Contents

[1 Aim 1](#_Toc531019243)

[2 Game types 1](#_Toc531019244)

[3 Design Considerations 1](#_Toc531019245)

[3.1 Piece detection 1](#_Toc531019246)

[3.1.1 Standard configuration 1](#_Toc531019247)

[3.1.2 Non-standard configuration 2](#_Toc531019248)

[3.2 Piece movement 3](#_Toc531019249)

[3.2.1 Electromagnet 3](#_Toc531019250)

[3.2.2 IR Sensor 3](#_Toc531019251)

[3.2.3 Robotic apparatus 3](#_Toc531019252)

[3.2.4 Obstacle detection 3](#_Toc531019253)

[3.3 Board composition 4](#_Toc531019254)

[4 Artificial intelligence 4](#_Toc531019255)

[5 Power supplies 4](#_Toc531019256)

[6 Minimum viable product 4](#_Toc531019257)

[7 Limitations 5](#_Toc531019258)

# Aim

The aim of this project is to build a chess application whereby the chess pieces will move themselves automatically.

Pieces can be moved either with hands or via verbal commands.

# Game types

This project will service three game types

1. 1 human vs 1 human
2. 1 human vs 1 AI
3. 1 AI vs 1 AI

# Design Considerations

A raspberry pi will be the central device controlling and managing the various peripherals in this project.

## Piece detection

### Standard configuration

This section assumes that the chess game starts in the standard configuration.

In the case where verbal commands are the only ones being given, the application doesn’t need to detect the pieces on the board, it will have a complete picture of where all pieces are on the board as all movements are conveyed to the application.

In the case where a player is moving pieces with his hand, the application needs to recognise what move this corresponded to. There are a few ways to do this. The options are outlined in the table below.

Table 1: Comparison of piece detection methods

|  |  |  |
| --- | --- | --- |
|  | Pressure Sensor Array | Pi Camera module |
| Supplier | Alibaba Chinese supplier | Raspberry pi foundation |
| API | Unclear, vendor hasn’t provided details | Well known python libraries |
| Cost (AUD) | 300 | 30 |
| Mode of detection | Compare map of pressure hotspots before and after move | Take pictures before and after move and compare them for differences, the two spots with the greatest differences will correspond to the move. |
| Granularity | Extremely precise | Variable with lighting conditions |
| Durability | Wear and tear over time may cause detection issues, build quality is also suspect due to the origin of the piece | Not coming in contact with any pieces, as a result it should last a long time. |

From the above comparison it is clear that the raspberry pi camera module provides a low cost, more durable way to detect pieces.

### Non-standard configuration

This section assumes that the chess game starts in a random configuration.

This poses a few challenges as now the application has no knowledge about any of the pieces on the board, it needs to obtain information about all the pieces on the board, a few solutions are outlined below:

1. The application performs no validation of any piece moves, it blindly moves pieces to the requested positions.
2. The user manually inputs all the pieces on the board via an API
3. NFC chips are embedded into each piece. The application traverses the entire board, reads the NFC chip and constructs an idea of the board.

A comparison of these approaches is performed below:

Table 2: Comparison of options for starting from a non-standard configuration

|  |  |  |  |
| --- | --- | --- | --- |
|  | No validation | Manual user input | NFC Chips |
| User friendliness | Just place the boards | Cumbersome and error prone | Easy |
| Timeliness | Easy |  |  |
| Materials cost | $0 | $0 | $185 total ($5 per tag and $25 for NFC reader) |
| Programming required | Minor amount, code paths are required that perform no validation | None, existing infrastructure can be leveraged | Interaction |
| Other considerations |  |  | NFC chips would need to be embedded into each piece, they would need to be secured. This is an additional cost |

## Piece movement

In order to move the chess pieces, ferromagnetic material is attached to the underside of each chess piece. An electromagnet will then be used to magnetically attract those pieces and move them to their desired positions. To prevent jittering the very middle of the piece should be grasped.

### Electromagnet

A grove electromagnet was chosen for use due to its low cost, this electromagnet connects to a GrovePi+ board which integrates directly with the raspberry Pi. The GrovePi+ facilitates the use of many sensors simultaneously.

### IR Sensor

An IR sensor can be used to detect where a piece is on the board square, a Sharp GP2Y0A41SK0F IR sensor provides a cost-effective way to do this. If this sensor is rotated a series of data points can be obtained to provide a better estimate of the centre of the piece.

### Robotic apparatus

Off the shelf solutions for the robotic apparatus have been evaluated. A comparison below shows the differences between two low-cost alternatives:

Table 3: Comparison of low-cost robotic arms

|  |  |  |
| --- | --- | --- |
|  | EleksLaser-A3 pro 2.5W | Makeblock XY stage |
| Cost (AUD) | $315 | $250 |
| Working area | 30x40cm | 31x39cm |
| Electrical requirements | 12V, 5A | 12V, 3A |
| Weight | 3.4kg | 3.17kg |
| Reviews | Good reviews about use | Complaints about jittering, difficulty of construction and outdated documentation |

The EleksLaser-A3 pro is being selected largely due to its reviews.

The EleksLaser can be programmed with GRBL software which has python compatible libraries, this permits seamless use with the raspberry pi

### Obstacle detection

When pieces are being moved optimal paths need to be selected for efficiency and to avoid obstacles. The A\* algorithm has high performance and accuracy.

## Board composition

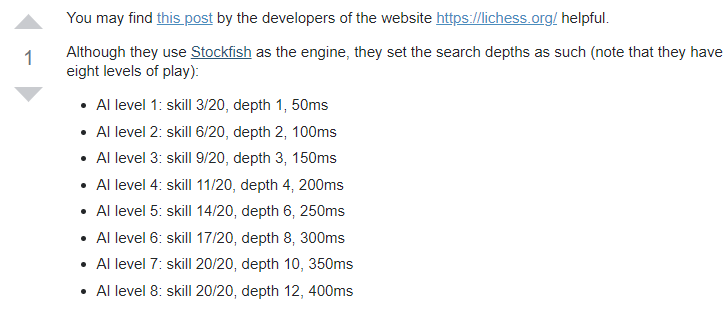
Due to the need to use a camera to detect board positions and an IR sensor to detect centred pieces, a need exists to use a board which facilitates the operation of these sensors. An overview of the selection criteria is shown below

|  |  |  |  |
| --- | --- | --- | --- |
|  | Finely threaded mesh | Wooden chessboard | Glass |
| Aesthetics | Cheapish look | High quality perception | Mid quality perception |
| Detecting pieces and moves | IR sensor and camera | Pressure sensor array (cost high) | IR sensor and camera |
| Cost | Low | Medium-Very high | Medium |

Potential issues with pressure sensors are highlighted in Table 1.

# Artificial intelligence

The AI selected must be able to play at 10 difficulty levels. Consensus online is to use the open-source Snapfish engine. The stackoverflow article located at <https://chess.stackexchange.com/questions/14770/gull-engine-which-depth-value-to-set-6-difficulty-levels/17698> provides the following pertinent information.



# Power supplies

The application should be rechargeable with one plug powering all the necessary devices.

A device will need to be programmed to supply 12V to the motor system and 5V to the raspberry pi

# Minimum viable product

The minimum viable product needs to fulfil the following criteria

* Only operate in standard mode
* Allow a human player to either use their hand or issue audio commands to move pieces.
* Pieces will move without touching each other
* Permit the selection of 10 difficulty levels.
* Recognise conditions such as check/checkmate
* Have a single electrical plug that will be rechargeable and supply 12V to the motor system and 5V to the raspberry pi.
* Handle some misregistered moves

The cost outline of this is outlined below

|  |  |
| --- | --- |
|  | Cost (AUD) |
| Raspberry pi | 41 |
| GrovePi+ | 42 |
| Raspberry Pi Camera Module v2 | 40 |
| Sharp GP2Y0A41SK0F IR sensor | 5 |
| EleksLaser-A3 pro 2.5W | 315 |
| Grove electromagnet | 15 |
| 12V supply | 31 |
| 10000 mAh supply | 30 |
|  |  |
| **Total:** | 519 |

# Limitations

The following cases are actively not being handled

* Resetting of the board