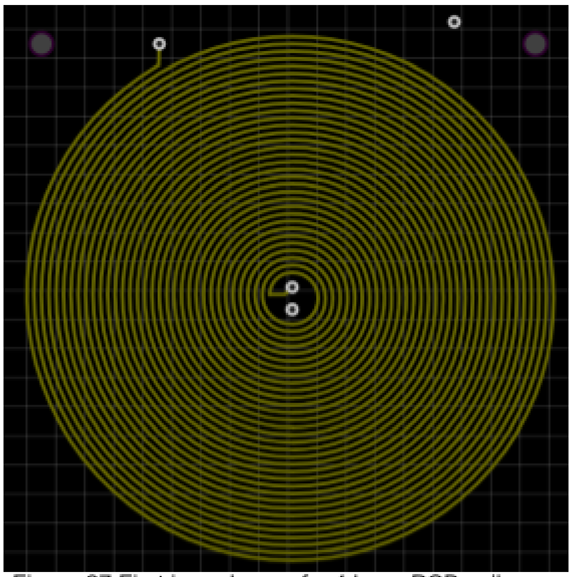


Figure 27 First inner layer of a 4 layer PCB coil

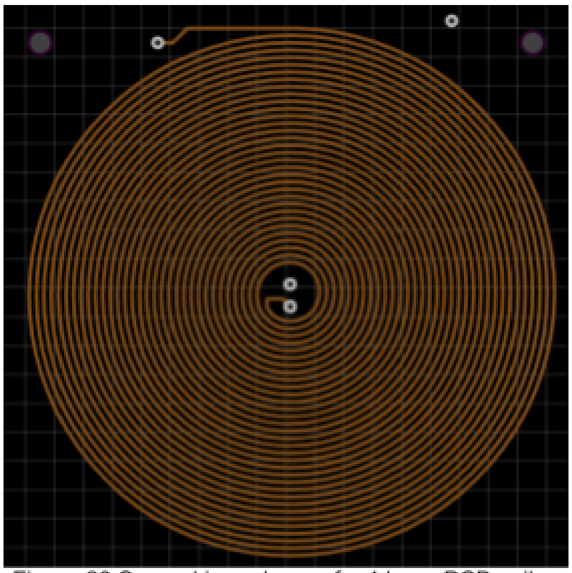


Figure 28 Second inner layer of a 4 layer PCB coil

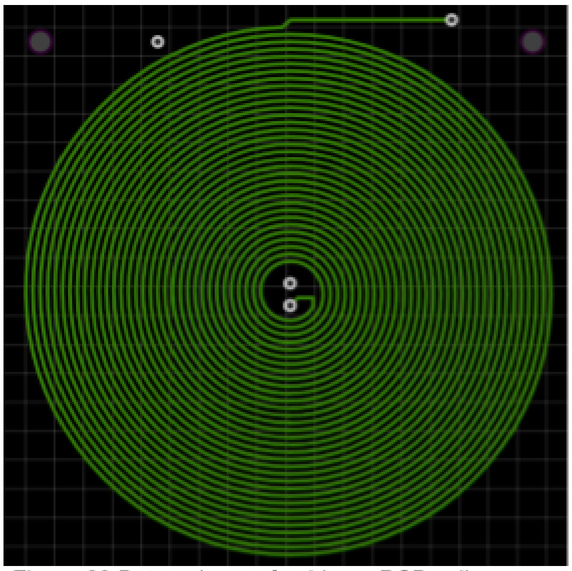


Figure 29 Bottom layer of a 4 layer PCB coil

The initial pcb design with the different types of coils can be found here:

https://gitfront.io/r/Sugarb0y/6d426b9c6a30af4b2aa1fd28a4547670a4b38d61/eChess/tree/Gerbers/

In figure 30 we can see the physical PCB.

When testing with different sizes of magnets, we quickly realized that the 2cm, 4 layered coils were producing the strongest magnetic field compared to the other coils driven at the same current. The coils were easily able to actuate a 2cm magnet with a wooden chess piece on top of it when in the situation depicted in figure 23, while still being able to actuate more weight.

The 2cm, 2 layered coils were also able to move a 2cm magnet with a piece on top of it, although not being able to move as much weight as the 4 layered coils. Still, their use is interesting since it would allow for overlap between coils of different layers of a 4 layer PCB, similarly to the arrangement of figure 25.

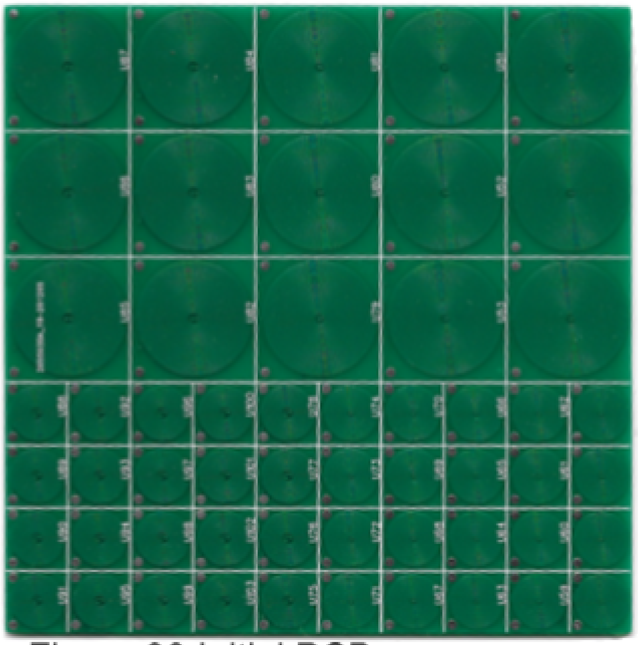


Figure 30 Initial PCB

When it comes to the 1cm in diameter coils, both the 2 and 4 layered ones, were barely able to move a 1cm in diameter magnet on their own, and were unable to move it with a piece on top of the magnet. They also generated a significant amount of heat, reaching even a temperature burning to the touch.

We also tried solving the issue of having to stack PCBs so that there is always overlap between a coil and a magnet by using a magnet with a diameter larger than that of the coils, however, this yielded negative results, as the magnets did not tend to get actuated until they were centered with respect to the coil, they just got actuated until they completely covered the target coil, meaning there would be not overlap between it and the next coil, preventing actuation. See figures below.

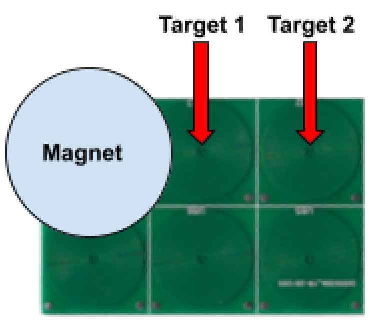


Figure 31 Initial position

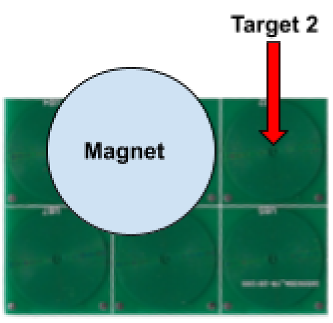


Figure 32 Final position

In figure 31 the Target 1 coil is able to actuate the magnet until it reaches the final position depicted in figure 32, where the target 2 coil is unable to actuate the magnet.

With the help of a teslameter, we were able to quantify the strength of the magnet field produced by each type of coil at different voltages. In the table below we can see the highest strength measured, which, as expected, was found at the center of the coil area. By taking measurements on different parts of a coil, we were able to see that the strength of the field next to the coil was very close to 0 T, and it progressively gets stronger the closer we measure to the center of the coil. We suspect the strength follows a gaussian curve, but can not accurately estimate it.

	2cm 2 layers			2cm 4 layers				1cm 2 layers		1cm 4 layers	
Voltage (V)	3	5	8.6	3	5	9	12	1.1	2.4	3	5
Amperage (mA)	355	540	1000	88	137	230	325	500	1000	332	476
Magnetic Field (mT)	3	5	8	1.5	2.3	3.7	5	2.5	5	3.2	5.1

Table 1 Magnetic field measurements of the PCB coils

These measurements are highly inaccurate, as when powered for prolonged periods of time, the coils heat up. This causes their resistance to start creeping up, therefore reducing the current flow through them, which produces a decrease in the magnetic field's strength.

We also measured the resistance of the coils, and they are as follows:

	2cm 2 layers	2cm 4 layers	1cm 2 layers	1cm 4 layers
Resistance (Ω)	7.5	31.2	1.7	7

Table 2 Resistance measurements of the PCB coils

2.2.2.2. Second design

We decided to tackle the problem, related to the need of stacking coils for the actuation of the magnet, by designing a new coil arrangement and PCB. Taking advantage of the fact that we previously discovered that a 2cm 2 layered coil is enough to actuate a magnet, we came with several possible PCB designs featuring overlapping coils.

In figure 33 we can see that there are different colored overlapping circles, each color represents a 2 layered coil, therefore we would need a PCB with a total of 4 layers in order to be able to print this design. We would need to print a PCB as big as the actuation surface, basically the size of the chess board, since we can not panelize this design, the reason being that, no matter where we decide to make a cut, we would have to cut through a coil.

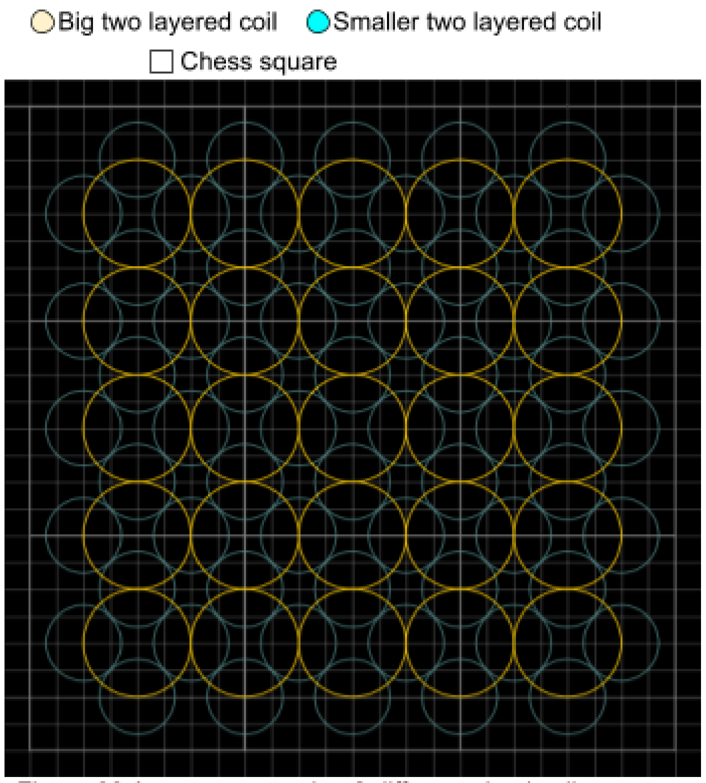


Figure 33 Arrangement using 2 different sized coils

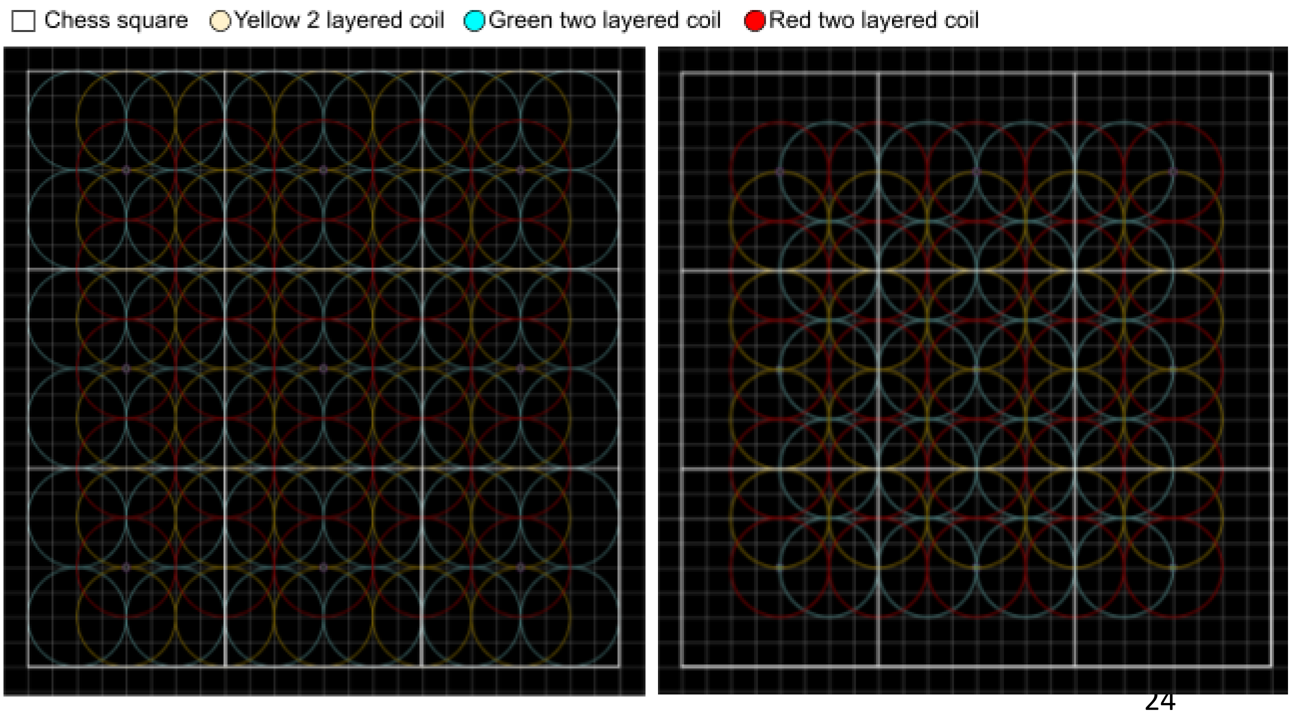


Figure 34 Arrangements using 3 overlapping coils

The arrangements of coils featured in figure 34 use a total of 6 layers per PCB, which would bring up the price of the PCB significantly, as 6 layered ones are way more expensive than 4 layered ones, and the design still does not allow for panelization to reduce the cost.

We decided to order the design in figure 33 since it would allow us to test, without having to print a PCB the size of the chess board, if we would be able to actuate the magnet with 2 degrees of freedom, and it was significantly cheaper than the 6 layered one.

The gerber file of the design can be found here. Below we can see the 2 out of the 4 PCB layers.

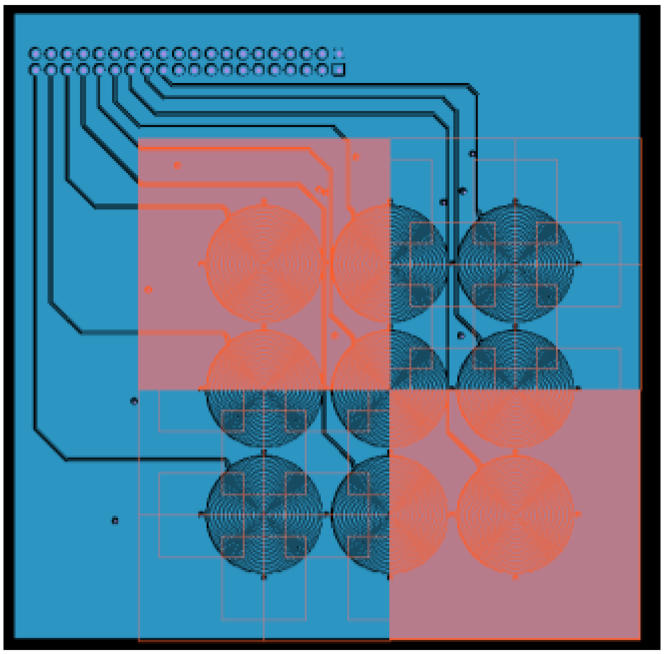


Figure 35 Second design PCB - top layer

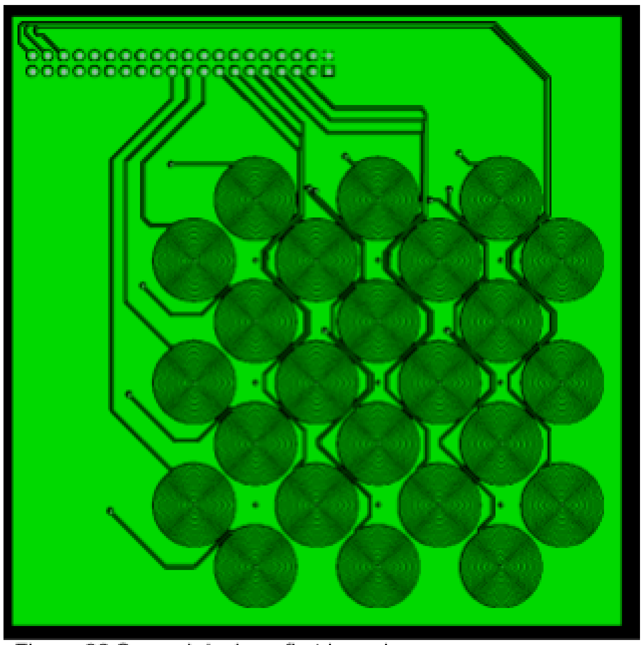


Figure 36 Second design - first inner layer

After testing, we were able to successfully actuate a magnet in 2 directions, up-down and left-right, like shown in this video:

https://youtu.be/ZYXnctin-2E

This proves that this arrangement of coils would be able to move a chess piece to and from any position on the board. However, we still think a modification to this design is necessary in order to allow for panelization to drive the cost of the PCB down.

2.2.2.3. Final design

After proving that our previous design is capable of actuating a magnet as we desired, we focused on making the design more affordable. In order to achieve this goal, we were forced to split the design into 2 separate boards that then would stack on top of eachother, thus allowing for panelization of the PCB. The bigger coils would be on one board, and the smaller ones on another one, this would also allow us to increase the layers of the coils, from 2 to 4, to compensate for the weaker magnetic field due to the increased distance from the magnet to the PCB.

To allow access for the connections to the coils while leaving a flat surface on which to actuate the magnets, the use of through-hole vias was necessary. That way it would be possible to make a connection to a coil from the top PCB through the bottom one. This poses an extra challenge in the design as every hole has to line up perfectly on both boards or the connection would be impossible.

When it comes to the bigger sized coils, they would be placed in the bottom of the 2 PCB stack, since the magnetic field is stronger than the smaller coils. The only thing worthy of comment of this PCB design are the aforementioned through-hole vias placed in columns between the coils.

As for the smaller coils several different designs were necessary for the panelization, since angled cuts had to be used and the different PCB designs had to fit like a puzzle, like depicted in figure 38.

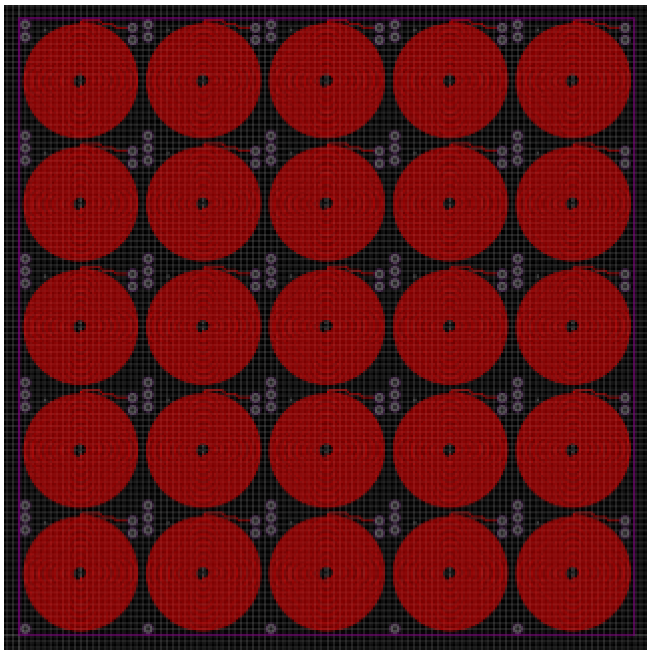


Figure 37 Bottom layer coils

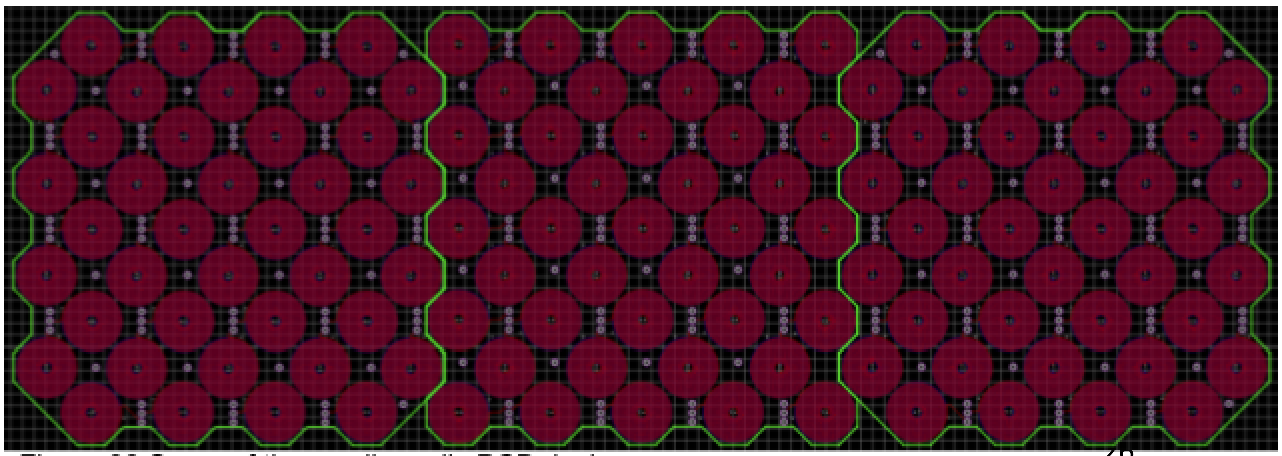


Figure 38 Some of the smaller coils PCB designs

The gerber files for this design can be found here:

https://gitfront.io/r/Sugarb0y/6d426b9c6a30af4b2aa1fd28a4547670a4b38d61/eChess/tree/Gerbers/final%20design/

In the figures below we can see the partially assembled PCB stack.

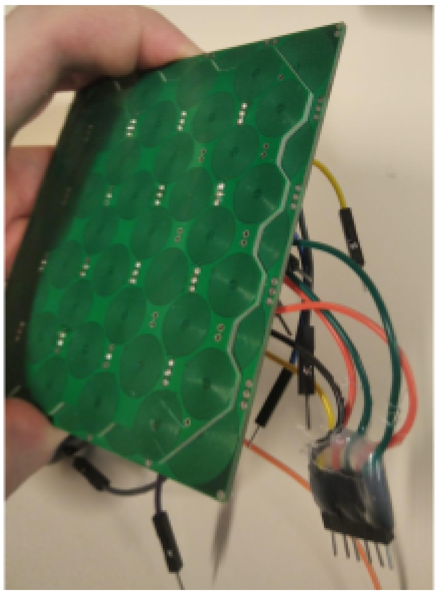


Figure 39 Side view of the 2 stacked and wired PCBs

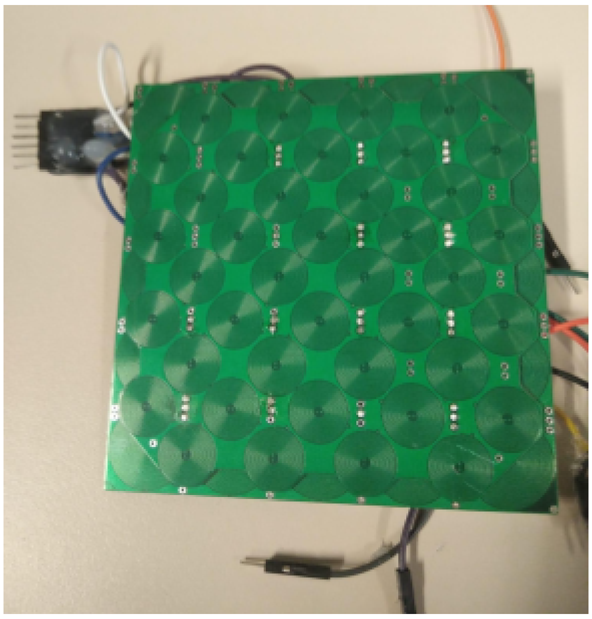


Figure 40 Two stacked and wired PCBs

Testing the final design manually, shown in this video:

https://youtu.be/8hdJ_XnOnzs,

we concluded that this specific coil arrangement is a viable option for a magnetic 2D linear actuator, so we proceeded to try to automate the driving of the coils. For this we made use of shift registers, like the SN74HC595N [24], and darlington arrays like the ULN2004A [25], to drive 12 different coils like in this video:

https://youtu.be/SkWbgyO4Gwg

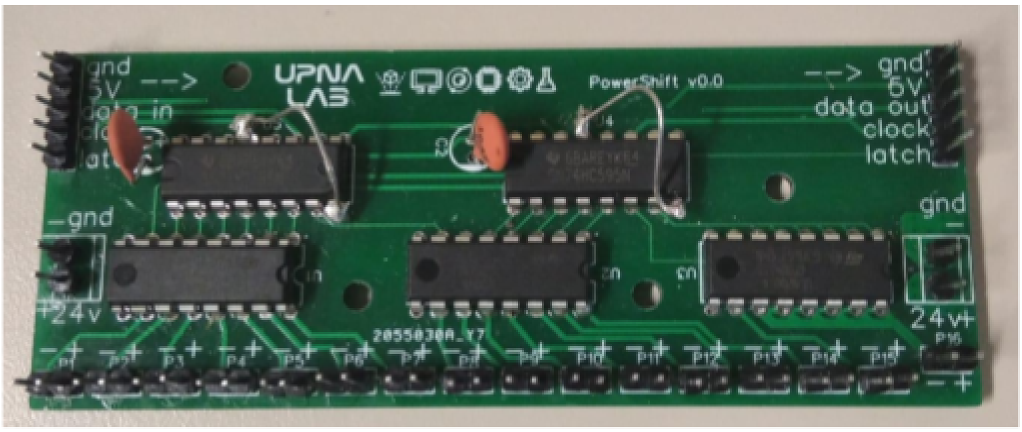


Figure 41 Driver circuit board

3. Software

To develop the software, we decided to use python as the language to code it in for its simplicity, and made use of Jupyter Notebooks [26] to increase productivity.

When it comes to implementation, we had to complete several steps:

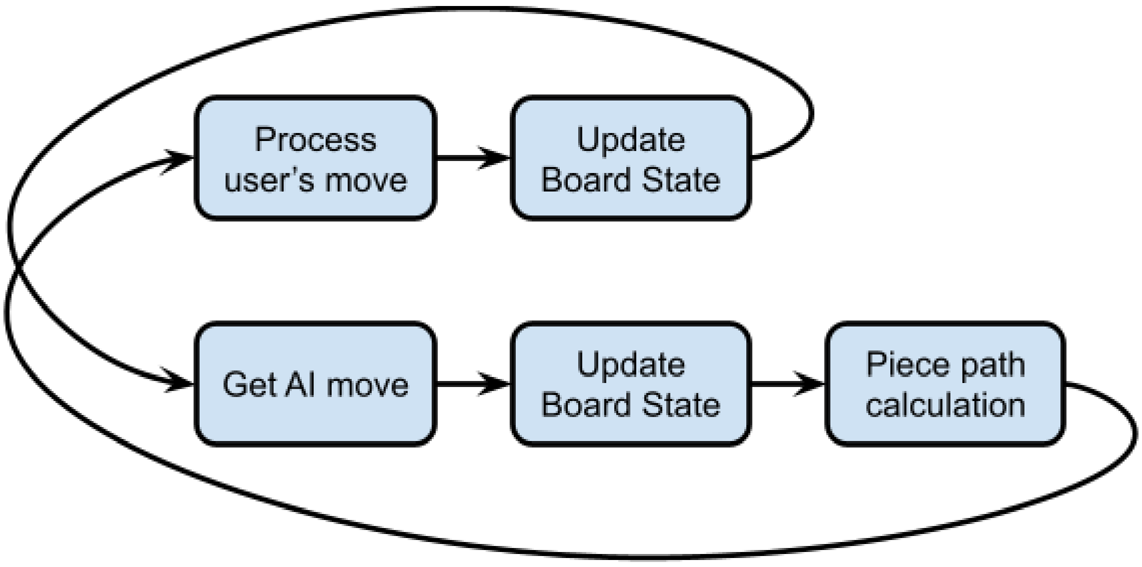


Figure 42 Diagram of software operations

- **Process user's move:**

  The microcontroller must send the movement done by the user, to the computer running the python code. For this, we decided to use the serial protocol to send strings that represent the move detected, for example the string "e2e4\n".

- **Update Board State:**

  Once we have received the move made by the AI or the player, we have to keep track of the board state so that we can, afterwards, send it to the AI to decide the best move. For this we make use of the python-chess library [27] to input the moves made, and then obtain the board state in Forsyth–Edwards Notation [28].

- **Get AI move:**

  To decide what move the board should make, we settled on using an online analysis board [29] where we can input the board state in FEN, and it will calculate the best move to be made. We automated this process by sending a POST request to the web page, containing the FEN and some authentication cookies, and then treated the obtained response to get the move in a string format, like “e2e4”.

- **Piece path calculation:**

  This has yet to be implemented but, it’s functionality should be to transform the string representation of the move returned by the AI, into which coils have to be activated and in what order, in order to move the chess piece from its initial to its final position. Since in most cases there is no direct path the piece could take, a path finding algorithm, like A* has to be used. This algorithm should be able to work with some restrictions, since it could be necessary to move other pieces before the one intended in order to make room for it to move, since a piece can not pass through the space left between 2 other pieces because the coils can not overcome the strong opposing magnetic fields of the magnets.

The jupyter notebook used to implement the code can be found here:

https://gitfront.io/r/Sugarb0y/6d426b9c6a30af4b2aa1fd28a4547670a4b38d61/eChess/blob/serial_com_arduino_python.ipynb

Future work

To complete the project we only have to implement the path finding algorithm, finish assembling and wiring the PCBs and integrate the actuation system with the software. The only problematic part could be the path finding algorithm, since no options have been explored yet, and if it has to be implemented from scratch is unknown.

There are however some improvements to the PCB design that could be made now that we know the coil arrangement works. The panelization aspect of the boards drastically decreases the cost of the project, which is great for the prototyping phase. However, this cost doesn't take into account all the labour required to assemble, solder and connect, all the pins, wires and cables needed to interconnect the different PCBs. We could remove almost all of the labour by just printing 2 full size PCBs, one for the big 2cm coils and the other one for the smaller coils, and if the view is from a production standpoint, it is the most logical option, since the price per unit of the PCBs decreases the more of them you order.

Adding wireless functionality to the project would be a very simple modification that would increase the usability of the board immensely, since it would remove the need of a cable connecting the board and the computer.

One of the most important aspects that needs work is the driving of the coils. Currently they are driven individually, meaning we would need a number of driver channels equal to the number of coils, which currently stands at 618. This obviously is impractical, therefore methods of reducing the number of channels should be explored in the future. Driving the coils by rows and columns, like in the driving method described in MAGHair [30], and similar to the way we power the sensor matrix, could be the best option, since it would reduce the number of channels by a factor of 4, the only inconvenience would be the need of a diode in series with each coil.

Another aspect that has not been explored is the fact that it should be quite simple to adapt the system to play different board games, not just chess. By changing the board on top of the coils, we can for example be able to play parchís, backgammon, connect 4 or tic tac toe. Some modifications might need to be made to the different systems to allow for this, like adding more sensors or adding more coils, to increase the play area or allow for more complex movements. Adding the ability to play other games also brings additional layers of difficulty when it comes to the software, since different game AIs have to be added. Even though overcoming these inconveniences might be hard and or time consuming, we think that being able to play other games besides chess would drastically increase the interest in this project in general and as a possible commercial product.

References

[1] DGT, “Electronic Boards > - Digital Game Technology,” [Online]. Available: http://www.digitalgametechnology.com/index.php/products/electronic-boards.

[2] MILLENNIUM, “MILLENNIUM | Munich chess computers for online & offline gaming.,” 17 November 2020. [Online]. Available: https://computerchess.com/en/.

[3] S. Off, “Square Off: World's Smartest Chessboard.,” [Online]. Available: https://squareoffnow.com/.

[4] u/candidate_master, “Reddit,” 26 February 2020. [Online]. Available: /r/HobbyDrama summarizes the Regium Debacle: chess - Reddit..

[5] “web.archive.org,” [Online]. Available: https://web.archive.org/web/20201117000203/https://regiumchess.com/crowdfunding/.

[6] “archive.is,” [Online]. Available: https://archive.is/TzG8y.

[7] A. Yang, 2 april 2020. [Online]. Available: https://www.youtube.com/watch?v=08_XtbatgW0.

[8] aa4cc, “YouTube,” 11 september 2012. [Online]. Available: Steering an iron ball along a circular path by shaping the magnetic .....

[9] aa4cc, “YouTube,” 7 september 2011. [Online]. Available: Planar manipulation with a flat magnet over an array of electromagnets.

[10] aa4cc, “Youtube,” 10 november 2011. [Online]. Available: https://www.youtube.com/watch?v=5fpp5iCPzd8.

[11] C. Bugeja, “Youtube,” [Online]. Available: https://www.youtube.com/c/CarlBugeja.

[12] R. Electronics, “MDCG-4 15.3mm Sub-miniature Reed Switch - Littelfuse,” [Online]. Available: https://docs.rs-online.com/4b0b/0900766b81498d7c.pdf.

[13] T. Instruments, “DRV5053 Analog-Bipolar Hall Effect Sensor datasheet - Texas ...,” [Online]. Available: https://www.ti.com/lit/gpn/drv5053.

[14] “En passant - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/En_passant.

[15] “Castling - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Castling.

[16] “Universal Chess Interface - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Universal_Chess_Interface.

[17] “Creality Ender-3 Pro FDM 3D Printer,” [Online]. Available: https://www.creality.com/goods-detail/ender-3-pro-3d-printer.

[18] T. F. T. Internet, “What's inside an automatic chess board? - YouTube,” 13 january 2019. [Online]. Available: https://youtu.be/j7Iq3HkebhM?t=229.

[19] “Magnetic Field of a Current Loop - Hyperphysics,” [Online]. Available: http://hyperphysics.phy-astr.gsu.edu/hbase/magnetic/curloo.html.

[20] JLCPCB, “PCB capabilities & PCB production specification - JLCPCB,” [Online]. Available: https://jlcpcb.com/capabilities/Capabilities.

[21] PCBWay, “PCB Capabilities - Custom PCB Prototype the Easy Way - PCBWay,” [Online]. Available: https://www.pcbway.com/capabilities.html.

[22] C. Bugeja, “Actuating Magnets with PCBs - YouTube,” 11 june 2018. [Online]. Available: https://www.youtube.com/watch?v=LudKMv7M13Q.

[23] C. Bugeja, “Linear PCB Motor | Hackaday.io,” 6 may 2018. [Online]. Available: https://hackaday.io/project/158017-linear-pcb-motor.

[24] T. Instruments, “Datasheet SNx4HC595 - Texas Instruments,” [Online]. Available: https://www.ti.com/lit/gpn/sn74hc595.

[25] STMicroelectronics, “Datasheet uln2001 - STMicroelectronics,” [Online]. Available: https://www.st.com/resource/en/datasheet/uln2001.pdf.

[26] “Project Jupyter | Home,” [Online]. Available: https://jupyter.org/.

[27] “python-chess: a chess library for Python — python ... - Read the Docs,” [Online]. Available: https://python-chess.readthedocs.io/.

[28] “Forsyth–Edwards Notation - Wikipedia,” [Online]. Available: https://en.wikipedia.org/wiki/Forsyth%E2%80%93Edwards_Notation.

[29] “Chesshub.com: Chess Analysis Board and Move Calculator,” [Online]. Available: https://chesshub.com/.

[30] Roger Boldu, Mevan Wijewardena, Haimo Zhang, and Suranga Nanayakkara. 2020. MAGHair: A Wearable System to Create Unique Tactile Feedback by Stimulating Only the Body Hair. In 22nd International Conference on Human-Computer Interaction with Mobile Devices and Services (MobileHCI '20). Association for Computing Machinery, New York, NY, USA, Article 16, 1–10. DOI:https://doi.org/10.1145/3379503.3403545