Wireless Chess Robotic Arm

WCRA-AI

Authors

Ahmed Sabry Ali Lilah

Alyaa Mosaad Ahmed Sherif

Moataz Mohammed Ibrahim Ali

Mohamed Ashraf Hassanen Ali

Mohamed Moataz Moustafa Hassan

Mostfa Hisham Abd Elgawad

Supervisor

Dr.Mohammed El-Sayed Ali Hammad

1. Table of Content

2. Abstract

Wireless Chess Robotic Arm (WCRAI for short) is a robotic arm capable of playing chess, it can be controlled over the network as well this is where the word wireless came from. The purpose of this project is to create a robot using modern technologies like computer vision, convolutional neural networks, and the Internet of things that's capable of playing chess. There is an increasing demand for smart robots that can make their own decisions in a changing environment and this idea is an interesting application of the concept. The arm was designed in a way that can generalize the usage of the arm later in pick-and-place operations, Servo motors controlled by an Arduino carried out the movement of the arm. movement limiting switches were added to protect the servo motors also attached to Arduino. A camera was placed above the chessboard and attached to a Raspberry pi, and numerous programs and functions were created in order to process and convert the image into notation describing the position of the chess game, depending on the mode (single player, multiplayer) notation is sent either to the chess engine (powered by NegaMax algorithm also running on the Pi) which decides which move to play or to the server to transmit it to the other player robot arm (a web page in current conditions).

2.1. Keywords

chess, image processing, computer vision, artificial intelligence, neural network, embedded system, kinematics, Internet of things(IoT), webpage, and decision making.

3. Introduction

“Chess is the Drosophila of artificial intelligence.” Alexander Kronrod

Chess is a game of unlimited beauty. Many aspects of human intelligence and cognitive skills appear in a game of chess. Decision-making, pattern recognition, algebraic and geometric thinking, problem-solving, critical thinking, and many more are used in a game of chess. Our goals in this project are to teach a computer how to play chess and embed this knowledge into a robot arm that can interact with its environment. Controlling this arm by a web page for two players mode will be a great bonus. This is a very interesting application for modern technologies like Ai, IoT, and web applications. Many other technologies are also used.

Web Page :

Web development is divided into the front and back end part. The Front End part consists of a web page that contains a user-friendly interface to take user inputs to move the robotic arm. The back-end part connects the page to MongoDB, which enables the page to be accessible to the Internet and send and receive data, write and read data from and to the MongoDB.

Arduino-Based Embedded System :

Arduino Uno kit is a microcontroller that controls the movement of the motors for the robotic arm, the screen, the control of the keypad, and the sending and receiving of data from and to the Raspberry Pi. All connections are explained in detail in the Embedded System part.

The robotic arm :

The robotic arm is an arm with three ranges of motion: the X-axis, the Y-axis, and the Z-axis. In addition to a box that contains the electronics, the power supply, and the connections between the electronic parts, motors, and the camera. The mechanical design of the arm is explained in detail in the mechanical design part.

Raspberry Pi :

The Raspberry Pi is the brain of the project as a whole because through it we send and receive data from the web page. It also performs the process of image processing to discover the chess board and whether there has been a change or the player has played his move. It also sends and receives data and commands to the Arduino. Raspberry Pi contains a chess engine that is responsible for choosing the best move based on the current state of the chess board. The camera is used to take a picture of the chess board, and each picture is compared with the one before it to determine if there has been a change in the state of the board.

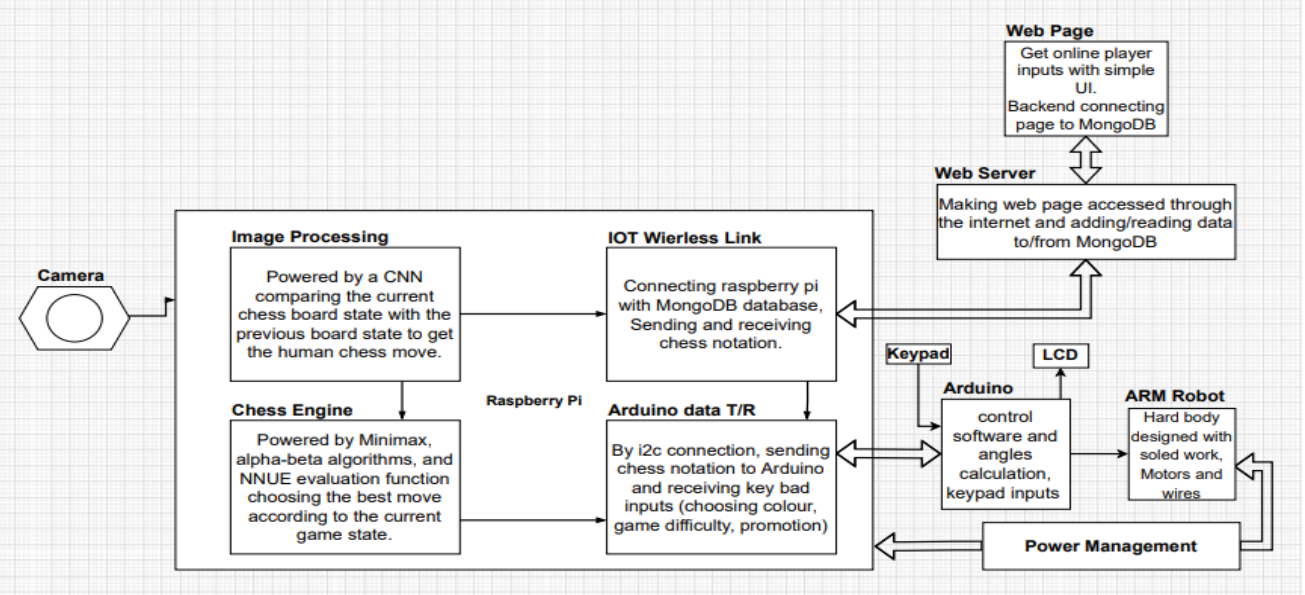


Figure (1) project block diagram

4. Mechanical Design

4.1. CAD Stage

The process started by designing the robotic arm on Solidworks according to the dimensions of the chessboard. This part is called CAD (computer-aided drawing). started drawing for the parts needed to install the motors according to the specifications and dimensions of the available motors in the market and suitable for the function of the robotic arm.

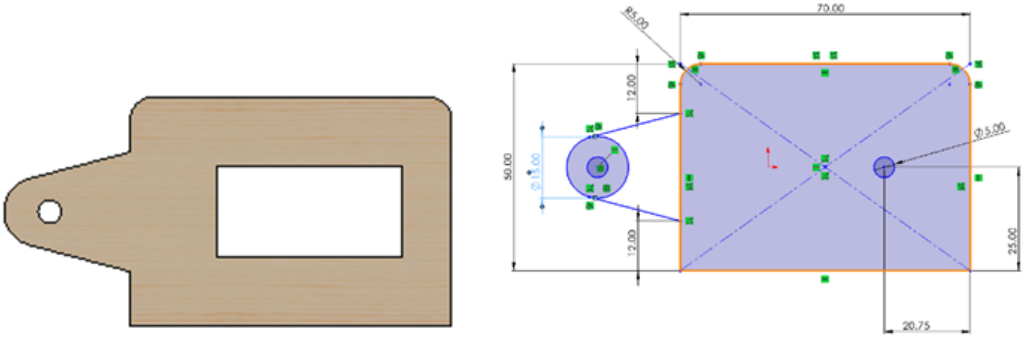


Figure (2) Base Shaft Anchor

A front pillar was made facing the motor responsible for the movement in the direction of the X-axis to distribute the load regularly. The shafts are designed to transform the movement from rotation to displacement according to the sum of the lengths of the pieces to cover the length of the chessboard and away from the robotic arm.

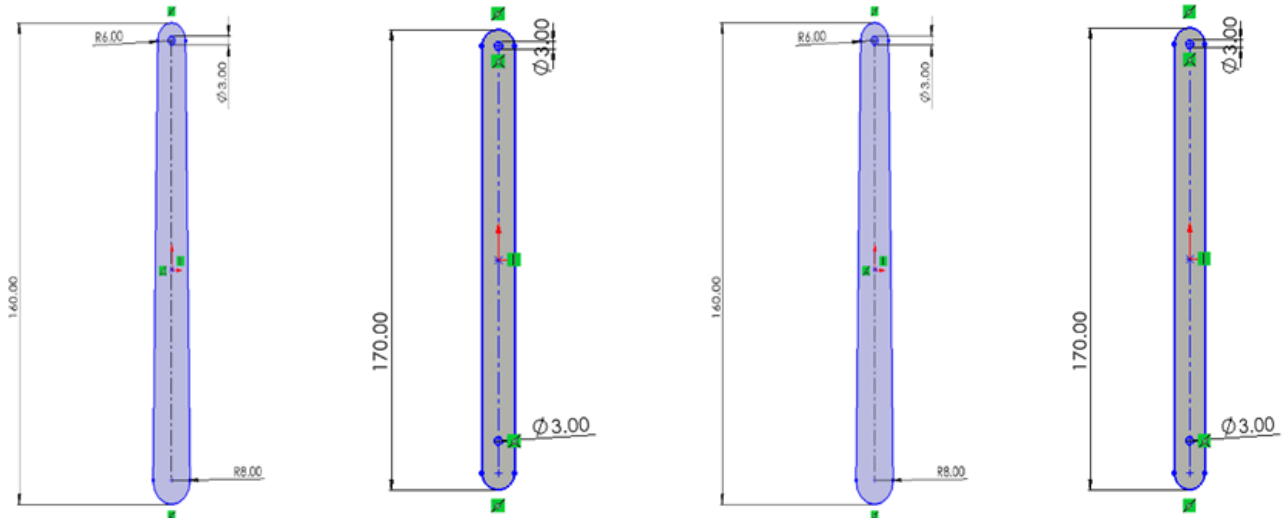


Figure (3) four main shafts

The piece responsible for installing the motor that is responsible for the movement was drawn in the direction of the Z-axis with the addition of some aesthetic shapes as this piece connects the field of movement in the x-axis and z-axis.

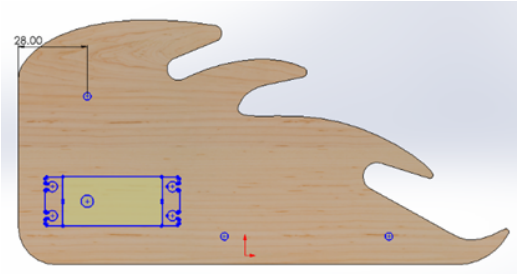


Figure (4) z-axis motor installer.

The grabber part was designed in addition to some aesthetic shapes of the horse's head shape.

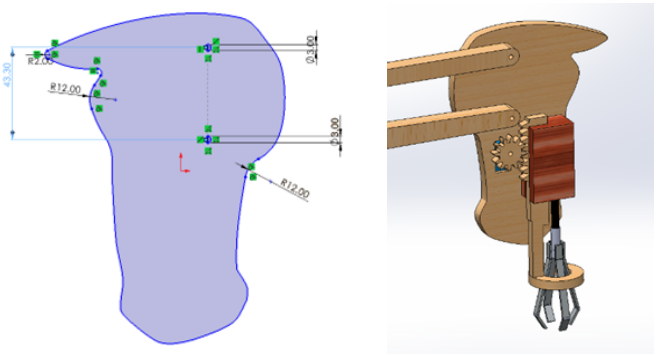


Figure (5) Grabber assembly.

The base that holds the entire robotic arm was designed taking into account the other parts such as the space of the engine mounts and the appropriate diameter of the base with the addition of two ducts to install the engine mounts.

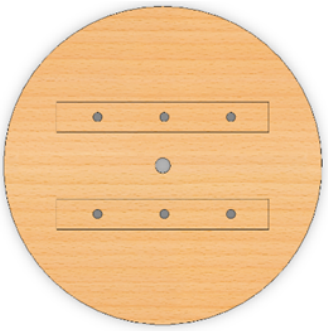


Figure (6) robotic-arm base

The base connecting rod with the motor for rotating in the direction of the y axis was manufactured on the lathe machine according to the diameter of the ball bearing available in the market, which was designed according to its dimensions.

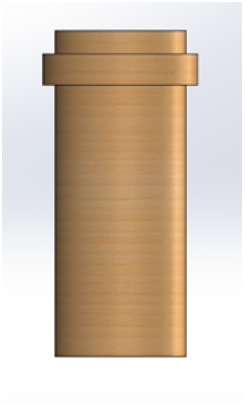


Figure (7) Base connecting rod to the base servo motor

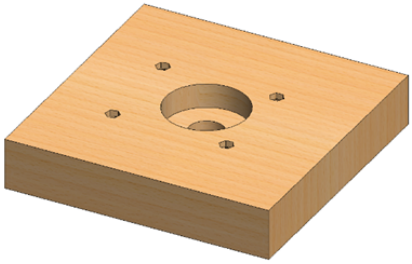


Figure (8) Ball bearing mount

Figure (9) Base motor holder.

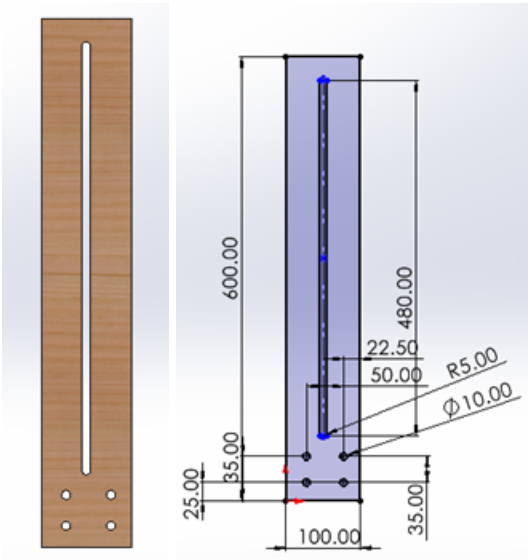


Figure (10) Camera slider

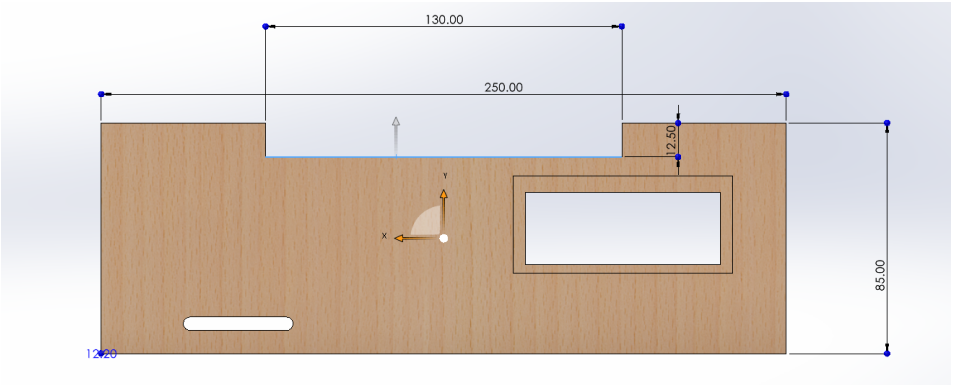


Figure (11) Front plate.

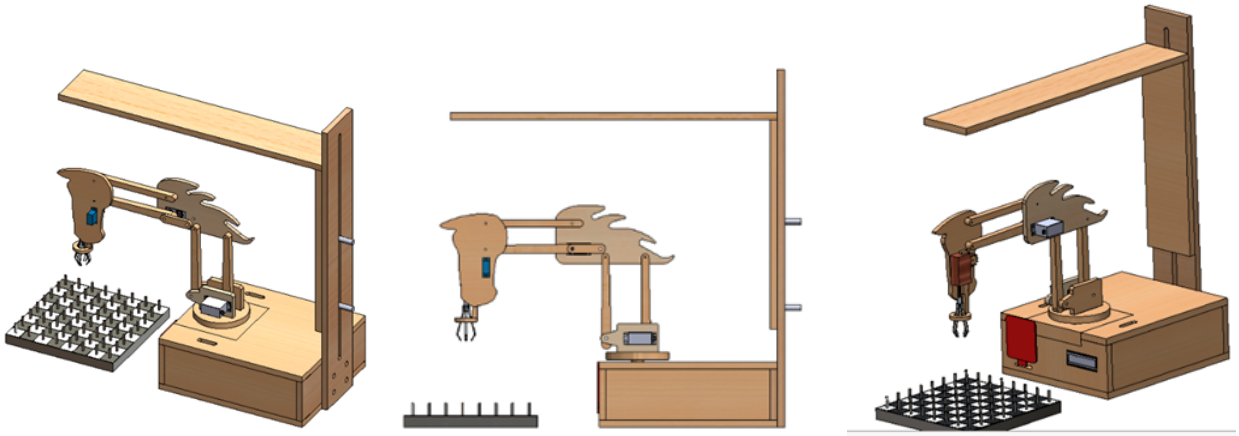


Figure (12) Complete assembly of the arm.

4.1.1. Axis and DOFs

As shown in the figures our robot is a 3-axis robot arm. One motor controls the angle of the base and two for the two arm anchor points as shown in figure (13)

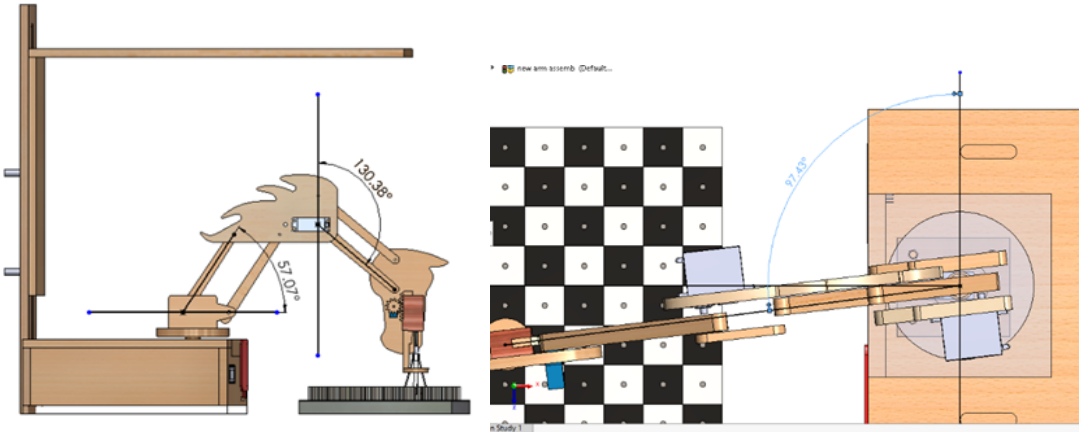


Figure (14)

The MCAD program is also used to verify the measurement of angles resulting from the equations of Inverse Kinematics. Then calculate the distance between the middle of a square and the middle of another square and use that distance to find the coordinates of any other square.

4.1.2. Range Of Motion.

Fixing our origin point to be at the bottom of the robot base center. And defining the X-axis to be in the direction normal to the robot front plate, the Y-axis normal to the X-axis while both are in the ground plane, and the Z-axis in the direction normal to the ground plane. We can then calculate our desired range of motion to be as follows.

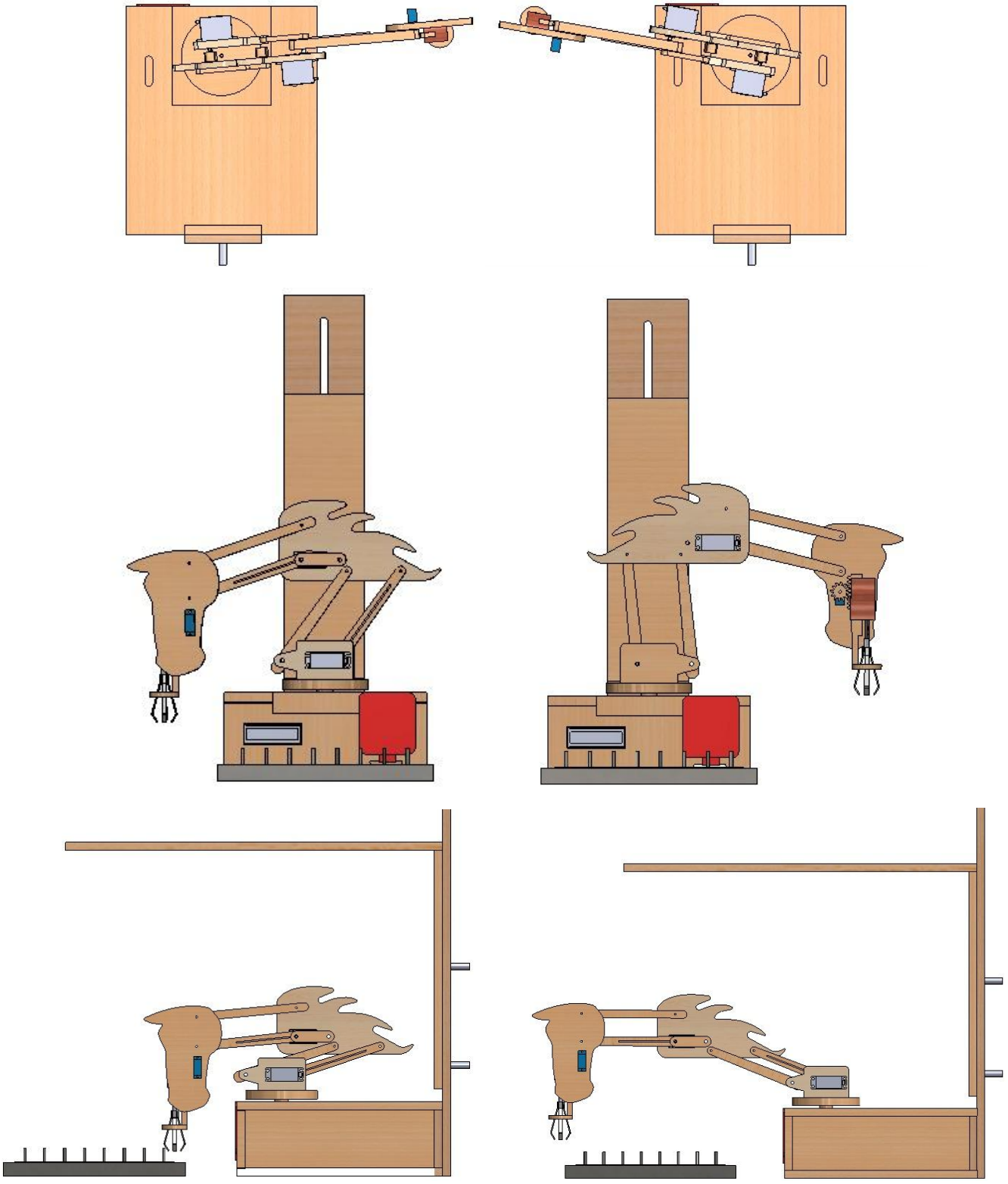


Figure (14) a, b, and c shows the robot's range of motion

4.2. Fabrication Stage

4.2.1. Material Choice

4.2.1.1. What are Servo motors

A servo motor is a rotary actuator that allows for control of the angular position. It consists of a suitable motor coupled to a sensor for position feedback.

Figure (15) Servo motors used for movement.

4.2.1.2. Body Material

It has been decided to use wood as the base material for manufacturing due to it being lightweight, low cost, and easy to handle.

4.2.2. Fabrication Process

After completing the drawing of all the necessary parts for the movement and the box containing the electronic parts. We sent the drawings to the CNC workshop. Each part was cut according to the specified dimensions. This part is called CAM computer-aided manufacturing. The parts are designed according to the specifications of servo motors, screws, and ball bearings available on the market.

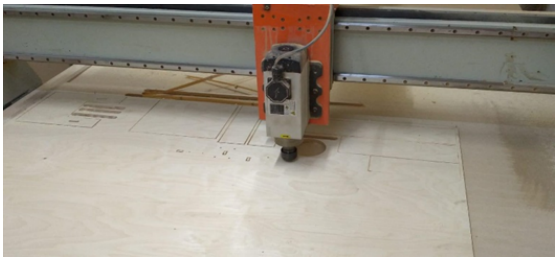


Figure (16) The parts while they are being cut.

Figure (17) The parts after they have been cut.

After the cutting process was done, the assembly was carried out using some types of screws, nuts, and some types of glue.

5. Inverse Kinematics

Inverse kinematics is the process by which we extract an equation that is used to determine the angles of the robotic arm joints if we give it the final position. Inverse kinematics is tied closely with the field of study of robotics and for complicated systems, it can be quite hard to figure out the control system. Lucky our robot isn't that complicated. And also we opted for an automated approach in order to solve this problem. By using the widely known and used MatLab package. We opted to use the Symbolic solver included in the software. And adhering to the principle that goes "don't reinvent the wheel". We got to throw the problem at it and left the symbolic solver to figure it out.

5.1. Symbolic Solver

A symbolic solver is a software package that is guided by the enormous mathematical wisdom and toolset such as equation refactoring, reducing the equations, and applying well-known equation solving patterns that mathematicians use to try to solve a problem given that the problem is properly modeled in a form of a mathematical equation. And a set of constraints that set limits on the space of available solutions, And a set of optimization parameters that allows the symbolic solver to approximate and simplify the solution and finally output the result in a symbolic form of the input Degrees of Freedom (DoF).

5.2. Model Equation

Before we try to find the quotations for the inverse kinematics problem we must first model our robot in the form of a mathematical equation.
The arm robot is a three-axis one with spherical-like DOFs. which means that there will be three variables for the three angles we need to control.
We also need to be able to use this robot over the board in a cartesian manner. The equations turn out to be the following.

Th1 = atan(Y/Z)
a1 + (r1 * sin(th2)) + a2 + (r2* cos(th3)) - a3 == Z
(-b1) + (r1 *cos(th2)) + b2 + (r2*sin(th3)) + b3 == R

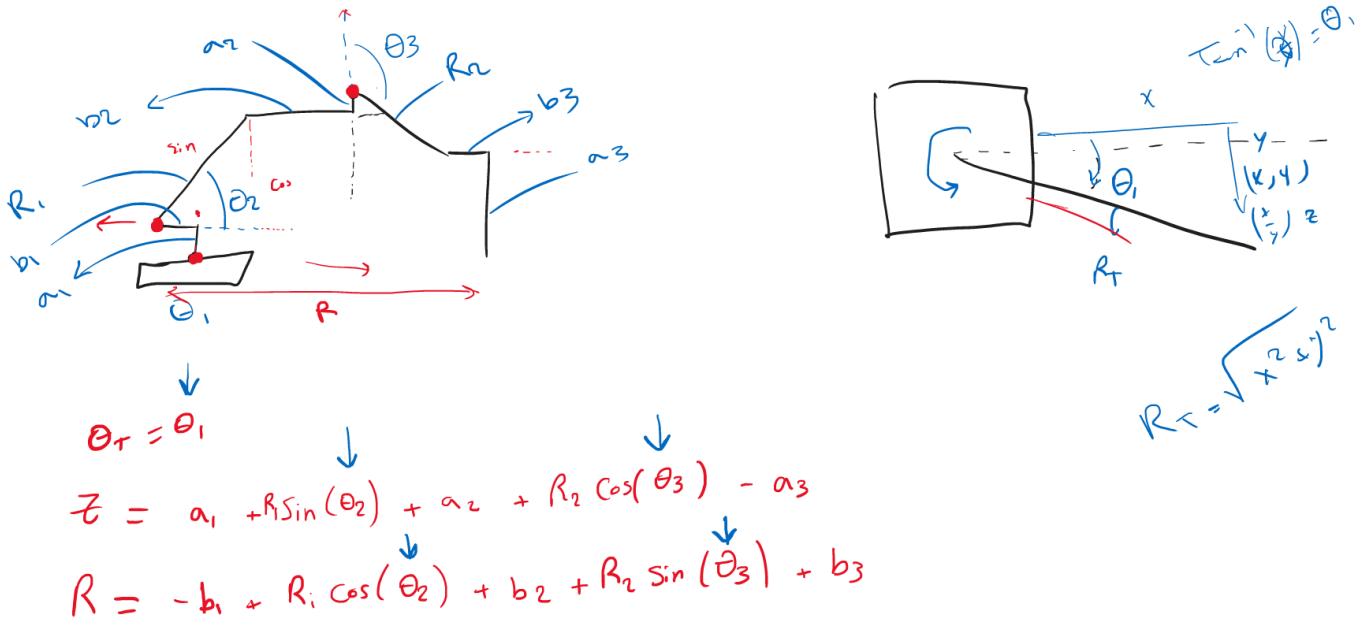


Figure (18) shows the extraction of equations from the robot body

Where:
X,Y, and Z are the cartesian coordinates.

a1, a2, and a3 are static shifts in the Z-axis due to the robot construction.
R is a new virtual axis that represents the length of the normal vector between the base of the robot and the robot tip in the which lies in the (x,y) plane, the |R| can be calculated knowing x,y by the following equation |R| = sqrt(x^2+y^2).
b1, b2, and b3 are static shifts in the R-axis due to the robot construction.
th1, th2, and th3 are the motor angles.

Then the equations are written into MatLab in a symbolic form.

```
eqn1 = a1 + (r1 * sin(th2)) + a2 + (r2* cos(th3)) - a3 == Z;
eqn2 = (-b1) + (r1 *cos(th2)) + b2 + (r2*sin(th3)) + b3 == R;
eqns = [eqn1, eqn2];
vars = [th2 th3];
```

5.3. Constraints

In order to solve the inverse kinematics problem, we must give it the range of movement of the robot. Such range of motion puts constraints on the inverse kinematics equations. In our case, the constraints are simple limits on the X, Y, and Z axes and the solution must be real.

5.4. Equation

After running the symbolic solver on the equations on Matlab the resulting equations turned out as follows.
th2 = -2*atan((544000000*Z + (- 4000000000000000000*R^4 + 2230560000000000000*R^3 - 8000000000000000000*R^2*Z^2 + 1564800000000000000*R^2*Z - 148712741600000000*R^2 + 2230560000000000000*R*Z^2 - 43629753600000000*R*Z - 45238441278960000*R - 4000000000000000000*Z^4 + 1564800000000000000*Z^3 + 160719253600000000*Z^2 - 3173602723680000*Z + 4662862914366351)^(1/2) - 5320320)/(- 2000000000*R^2 + 13640000*R - 2000000000*Z^2 + 39120000*Z + 42417447))

th3 = 2*atan((584000000*R + ((2000000000*R^2 - 557640000*R + 2000000000*Z^2 - 39120000*Z + 38861593)*(- 2000000000*R^2 + 557640000*R - 2000000000*Z^2 + 39120000*Z + 119986407))^(1/2) - 81415440)/(2000000000*R^2 - 557640000*R + 2000000000*Z^2 + 544880000*Z + 38990073))

6. Embedded System Architecture

Introducing our system components Arduino, Raspberry Pi 4 Model B, LCD, Keypad, Servo motors, System Components

6.0.1. Control Devices

6.0.1.1. Arduino

Arduino UNO is a microcontroller board based on the **ATmega328P**. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button.

Features :

AVR CPU at up to 16 MHz

1x Dual-mode controller/peripheral I2C

Six PWM channels

RISC-based microcontroller

-**Memory**:

16 KB Flash

512B EEPROM

512B SRAM

-**Power**: 2.7-5.5 volts.

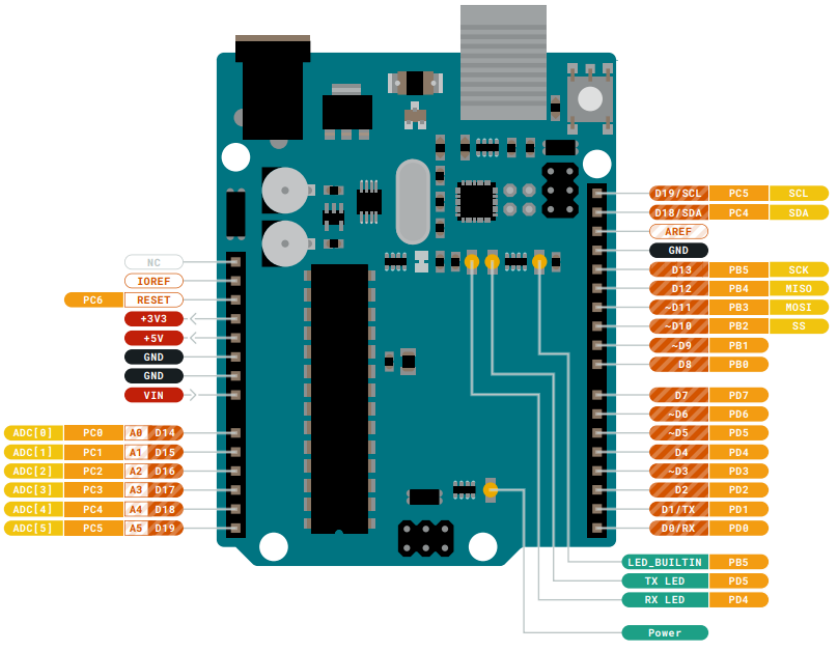


Figure (19) Arduino board

6.0.1.2. Raspberry Pi 4

Raspberry Pi 4 Model B with a 1.5 GHz 64-bit quad-core ARM Cortex-A72 processor, two USB 2.0 ports, two USB 3.0 ports, 1–8 GB of RAM,

-SD card support: Micro SD card slot for loading operating system

and data storage

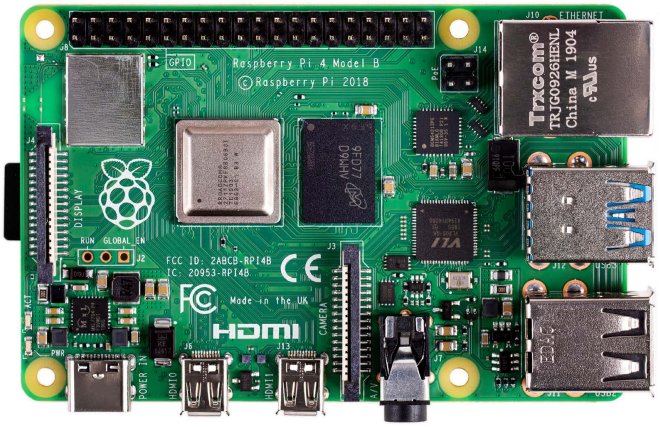


Figure (20) Raspberry Pi Board

6.0.2. Motors

SERVO MOTOR SG90

6.0.2.1. Hardware Connection

Wiring

- Black Wire
  - It is the Ground polar of the motor.
- Red Wire
  - It is the Vcc polar of the motor.
- • Orange Wire
  - It is the Control Signal of the motor.

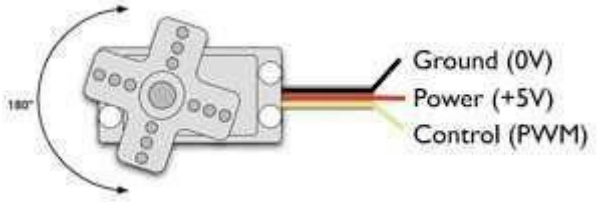


Figure (21) Servo wiring diagram

The pulse width sent to servo ranges as follows

- Minimum: 1 millisecond Corresponds to 0 rotation angle.
- Maximum: 2 milliseconds Corresponds to 180 rotation angle.

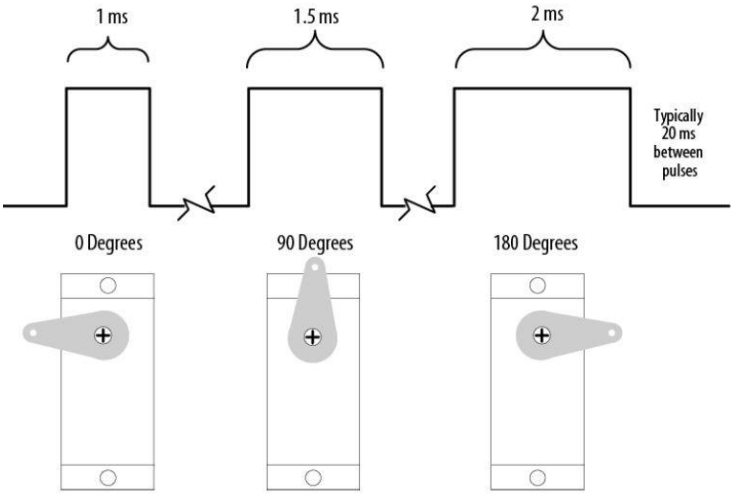


Figure (22) Servo Timing control for the angular position

6.0.2.2. How to control the servo motor

Servos are controlled by sending pulse width modulation (PWM), through the control wire. There is a minimum pulse, a maximum pulse, and a repetition rate. A servo motor can usually only turn 90° in either direction for a total of 180° movement. The PWM sent to the motor determines the position of the shaft, and based on the duration of the pulse sent via the control wire; the rotor will turn to the desired position. The servo motor expects to see a pulse every 20 milliseconds (ms) and the length of the pulse will determine how far the motor turns.

:6.1.2.3. Servo Motors Connection to Arduino

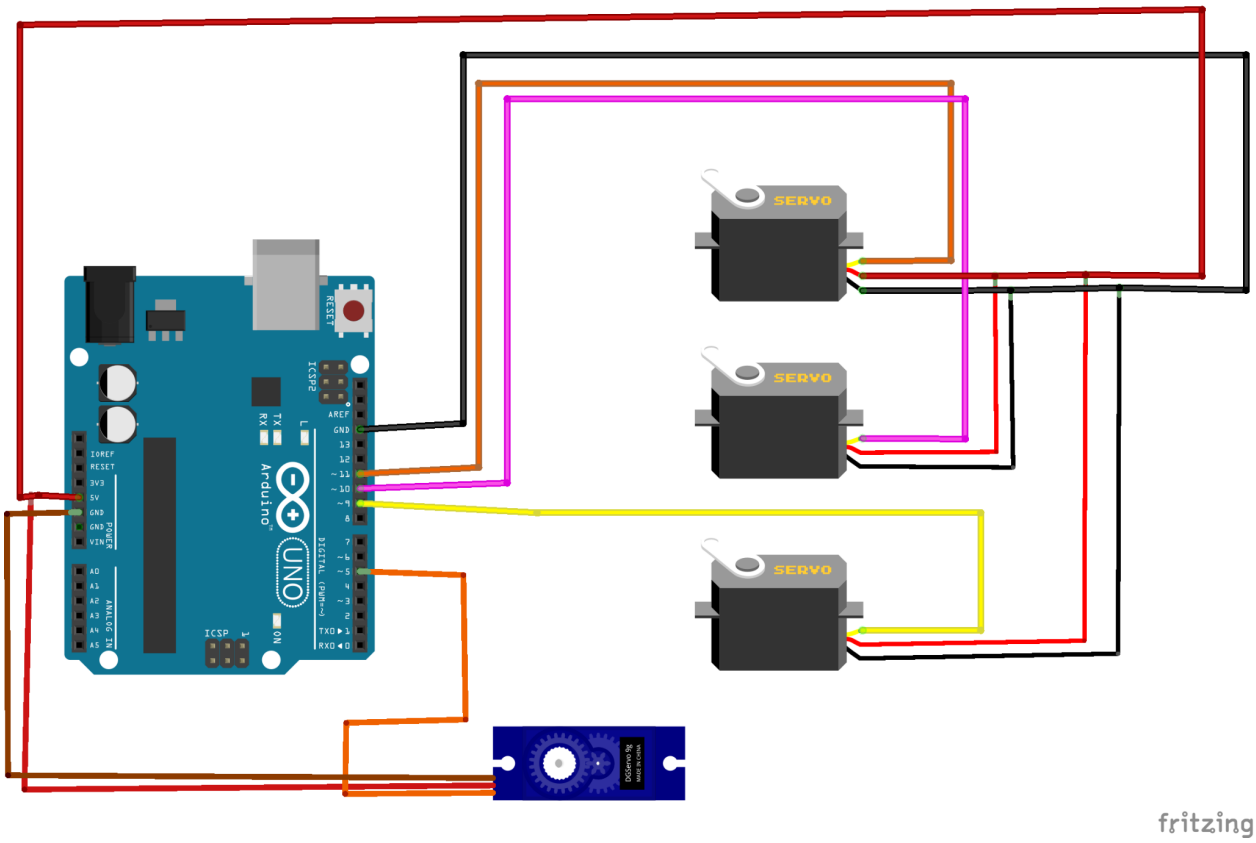


Figure (23) Servo to Arduino connection diagram

6.0.3. Keypad and Switches

The keypad is a set of mechanical switches which are arranged into a matrix.

Figure (24) keypad used to get user inputs

6.0.3.1. How Keypad Works

After arranging the switches into a matrix,

we will have four columns and four rows, So, we can set any set of them as output and the other as an input.

• Assume that the columns set is output, and the rows set is input.

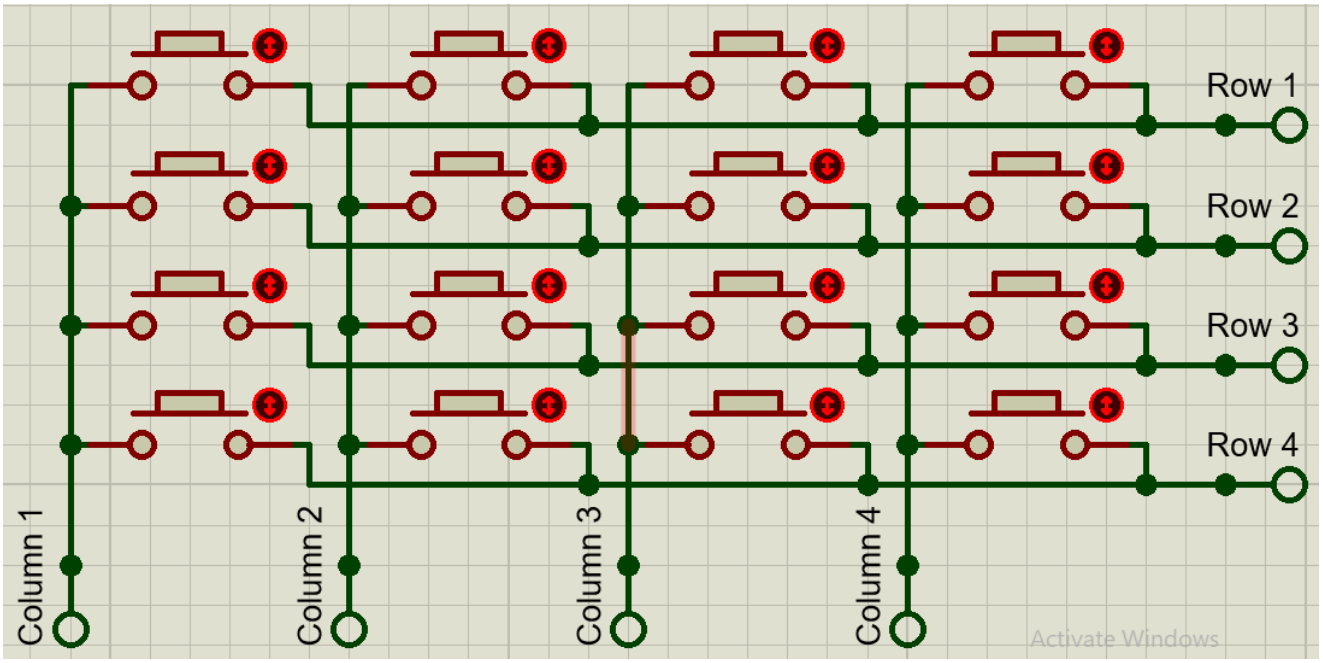


Figure (25) Keypad button matrix

• Set all the columns by HIGH at first.

• Then, change the C1 signal to LOW then check if any row read Zero or not.

• If “R3” reads Zero signal, so the pressed
button is “C1” “ R3”. Then return the value of the pressed key.

6.0.3.2. Keypad Connection to Arduino

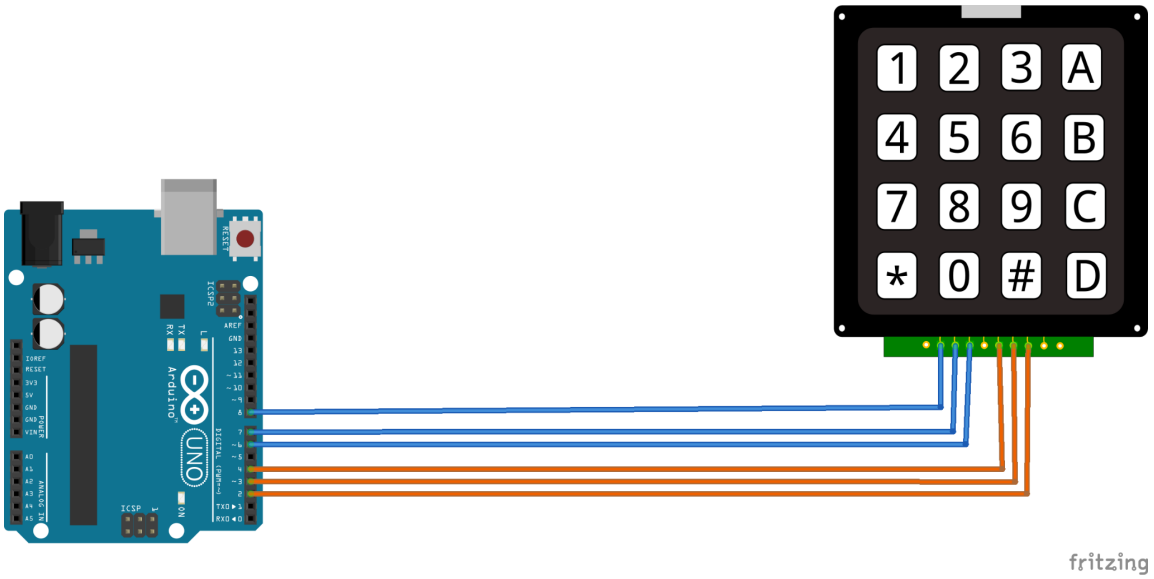


Figure (26) Keypad to Arduino connection

6.0.4. LCD

6.0.4.1. Our LCD Module

16 X 2(That means, the LCD displays are capable of displaying lines each having 16 characters).

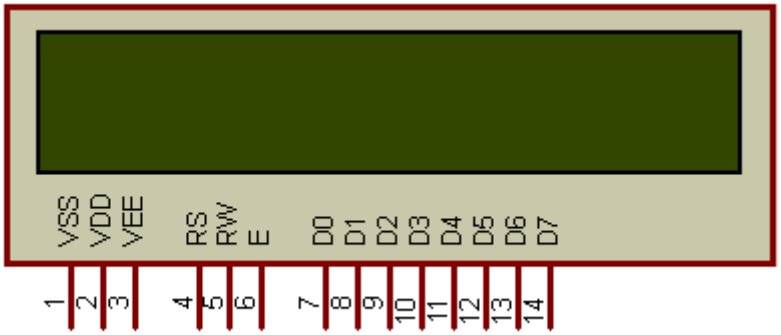


Figure (27) LCD schematic symbol

6.0.4.2. LCD Connection to Arduino

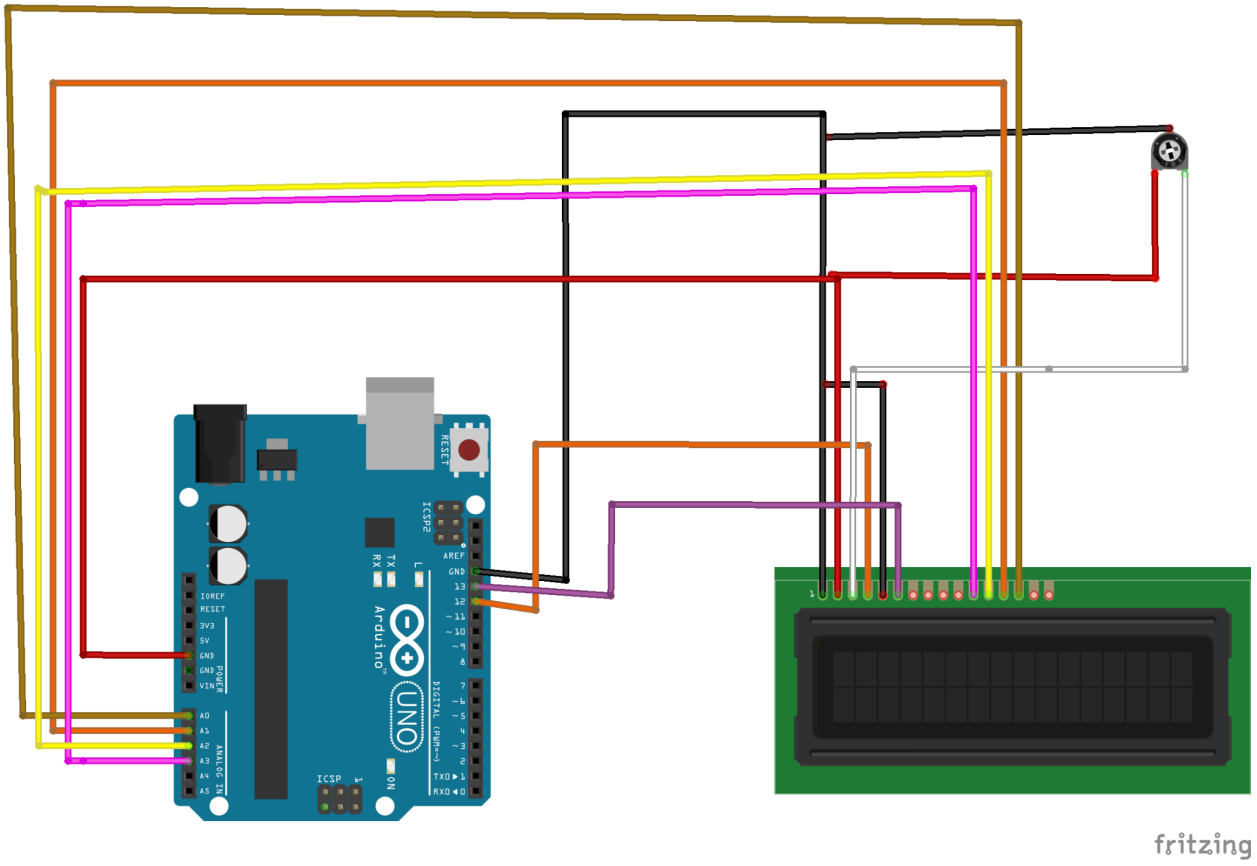


Figure (28) LCD to Arduino connection diagram

6.1. Algorithms

6.1.1. Motor Control Algorithm

6.1.1.1. First Algorithm:

use inverse kinematics to move from one position to another. This algorithm takes target and current positions as an input, it's from Void type (returns nothing) it only moves the arm from a square to another using previously mentioned inverse kinematics.

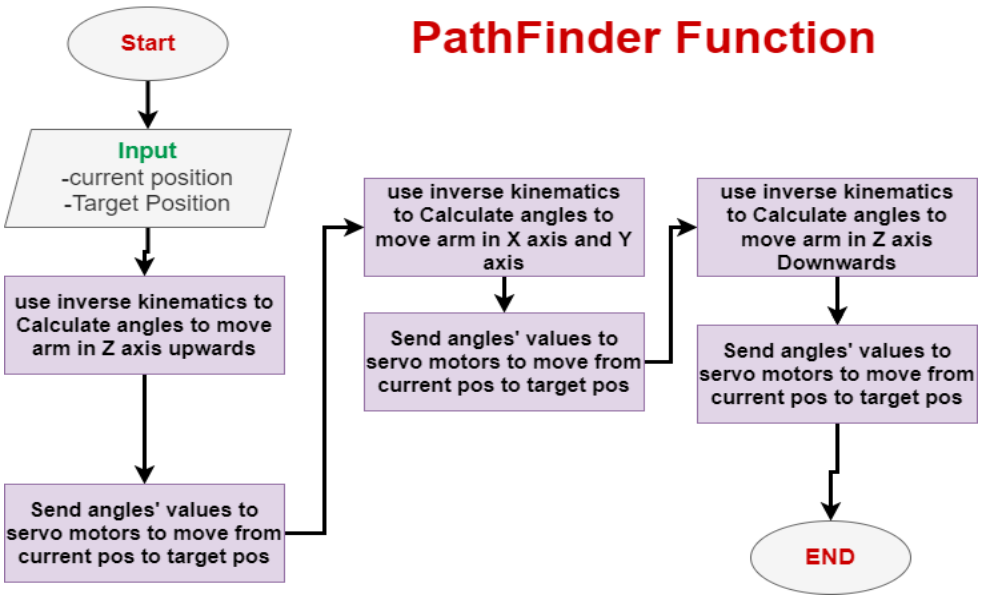


Figure (29) Pathfinder function diagram

6.1.1.2. Second Algorithm

to make the robotic arm pick up a piece from position to another one. It uses a pathfinder algorithm, with added functions to grab and drop pieces (grab the piece from the target position in the first pathfinder function to the target position on the second pathfinder function).

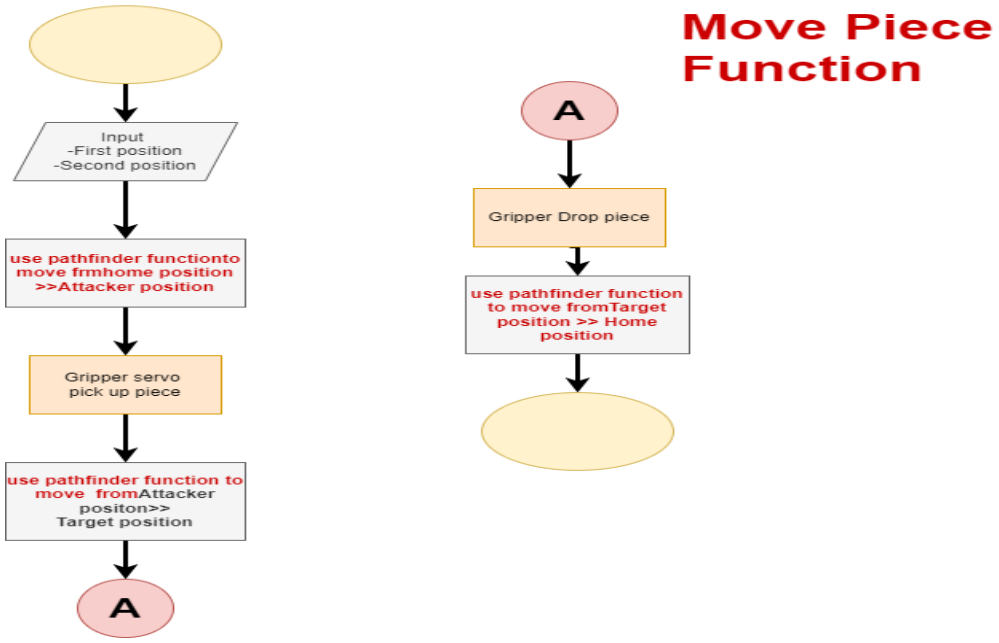


Figure (30) Move piece function diagram

6.1.1.3. Third Algorithm

Slightly different from Move piece, this algorithm moves a piece to an occupied square. This function is used when removing a piece from the chess board by moving the piece to the kill zone and moving another piece to the previously occupied square by the enemy piece.

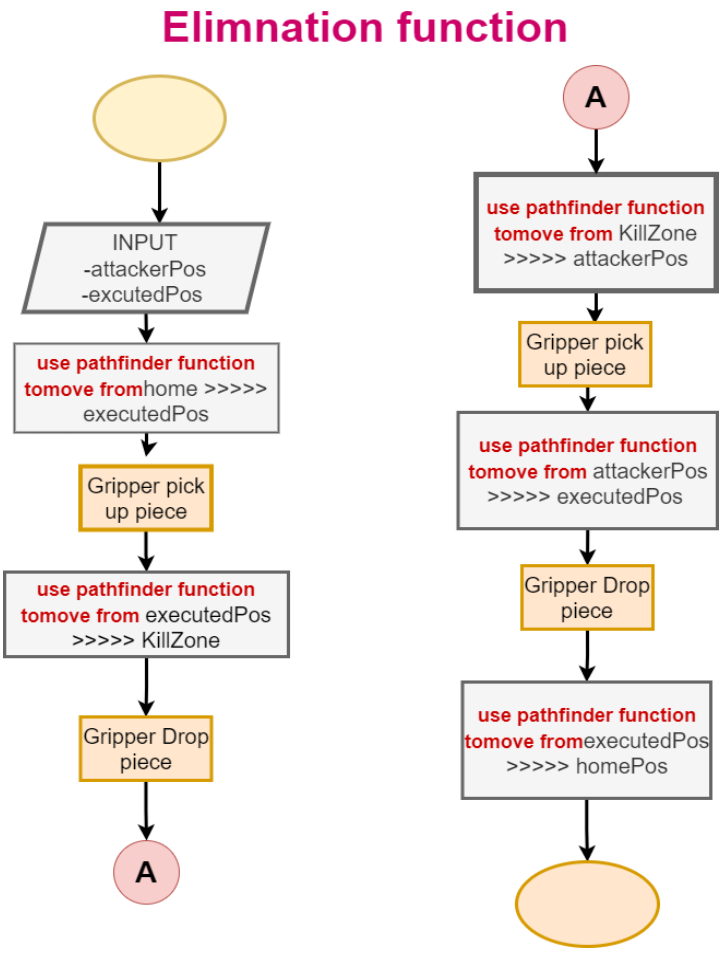


Figure (31) Elimination function diagram

6.1.1.4. Fourth Algorithm

When a user makes a promotion move which piece does he need to promote Queen or bishop or knight or rook. This function is targeted by raspberry pi when the player prompts a pawn.

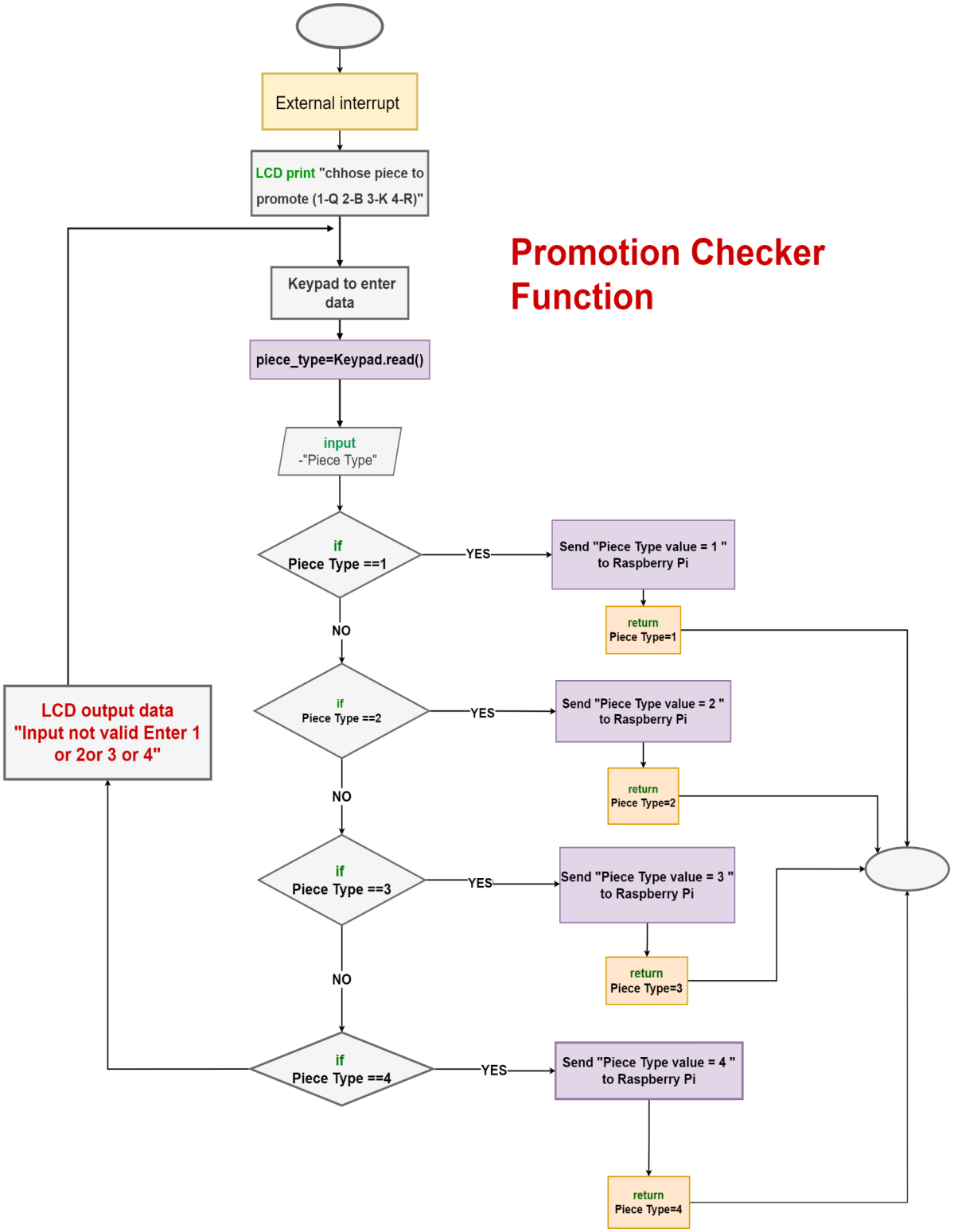


Figure (32) Promotion checker function diagram

6.1.2. Keypad input Algorithm and LCD Output Algorithm

LCD is used as an output device to monitor questions for the user while Keypad is used as an input device to enter his choices.

6.1.2.1. First Question

to choose mode online or offline

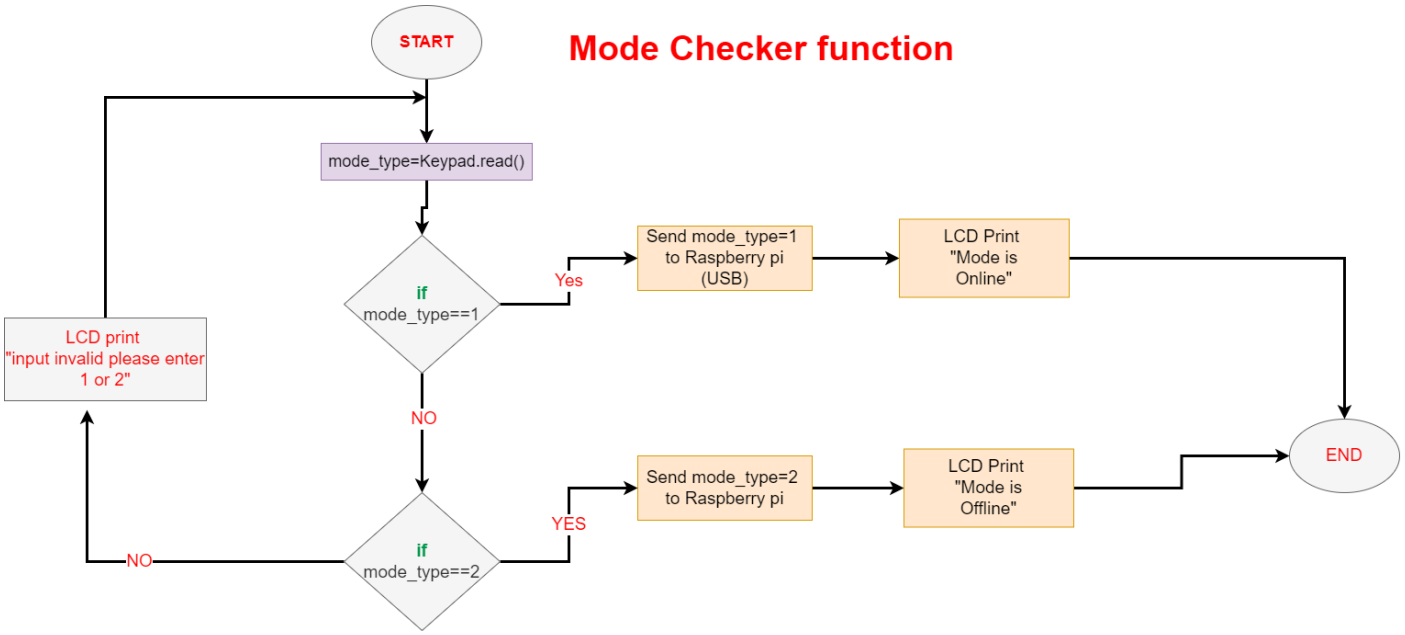


Figure (33) Mode checker function diagram

6.1.2.2. Second Question:

if the user wants to play with black pieces or white pieces

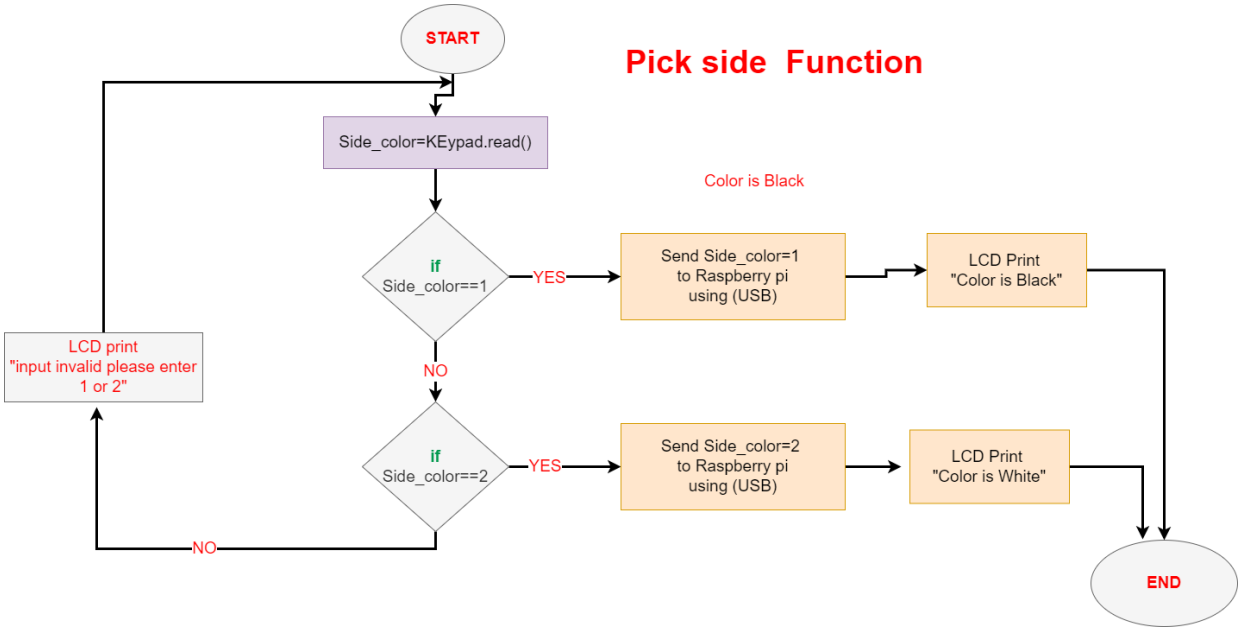


Figure (34) Pick side function diagram

6.1.3. Data/Instruction Communication Algorithm

6.1.3.1. First Communication Protocol is I2C:

Specs of I2C:

-Wired.

-Serial.

-Multi-Master Multi-Slave “MMMS”.

- Half Duplex.

Hardware Connection of I2C:

The TWI protocol allows the systems designer to interconnect up to 128 different devices using only two bi-directional bus lines, one for clock (SCL) and one for data (SDA).

Any master can start the communication at any time.

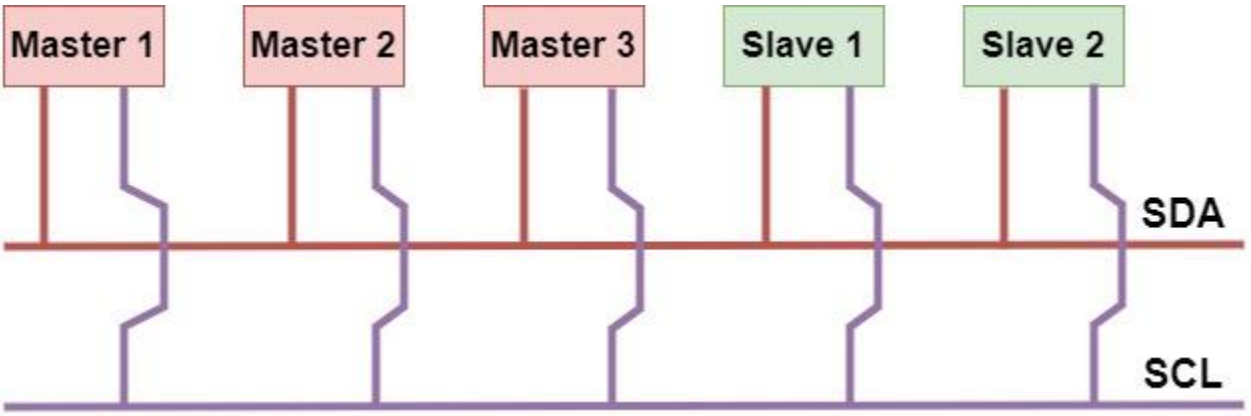


Figure (35) I2C communication lines diagram

General I2C Operation :

The general procedure for a master to access a slave device is the following:

1. Suppose a master wants to send data to a slave:

• Master-transmitter sends a START condition and addresses the slave-receiver

• Master-transmitter sends data to slave-receiver

• Master-transmitter terminates the transfer with a STOP condition

2. If a master wants to receive/read data from a slave:

• Master-receiver sends a START condition and addresses the slave-transmitter

• Master-receiver sends the requested register to read to slave-transmitter

• Master-receiver receives data from the slave-transmitter

• Master-receiver terminates the transfer with a STOP condition.

Data Frame of I2C :

Figure (36) I2C Data frame

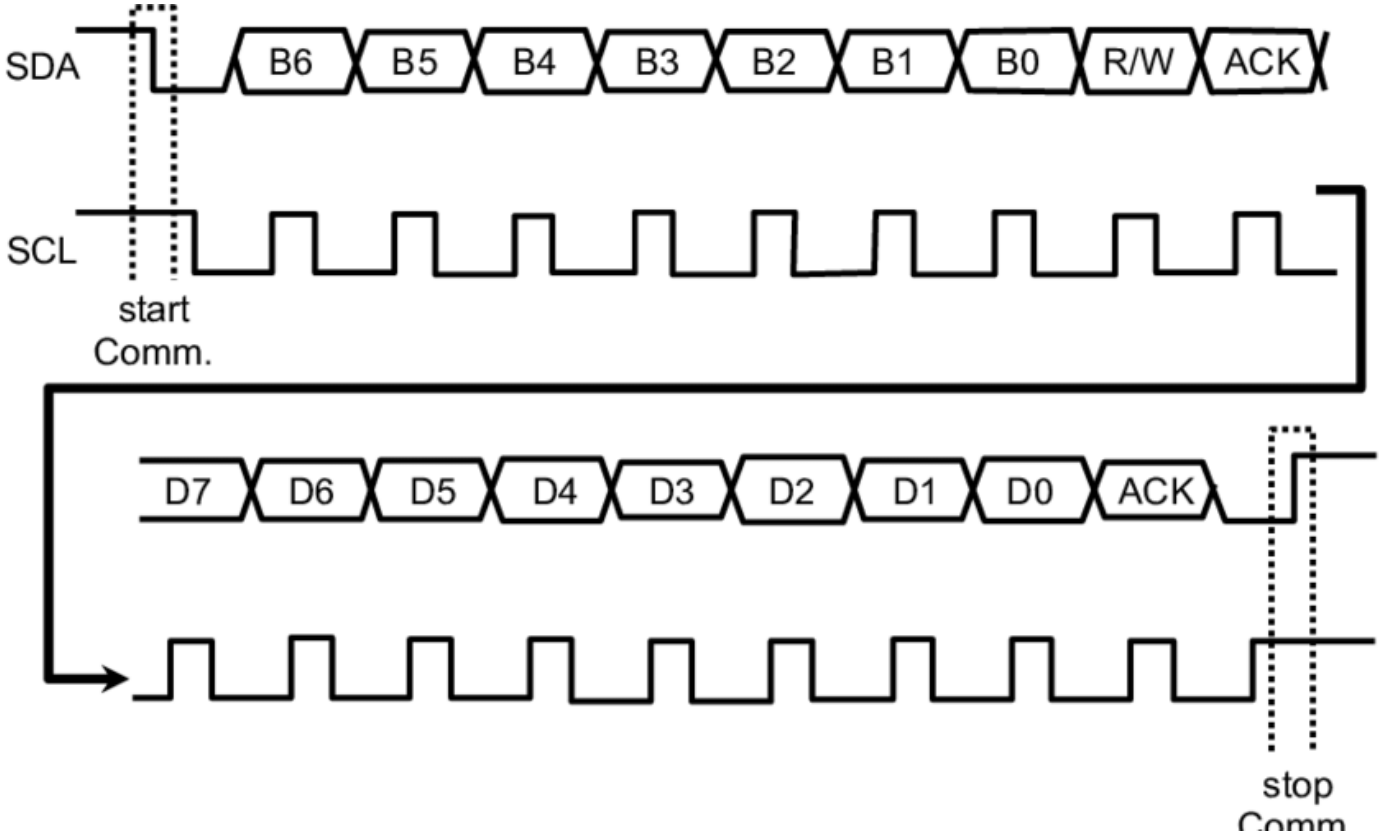


Figure (37) I2C Timing diagram

I2C algorithm:

Chess notation (generated by raspberry pi or web page)is received to Arduino by i2c connection with an extra letter that represents the move type (normal move, elimination, castling or promotion).

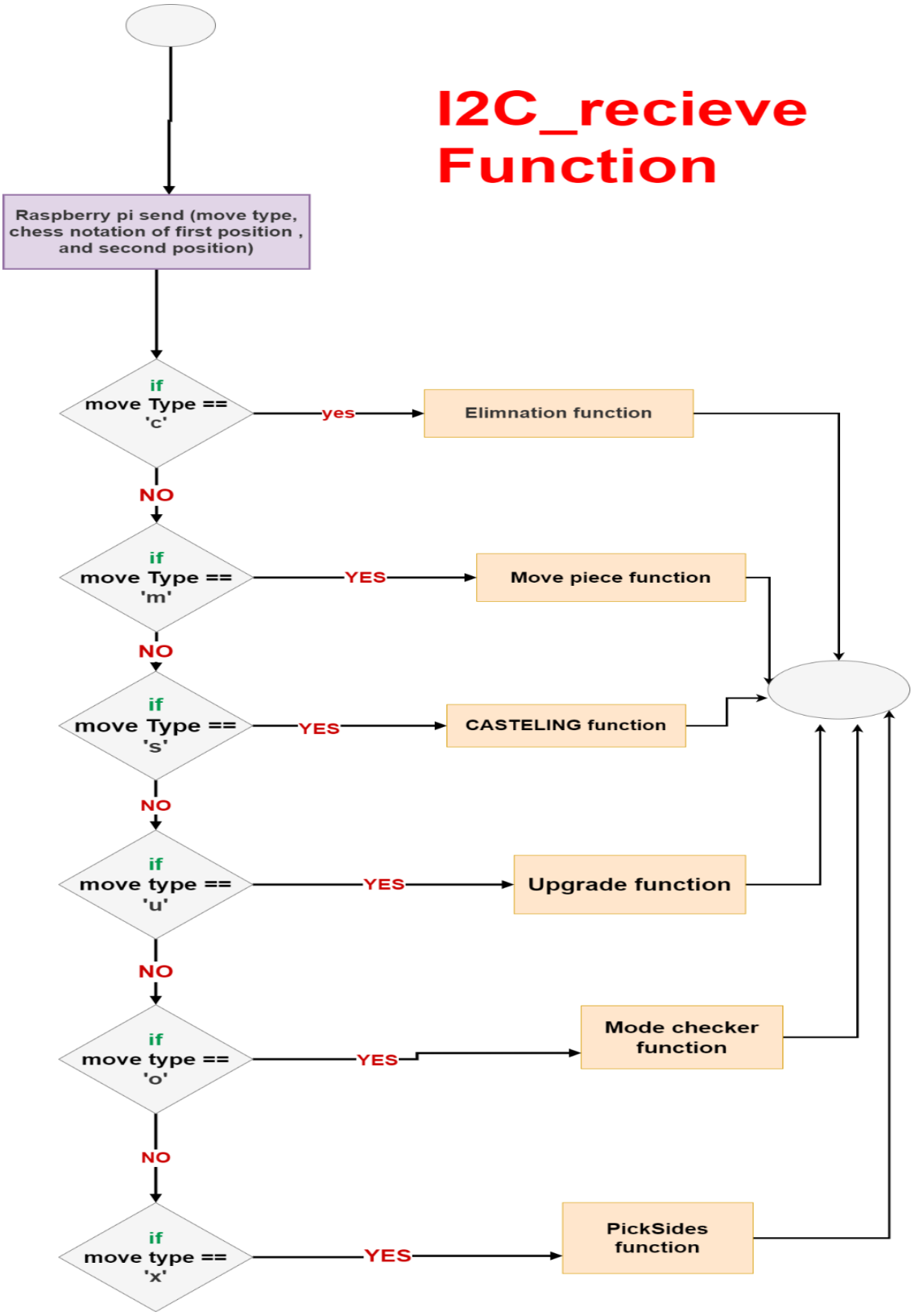


Figure (38) I2C receive function

I2C connection between Arduino and raspberry pi :

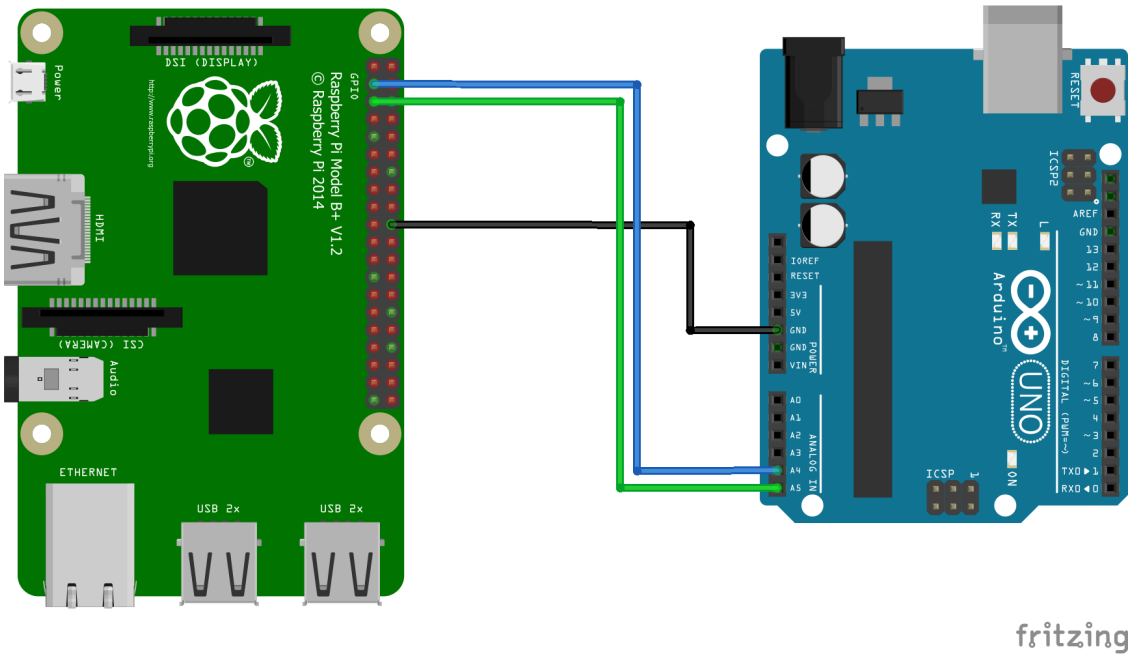


Figure (39) Raspberry to Arduino I2C connection diagram

6.1.3.2. Second communication protocol is USB :

It's used to send data from Arduino(master) to Raspberry pi(Slave)
Universal Serial Bus (USB) is a two-wire serial communication protocol. It allows 127 devices to be connected at any given time. USB supports plug & play functionality.
USB protocol sends and receives the data serially between host and external peripheral devices through data signal lines D+ and D-. Apart from two data lines, USB has VCC and Ground signals to power up the device. The USB pinout is shown in the Figure below.

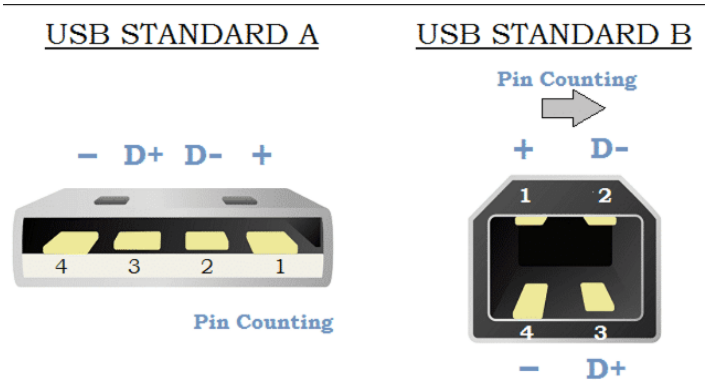


Figure (40) USB connector sizes

USB-A is used for the Raspberry pi side.
USB-B is used for the Arduino side.

connection :

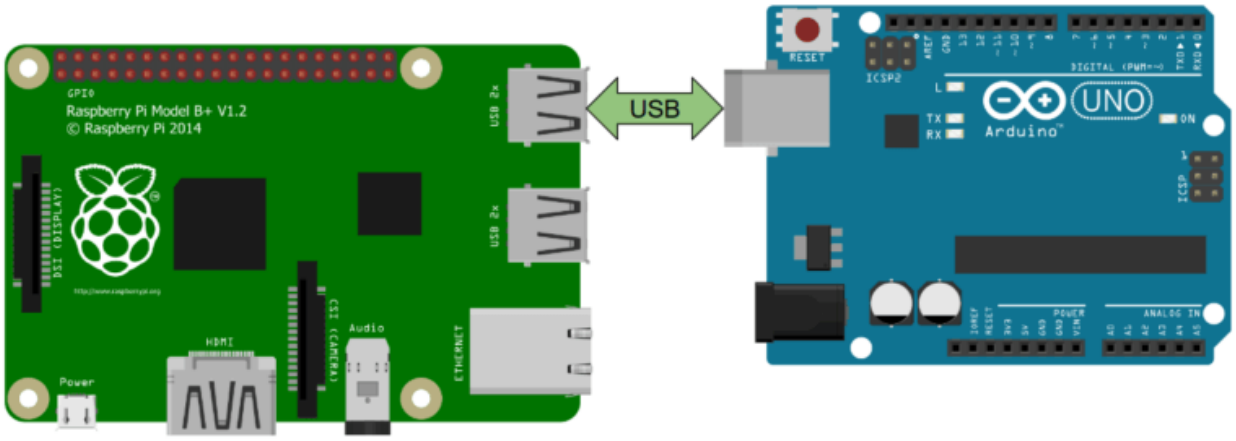


Figure (41) Raspberry to Arduino USB connection diagram

DATA Packets

The data packet may be of variable length, dependent upon the data. However, the data field will be an integral number of the bytes. There are two types of data packets each capable of transmitting up to 1024 bytes of data..

USB protocol PID

There are 4 bits to the PID, however, to ensure it is received correctly, the 4 bits are complemented and repeated, making an 8-bit PID in total.

- PID (Packet Identifier Field): PID stands for Packet ID in USB protocol. This field is used to identify the type of packet that is being sent and hence the format of the packet data. The remaining 4 bits are a 1's complement of this, acting as the check bits. Part of this field determines which of the four groups (token, data, handshake, and special) the packet belongs to, and also specifies an input, output, or setup instruction.

Endpoint Field: The endpoint field in the USB protocol is 4 bits long and allows for additional flexibility in addressing. Endpoints are usually split for the data going IN or OUT. Endpoint 0 is a special case referred to as the CONTROL endpoint and every device has an endpoint 0.

CRC Field: The Cyclic Redundancy Checks are performed on the data within the packet payload. All token packets have a 5-bit CRC while the data packets have a 16-bit CRC.

EOP Fiel: Each packet is terminated with an End of the Packet (EOP) field. This consists of a single-ended zero (SE0) for two-bit times followed by the J for 1-bit time.

USB Algorithm

This algorithm is mainly used to communicate to raspberry pi, transimating input. When we want to upgrade a piece, the piece type is sent from the Arduino to the Raspberry Pi by .

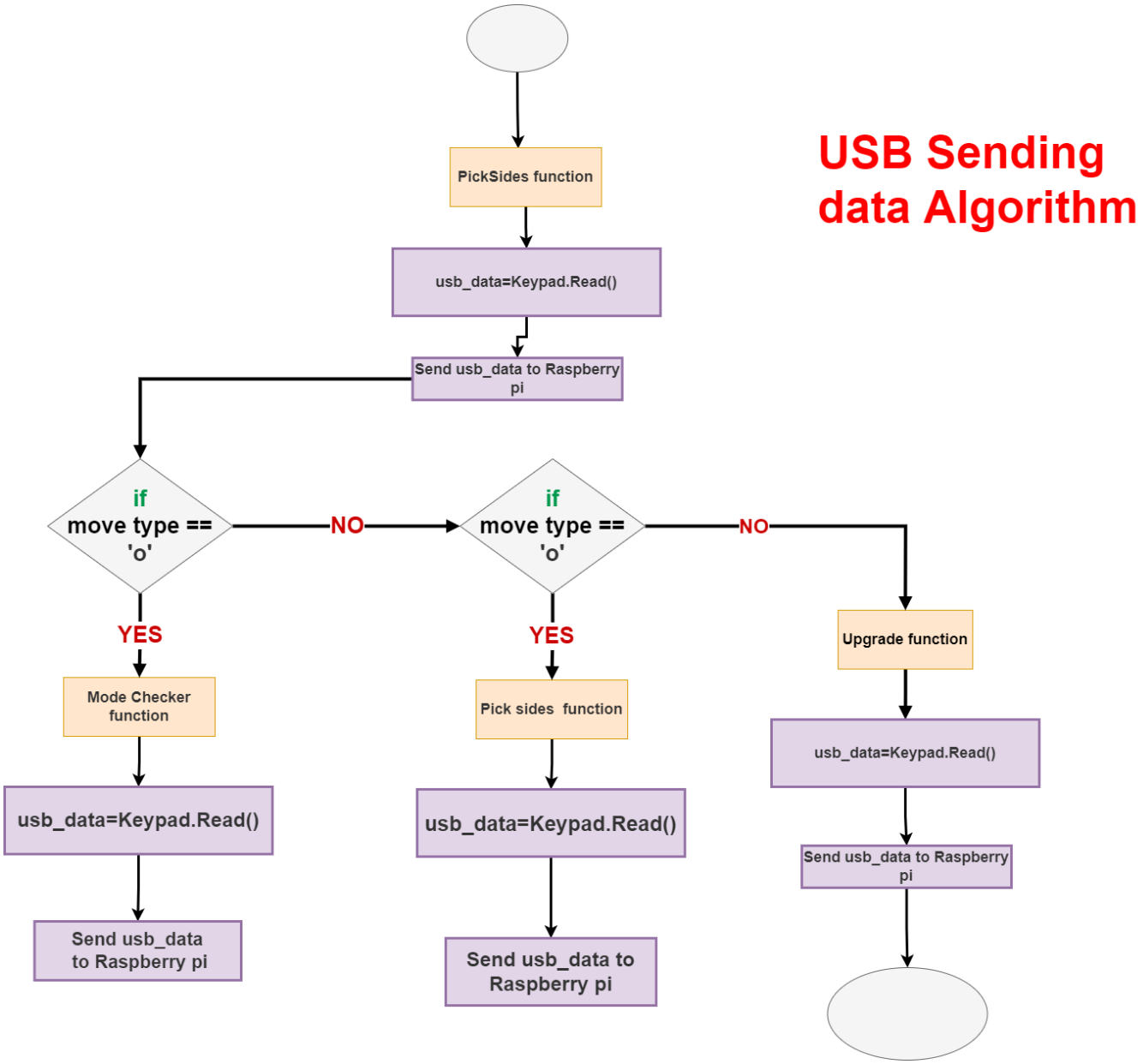


Figure (42) USB communication diagram

7. Image Processing

7.1. Image Capturing

Image capturing is done on the raspberry pi via a number of methods. The PiCamera module in python, the OpenCV interface for legacy camera interface, or libcamera API of the Raspberry Pi os. Unfortunately, the OpenCV library is not compatible yet with the new libcamera API and it uses legacy camera API, this introduces a challenge though for the OpenCV implementation is not very flexible with the raspberry pi camera module, moreover, we can't set the capturing parameters to utilize the full resolution and raw data of the camera module, as for the lib camera we can use it from the command-line interface (CLI) and set the resolution and format to the desired values but it doesn't interface that easily with the code base, a solution was developed through invoking a system process via the python code that launches the Shell and runs a command that saves the image in an intermediate file at a specific location in the code files hierarchy then we open the file with OpenCV interface and extract all the wanted data. After that, we delete the temporary file so that it won't cause any problems with the code later on.

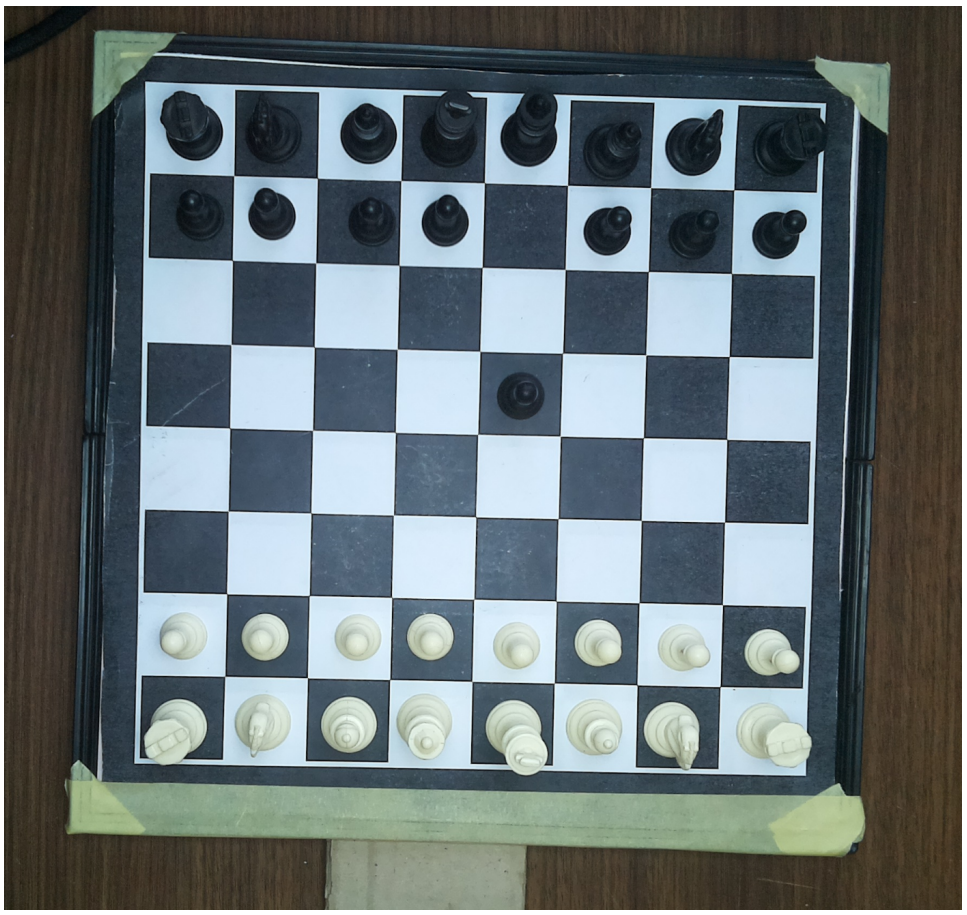


Figure (43) Board image taken by pi camera

7.2. Board Detection

After capturing the image from the camera comes the issue of detecting the chess board, here we invoke the “findChessBoard” method of the OpenCV library while enabling the option for the use of multipath search by applying multiple filters like an adaptive binary threshold, etc…, this method then returns and flag for the state of success or failure, and an array of tuples with the positions of the inner points of the chess board grid.

Figure (44) Output of the chess board finding function showing the inner board corners

7.3. Perspective Warp Transformation

Given that the board positioning will not be perfect every time the robot is started we can't rely on the shape of the board to be a perfect square on every frame. And with a need for splitting the board into individual tiles, this makes it impossible to rely on a hardcoded algorithm to cut the board. Thus more work would have to be done on the image. A four-point perspective wrap algorithm is implemented to transform the board to a perfect square from every frame.

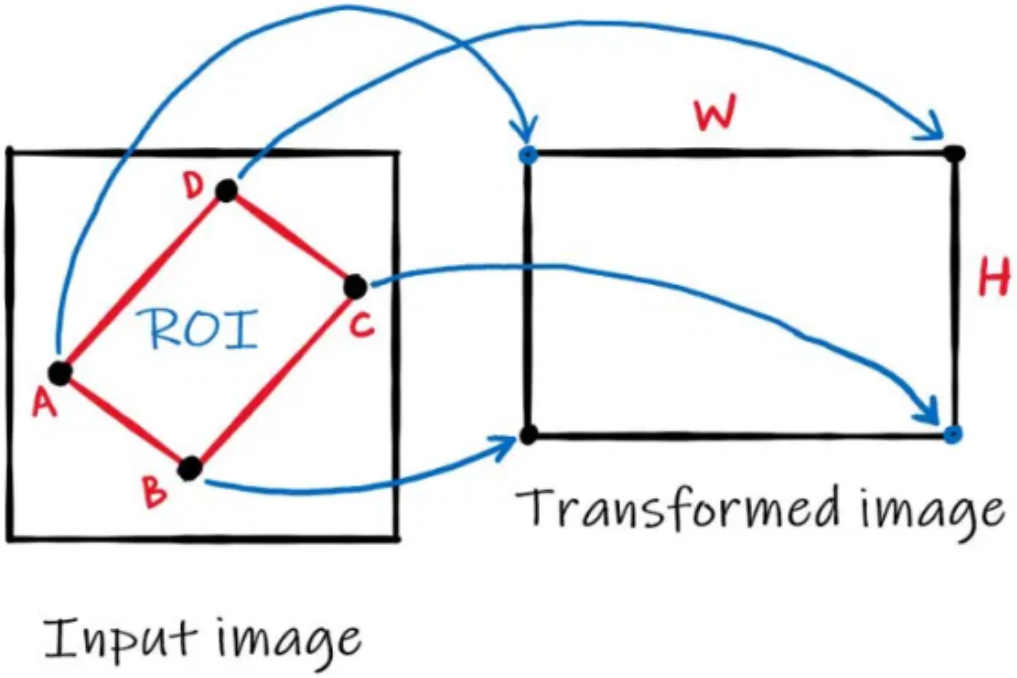


Figure (45) a perspective transform illustration

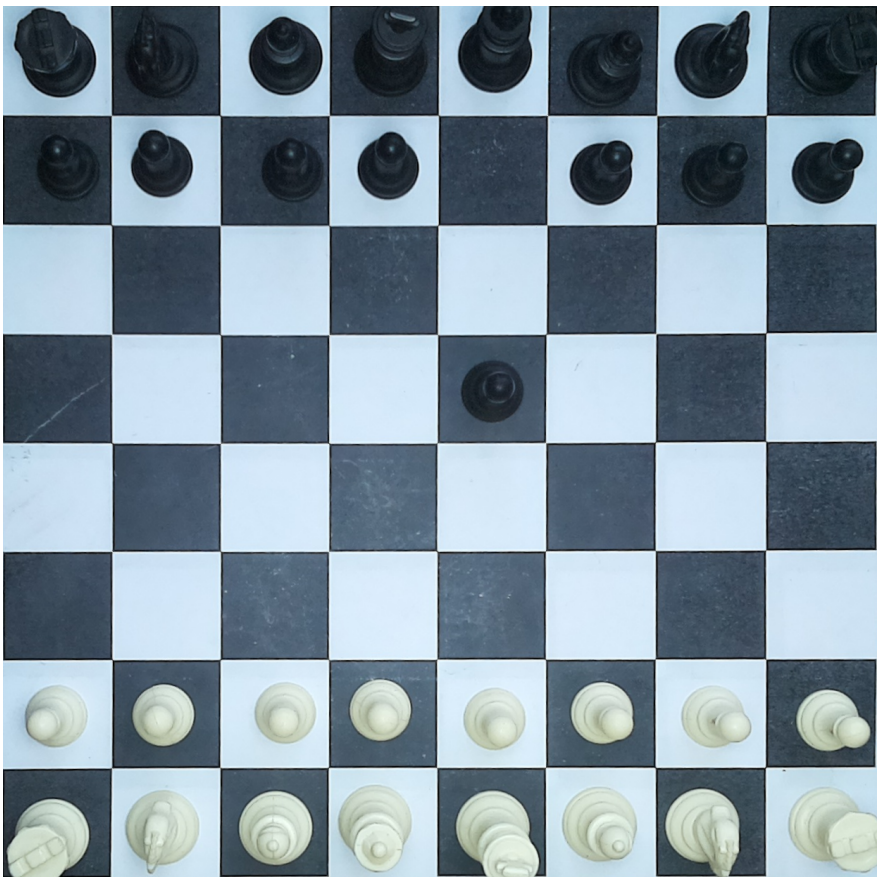


Figure (46) board after perspective transformation

7.4. Tile Splitting

After the perspective warp transformation, the extracted board is guaranteed to be a square with a specific size. Then a simple slicing algorithm splits each axis of the board into 8 regions and then extracts each tile (square) into a separate image object.

Figure (47) extracted individual tiles from the chess board

7.5. Data Collection

The data collection process was an arduous process that took a long time. is done using a program that uses all the previously mentioned helper functions to collect the data that is needed to train a CNN model. The model used is quite simple. Since it is a simple task at hand. The CNN is needed to recognize whether the square provided is empty, has a white piece, or has a black piece. Using this information all chess moves made by humans on the board can be recognized. After that, the move is passed to the WCRA chess engine on appropriate chess notation.
The program opens a video stream via the OpenCV library and waits for the input to capture an image. While waiting the human operator can change the position of the board and the positions of the chess pieces and lighting conditions. After the image is captured it identifies the chess board and applies the warp transformation to it in order to extract it into the shape of a square then slices the board into individual tiles. And rename each one of the tiles according to two counters. The first tracks the board number and the second tracks the tile number. The naming is in this form “Board#Square#” where # is a placeholder for the counters. A 314 board image was collected making a total of 314 * 64 = 20096 image used for training. After that, a classification of the data into three categories is done by hand (With the help of a windows file interface).
Row data was uploaded to Kaggle as an open-source and can be accessed by the following link (QR code).
https://www.kaggle.com/datasets/mohamedmoataz99/wcra-ai

8. CNN Chess Piece Recognition

8.1. Convolutional Neural Network(CNN)

“Natural intelligence is a product of evolution. Therefore, by simulating biological evolution, we might expect to discover how living systems are propelled towards high-level intelligence.”

— Michael Negnevitsky Books, Artificial Intelligence: A Guide to Intelligent Systems

Biology inspires us with lots of elegant solutions to our problems. One of these is Artificial Neural networks (ANN). The field of machine learning has taken a dramatic twist in recent times, after the rise of the Artificial Neural Network (ANN). These biologically inspired computational models are able to far exceed the performance of previous forms of artificial intelligence in common machine learning tasks. a neural network is a structure trained to recognize input patterns taking a vector of input values and producing a vector of output values to train the weights of neurons. Actually, neural networks were invented a long time ago, in 1943, when Warren McCulloch and Walter Pitts created a computational model for neural networks based on algorithms. Then the idea went through a long hibernation because the immense computational resources needed to build neural networks did not exist yet. Recently, the idea has come back in a big way, thanks to advanced computational resources like graphical processing units (GPUs). They are chips that have been used for processing graphics in video games, but it turns out that they are excellent for crunching the data required to run neural networks too. That is why we now see the proliferation of neural networks. One of the most impressive forms of ANN architecture is that of the Convolutional Neural Network (CNN). CNN's are primarily used to solve difficult image-driven pattern recognition tasks and with their precise yet simple architecture, offer a simplified method to solve our problems.

A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of a combination of linear and nonlinear operations, i.e., convolution operation and activation function. Convolution is a specialized type of linear operation used for feature extraction, where a small array of numbers, called a kernel, is applied across the input, which is an array of numbers, called a tensor. An element-wise product between each element of the kernel and the input tensor is calculated at each location of the tensor and summed to obtain the output value in the corresponding position of the output tensor, called a feature map (Figare 42). This procedure is repeated by applying multiple kernels to form an arbitrary number of feature maps, which represent different characteristics of the input tensors; different kernels can, thus, be considered as different feature extractors (Fig. 43). Two key hyperparameters that define the convolution operation are the size and number of kernels. The former is typically 3 × 3, but sometimes 5 × 5 or 7 × 7. The latter is arbitrary and determines the depth of output feature maps.

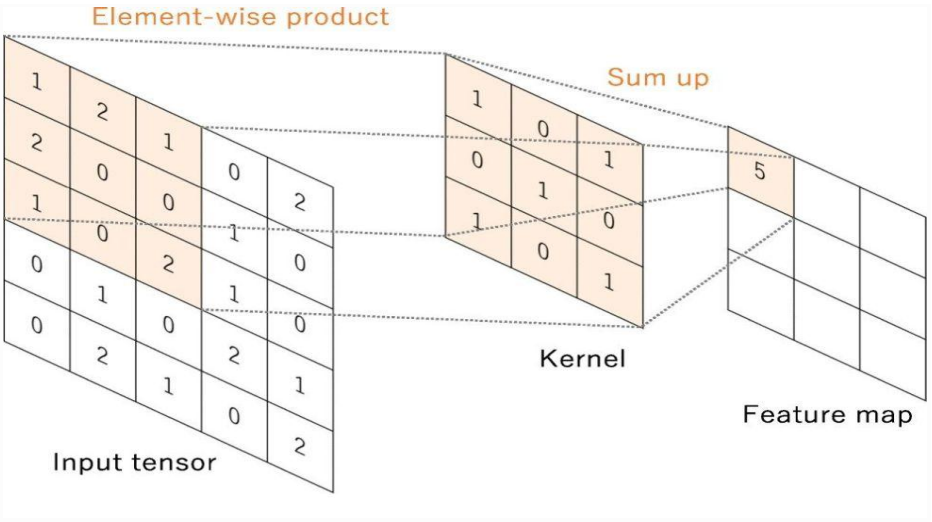


Figure (48) An example of convolution operation with a kernel size of 3 × 3, no padding

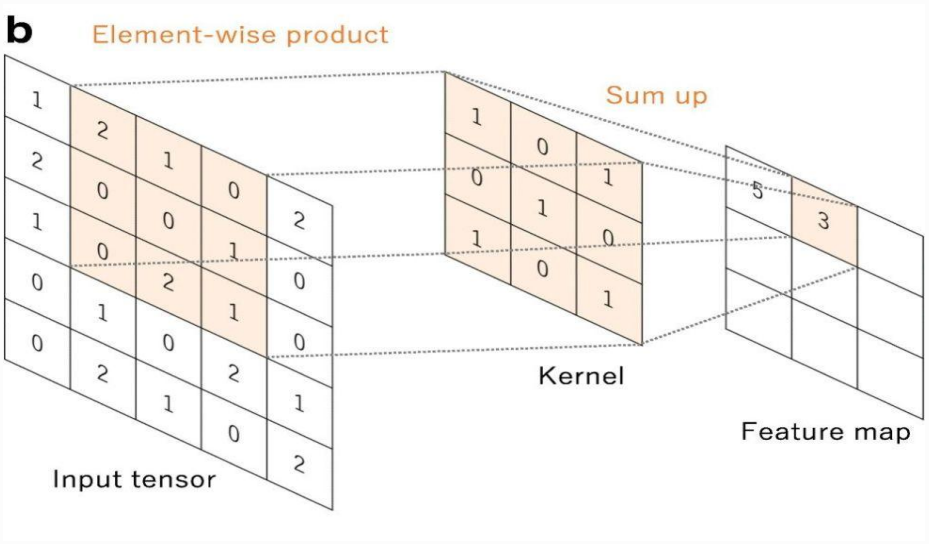


Figure (49) An example of convolution operation with a kernel size of 3 × 3, no padding

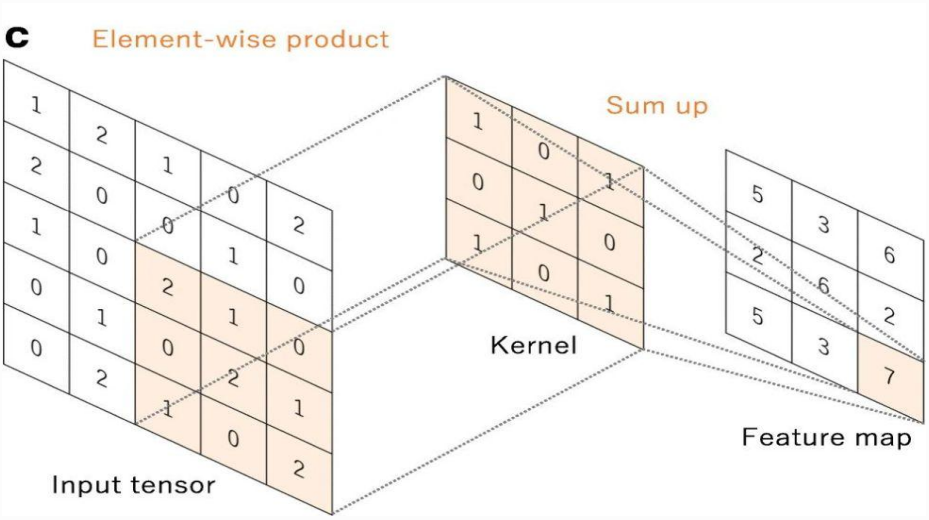


Figure (50) An example of convolution operation with a kernel size of 3 × 3, no padding

8.2. Motivations to Use CNN

Using CNNs for deep learning is popular due to three important factors:

- CNNs eliminates the need for manual feature extraction(features are learned directly by the CNN).
- CNNs produce highly accurate recognition results.
- CNNs can be retrained for new recognition tasks, enabling you to build on pre-existing networks.
- CNNs provide an optimal architecture for uncovering and learning key features in image and time-series data. CNNs are a key technology for many applications.

8.3. Data Preparation

After collecting the images needed for the model training some preparation had to be done. First using the OpenCV library the data has been read from the desk and then stored in NumPy arrays. The NumPy arrays are arrays that are optimized for efficiency in calculations but they store the image data as raw data thus they take a very large space. Due to the size of the images in the NumPy arrays, they cannot be saved that way, luckily there is a way to solve this. We can save the NumPy arrays in a compressed form called (.npz) files. These files act as a file compressor using the zip algorithm but are optimized for NumPy arrays.

8.4. Model

8.4.1. Architecture

Simple problems need simple solutions, has a simple problem on hand a simple CNN architecture is used.

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 188, 188, 3)       84

 max_pooling2d (MaxPooling2D  (None, 37, 37, 3)        0
 )

 conv2d_1 (Conv2D)           (None, 35, 35, 3)         84

 max_pooling2d_1 (MaxPooling  (None, 17, 17, 3)        0
 2D)

 flatten (Flatten)           (None, 867)               0

 dense (Dense)               (None, 100)               86800

 dense_1 (Dense)             (None, 3)                 303

=================================================================
Total params: 87,271
Trainable params: 87,271
Non-trainable params: 0
_________________________________________________________________
```

Figure (51) CNN model specs (Summary).

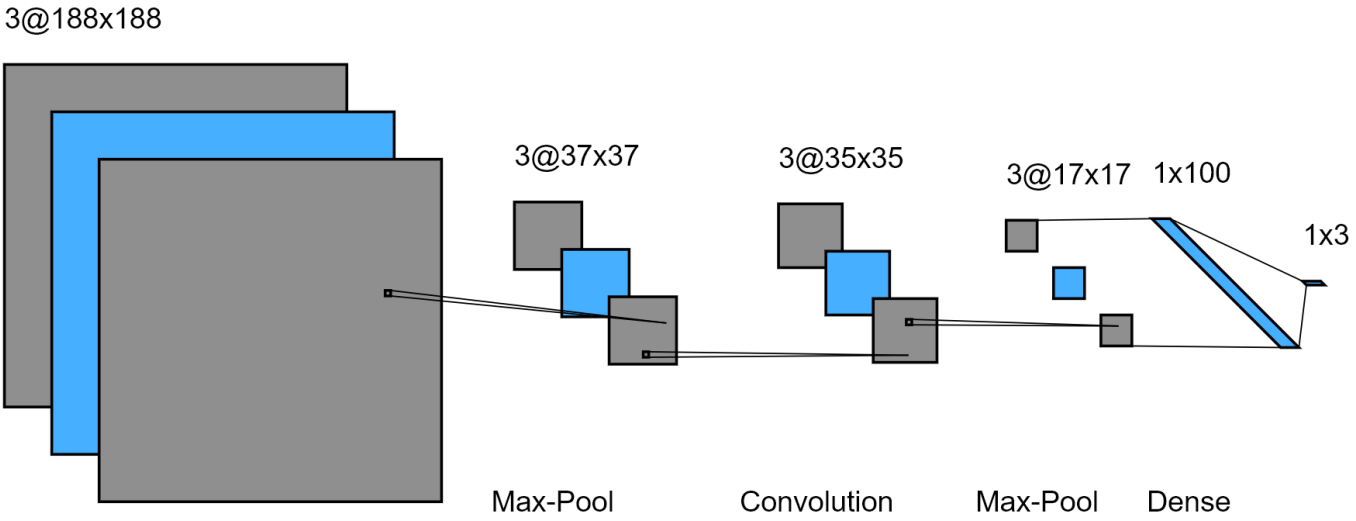


Figure (52) CNN model Archetictre.

The model consists of 7 layers Two conv2d layers (2D Convolution Layer, this layer creates a convolution kernel that is wind with layers input which helps produce a tensor of outputs.) Two max_pooling layers (Downsamples the input along its spatial dimensions (height and width) by taking the maximum value over an input window (of size defined by pool_size) for each channel of the input. The window is shifted by strides along each dimension). Flatten layer (used to make the multidimensional input one-dimensional, commonly used in the transition from the convolution layer to the fully connected layer). then our fully connected NN. output layer with 3 neurons representing our classification stats.

8.4.2. Training

Tensorflow provides us with the methods needed to train this model, our data was divided to 75% for training and 25% testing, with a relu as an activation function , $1 * 10^{-4}$ learning rate , 150 batch size and 100 epochs. training took 5 hours and gives a total accuracy of 0.9914 , which is great and gives amazing results after further tasting.

9. Chess Engine

9.1. Introduction To Chess Engine

Chess is a zero-sum, Turn-based, complete information (players can see the state of the game at all times) board game played between two players. That means any movement will result in an advantage for one side and an equivalent loss for the other. Our game can be visualized as a tree (figure 47). A Depth First Search Algorithm (DFS) can be used to guide us to traverse the tree and pass down the level value. The system derived its playing strength mainly from brute force computing power, But can we solve chess by brute force?

Minimax algorithm is a DFS algorithm to solve such zero-sum games. Applying alpha-beta pruning will lead to better results (an optimization technique for the minimax algorithm.), those methods are the same methods used by Deep Blue (IBM-developed supercomputer) team to win against the world chess champion Garry Kasparov in the 1997 rematch.

An evaluation function is required to judge each chess possession (tree node) to help Minimax choose the best (or accepted) chess move. Such a function can be billed by some domain knowledge. Do we have the eunuch domain knowledge to develop such a function?. Can we develop one with no domain knowledge?

Creating a computer program that can play chess is a daunting but interesting task, not to mention adding it to a robot arm. even when trying to achieve only a rudimentary level of intelligence. Multiple chess engines abound on the internet, both closed and open source, and each one requires either a team of professional developers working over a moderate period of time or a single expert working over a long period of time to create. Adding our own chess engine to the robot was really a plus, A lot was learned during this journey.

It's been decided to use python as a programming language for the sake of simplicity. Also, We don't have to worry about making the tree this thing is taken care of by recursion.

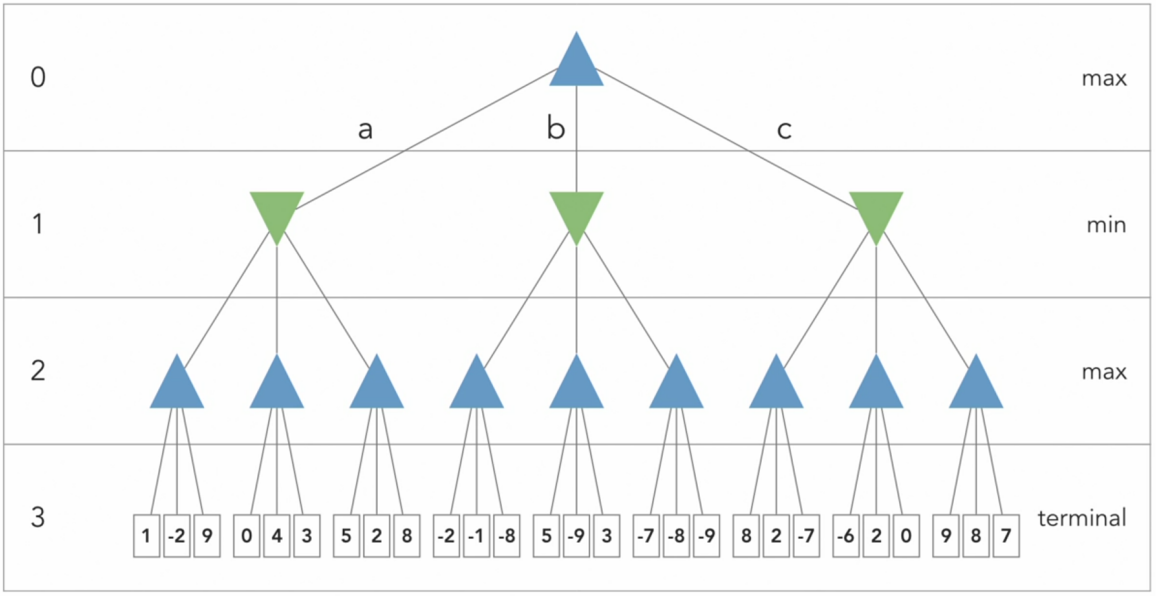


Figure (53) search tree

9.2. A Brute Force Attempt to Solving Chess

Chess is not in the category of solved games (strategy for playing perfectly without ever losing is unknown). Let's do some assumptions to show why:
Consider a game with only 40 pairs of moves (40 for white, 40 for black)
So we have 10^{40} different games, if there is a very very powerful computer that can process a game in 1ns, producing one move would take 10^{40}ns, that's about $300 * 10^{21}$ years, a million-core supercomputer would take $300 * 10^{15}$ years.
See (figure 48) to know how big is $300 * 10^{15}$

Age of the Earth:	4,500,000,000 years
Age of the universe:	13,800,000,000 years
Producing one move:	300,000,000,000,000,000 years

Figure (54) Time table example

If we are going to use this approach we must do better than that.

9.3. Formalization of the problem

A game can be defined as a type of search in AI which can be formalized of the following elements:

- Initial state: It specifies how the game is set up at the start.
- Player(s): It specifies which player has moved in the state space.
- Action(s): It returns the set of legal moves in state space.
- Result(s, a): It is the transition model, which specifies the result of moves in the state space.
- Terminal-Test(s): Terminal test is true if the game is over, else it is false in any case. The state where the game ends is called the terminal state.
- Utility(s, p): A utility function gives the final numeric value for a game that ends in terminal states s for player p. It is also called the payoff function. For Chess, the outcomes are a win, loss, or draw and its payoff values are +1, -1, 0. And for tic-tac-toe, utility values are +1, -1, and 0.

This utility function requires the game to be in a terminal state. We discussed that we cannot solve chess by brute force. Likely we don't have to end the game to get a good chess move. We will call an evaluation function (something to judge the current chess position) at each node and stop our search tree at a certain depth and choose the best available move according to our evaluation function.We can likely develop one or use the stockfish NNUE evaluation function but we will talk about it later.

9.4. Minimax Algorithm

Minimax is a decision rule used in artificial intelligence, decision theory, game theory, and statistics, for minimizing the possible loss for a worst-case (maximum loss) scenario. It is used primarily in zero-sum games played against computers. A zero-sum game is one in which points awarded to one player come at the expense of the other player, therefore maintaining an overall balance between players. One example of a zero-sum game is, of course, chess. Minimax is essentially an operation on a tree that minimizes the worst-case scenario. Each level of the tree, which can have any integer as a branching factor, represents the options available to the player in a given turn (valid moves). Because chess is a two-player game, each level of the tree represents the alternating turn order. Each branch in the tree represents a choice by the player. The operation of the algorithm is as follows. The computer maps out all of the possible moves or

game state up to the number of turns it wants to look ahead (depth). Once the tree is generated, scores are assigned to the terminal nodes representing the relative value to the computer. For example, if each move in chess can be scored from 1 to 100, numbers closer to 100 might be better for the computer, and numbers closer to 1 might be better for the human. For a tree with a depth of 2, the computer would simply choose the node with the highest number, and take that branch (make its move). However, in games like chess, players must think more than one step ahead, or they are usually doomed to fail. Therefore, if the depth of the tree is 3, the branches to the terminal nodes represent the options available to the human player. The computer assumes the human will choose his or her own best move and uses this fact to determine what the human will most likely do given each possible move by the computer. Yet again, if the computer wishes to look a further step ahead, the next level in the tree represents the moves the computer can make after the human has responded to its first move. This search can continue for as deep as the computer is instructed to go. It is now clear why the algorithm is named Minimax: it is a series of alternating choices between a minimum number (best for a human player) and a maximum number (best for a computer).

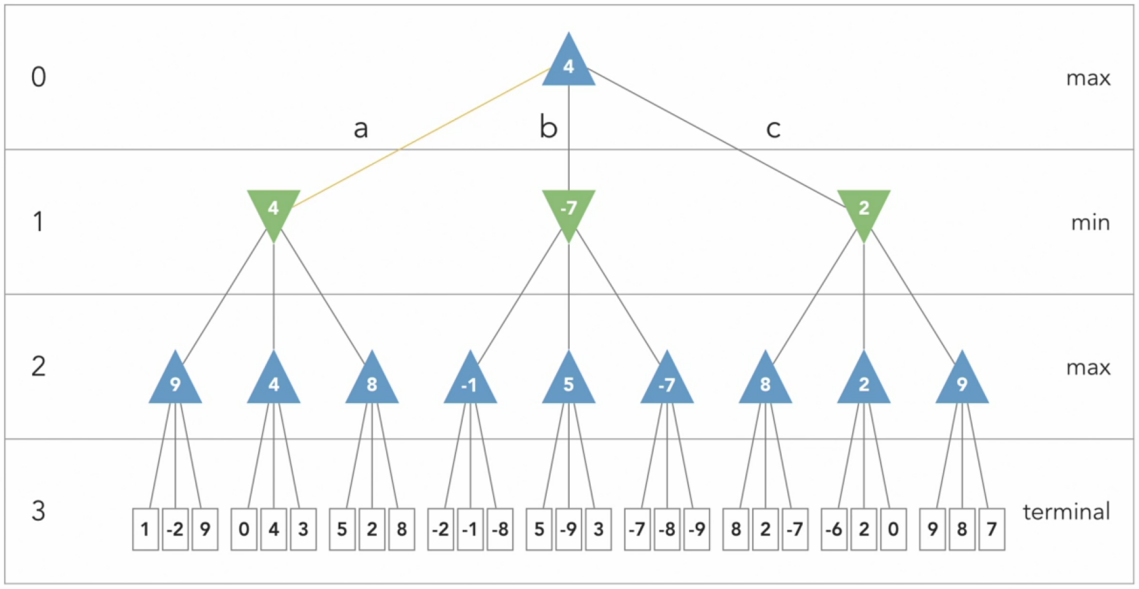


Figure (55) Minimax tree

As shown in the figure, the computer analyzes the tree starting with the terminal nodes. At level 3, it is the computer‘s turn to move, and the computer will choose the move with the highest value. Therefore, every node on level 2 is filled with the highest value of the nodes below it. On level 1, it is the human‘s turn, so the human will always pick all of the lower numbers. The computer chooses the move associated with the subtree with the greatest value because this number represents the worst scenario that could possibly occur. Because the number of possible game states in chess is incredibly large, a method was derived to speed up this search. In some cases, it can be logically deduced that one subtree will always produce an outcome that is relatively worse than another subtree. Therefore, it would not change the outcome of the game to continue evaluating that subtree, meaning some unnecessary calculations are being performed. Specifically, if it is the computer‘s turn to play, and the algorithm finds a value in a subtree that is less than or equal to the value at the top of another subtree, the computer stops searching and chooses the other subtree. This is because there exists a worse state in the subtree the algorithm is checking than the worst possible state in the other subtree already evaluated. Because the human will inevitably choose this state which is worse for the computer, the computer already knows that it should choose the opposite move. This process of pre-maturely halting an evaluation of a subtree is known as α/β Pruning, and it speeds up the computer‘s search dramatically.

9.5. Alpha-Beta Pruning

Alpha-beta pruning is a modified version of the minimax algorithm. It is an optimization technique for the minimax algorithm. As we have seen in the minimax search algorithm, the number of game states it has to examine is an exponential function in the depth of the tree. Since we cannot eliminate the exponent, but we can cut it in half. Hence there is a technique by which without checking each node of the game tree we can compute the correct minimax decision, and this technique is called pruning. This involves two threshold parameters Alpha and Beta for future expansion, so it is called alpha-beta pruning. It is also called the Alpha-Beta Algorithm. Alpha-beta pruning can be applied at any depth of a tree, and sometimes it not only prunes the tree leaves but also entire sub-trees.
The two-parameter can be defined as:

a. Alpha: The best (highest-value) choice we have found so far at any point along the path of Maximizer. The initial value of alpha is -∞.
b. Beta: The best (lowest-value) choice we have found so far at any point along the path of Minimizer. The initial value of beta is +∞.

Say it is White's turn to move, and we are searching to a depth of 2 (that is, we are considering all of White's moves, and all of Black's responses to each of those moves.) First, we pick one of White's possible moves - let's call this Possible Move #1. We consider this move and every possible response to this move by black. After this analysis, we determine that the result of making Possible Move #1 is an even position. Then, we move on and consider another of White's possible moves (Possible Move #2.) When we consider the first possible counter-move by black, we discover that playing this results in black winning a Rook! In this situation, we can safely ignore all of Black's other possible responses to Possible Move #2 because we already know that Possible Move #1 is better. We really don't care exactly how much worse Possible Move #2 is. Maybe another possible response wins a Queen, but it doesn't matter because we know that we can achieve at least an even game by playing Possible Move #1. The full analysis of Possible Move #1 gave us a lower bound. We know that we can achieve at least that, so anything that is clearly worse can be ignored.
The situation becomes even more complicated, however, when we go to a search depth of 3 or greater because now both players can make choices affecting the game tree. Now we have to maintain both a lower bound and an upper bound (called Alpha and Beta.) We maintain a lower bound because if a move is too bad we don't consider it. But we also have to maintain an upper bound because if a move at depth 3 or higher leads to a continuation that is too good, the other player won't allow it, because there was a better move higher up on the game tree that he could have played to avoid this situation. One player's lower bound is the other player's upper bound.

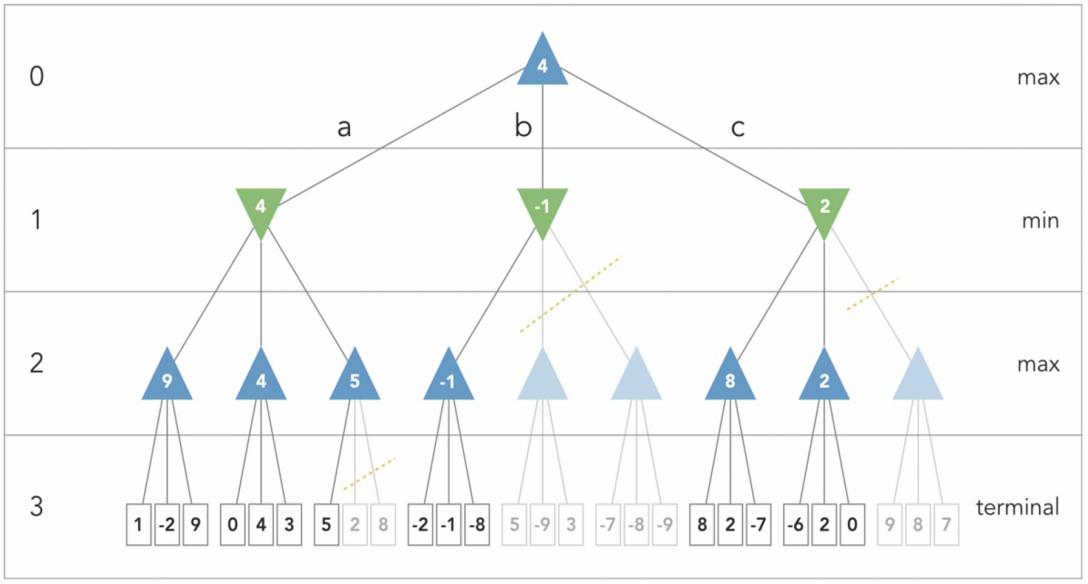


Figure (56) shows pruned branches using alpha-beta pruning

9.5.1. Pseudo-code for Alpha-beta Pruning

```
function minimax(node, depth, alpha, beta, maximizingPlayer) is
if depth ==0 or node is a terminal node then
return static evaluation of node
if MaximizingPlayer then      // for Maximizer Player
   maxEva= -infinity
   for each child of node do
   eva= minimax(child, depth-1, alpha, beta, False)
  maxEva= max(maxEva, eva)
  alpha= max(alpha, maxEva)
   if beta<=alpha
 break
 return maxEva
else                    // for Minimizer player
   minEva= +infinity
   for each child of node do
   eva= minimax(child, depth-1, alpha, beta, true)
   minEva= min(minEva, eva)
   beta= min(beta, eva)
    if beta<=alpha
  break
 return minEva
```

9.6. Move Ordering and complexity

The savings of alpha-beta can be considerable. If a standard minimax search tree has x nodes, an alpha-beta tree in a well-written program can have a node count close to the square root of x. How many nodes you can actually cut, however, depends on how well ordered your game tree is. If you always search for the best possible move first, you eliminate most of the nodes. Of course, we don't always know what the best move is, or we wouldn't have to search in the first place. Conversely, if we always searched for worse moves before the better moves, we wouldn't be able to cut any part of the tree at all! For this reason, good move ordering is very important and is the focus of a lot of the effort in writing a good chess program. As pointed out by Levin in 1961, assuming constant b moves for each node visited and search depth n, the maximal number of leaves in alpha-beta is equivalent to minimax, b ^ n. Considering always the best move first, it is b ^ ceil(n/2) plus b ^ floor(n/2) minus one. The minimal number of leaves is shown in the following table which also demonstrates the odd-even effect:

This table shows the difference between minimax and alpha-beta with optimum move ordering.

number of leaves with depth n and b = 40

depth n	b^n	$b^{\lceil n/2 \rceil} + b^{\lfloor n/2 \rfloor} - 1$
0	1	1
1	40	40
2	1,600	79
3	64,000	1,639
4	2,560,000	3,199
5	102,400,000	65,569
6	4,096,000,000	127,999
7	163,840,000,000	2,623,999
8	6,553,600,000,000	5,119,999

b: tree branching factor.

n: maximum depth.

Time complexity T(b,m) = $O(b^n)$,

space complexity $S(b,n) = O(n)$

for minimax Using alpha-beta pruning will reduce the time complexity to be $O(b^{n/2})$.

That means applying alpha-beta pruning will speed up the search, we can go twice as deep using it in the same original minimax time. The Alpha-beta pruning to a standard minimax algorithm returns the same move as the standard algorithm does, but it removes all the nodes which are not really affecting the final decision but making the algorithm slow. Hence pruning these nodes makes the algorithm fast.
As we discussed, the effectiveness of alpha-beta pruning is highly dependent on the order in which each node is examined. Move order is an important aspect of alpha-beta pruning. Even a random shuffle will be good as a starter or, Use domain knowledge while finding the best move. for Chess, try order: captures first, then threats, then forward moves, backward moves as we implemented our chess engine. There are many other move ordering techniques some techniques, some are more complicated but give very good results.

9.7. Evaluation Function

The evaluation function is the most important module, After DeepBlue lost its first match against Garry Kasparov in 1996, Grandmaster Joel Benjamin was hired by IBM to improve DeepBlue's chess knowledge, i.e. its evaluation function. It is interesting to look at all this from two different perspectives. From the developer's perspective, you have an incredibly powerful system that is notoriously difficult to program, has several bugs, and lacks a good evaluation function. From an outsider's perspective without a background in computer science, you have a chess computer whose developers claim how powerful it is, but which actually plays bad chess and loses to a chess program (a German chess program named Fritz) running on a standard home computer (DeepBlue actually lost against Fritz with White at the 1995 World Computer Chess Championship in Hong Kong, despite having magnitudes more computing power available). It is not difficult than to completely misjudged DeepBlue's true capabilities.

Nowadays the main improvement in chess engines comes in changing the evaluation function. The input for this module is a chess position, and the output is a number. If this number gives us an evaluation of 0, it means the position is equal for both players. The higher the positive number more advantages for white, and the lower the negative number, the more advantage for black. There is different data the algorithm looks to come out with an evaluation. This has been hardcoded to mimic a grand master's mind. This is fundamental for understanding chess and chess engines. Stockfish(which is considered one of the top traditional chess engines) uses a hand-crafted evaluation function. AlphaZero was developed by the artificial intelligence and research company DeepMind, which was acquired by Google. It is a computer program that reached a virtually unthinkable level of play using only reinforcement learning and self-play in order to train its neural networks. In other words, it was only given the rules of the game and then played against itself many millions of times (44 million games in the first nine hours, according to DeepMind).

When AlphaZero won against Stockfish it revolutionized the world of computational chess, not only because it was proclaimed the best chess player in history, but also because it did it in a new way, very different from how computer engines were previously programmed. AlphaZero uses neural networks to make extremely advanced evaluations of positions which negates the need to look for 70 million positions per second like Stockfish, it calculated around 80.000 positions per second. DeepMind (the company that develops AlphaZero), stated that AlphaZero reached the benchmarks to defeat Stockfish in a mere four hours. Instead of the usual alpha-beta search algorithm with domain-specific enhancements that other engines use. AlphaZero uses a general-purpose Monte Carlo tree search. But this approach was followed mainly because AlphaZero also plays shogi and go in addition to chess. We are interested in using neural networks as evaluation functions for now.

9.7.1. hand-crafted evaluation Function

A handcrafted evaluation function typically has a material balance term that usually dominates the evaluation. The conventional values used for material are Queen=9, Rook=5; Knight or Bishop=3; Pawn=1; the king is assigned an arbitrarily large value, usually larger than the total value of all the other pieces (or zero since it cannot be captured). In addition to the material term we can take into account the position of

the pieces, for example, a knight on the middle of the board is better than a knight on the edge, it can go to more squares and control much space. This was the first approach we tried as shown on the figure below.

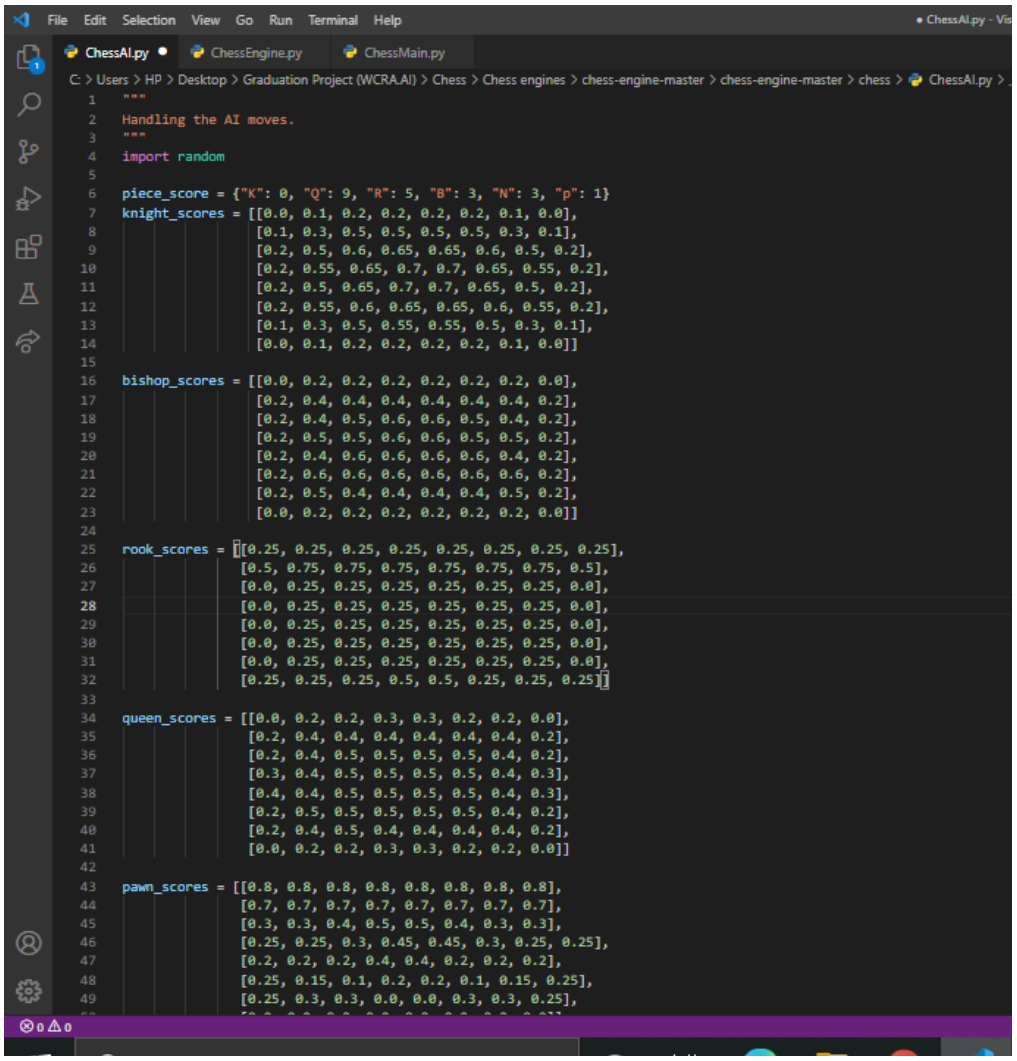


Figure (57) shows hand-coded evaluation function.
Shows the values assigned to each piece and values assigned to each square for a piece on an 8*8 matrix.

The results of that approach were not very promising, we can do a lot to make it better like adding many other parameters and tuning those parameters with some weights. Parameters like Pawn-structures, Center-control, King-safety, Piece Development, Piece Interactions, tempo….. etc (each term of these has a meaning and will add extra knowledge to evaluate a chess position).
Adding those parameters to our humble evaluation function requires lots of domain knowledge, and since we are not chess grandmasters nor planning to study chess instead of engineering (altho the first one sounds much more interesting) we need to follow different approaches.

9.7.2. ML-Based Evaluation Function

“If you program a machine, you know what it’s capable of. If the machine is programming itself, who knows what it might do? ” - Garry Kasparov

Machine learning is about extracting knowledge from data. It is a research field at the intersection of statistics, artificial intelligence, and computer science and is also known as predictive analytics or statistical learning. In the early days of “intelligent” applications, many systems used hand-coded rules of “if" and “else” decisions to process data or adjust to user input. Manually crafting decision rules is feasible for some applications, particularly those in which humans have a good understanding of the process to model. However, using hand-coded rules to make decisions has major disadvantages:

- The logic required to make a decision is specific to a single domain and task. Changing the task even slightly might require a rewrite of the whole system.

- Designing rules requires a deep understanding of how a decision should be made by a human expert.
- a hand-coded approach will fail in some problems like detecting faces in images.

If we have enough data we can train a neural network to evaluate chess positions.
Using NN to evaluate chess positions is relatively new after Yu Nasu invented the efficiently updatable neural network (NNUE) in 2018. On 6 August 2020, NNUE was for the first time ported to a chess engine, Stockfish 12 As of 2022, all of the top-rated classical chess engines, have an NNUE implementation to remain competitive.
NNUE runs efficiently on central processing units without a requirement for a graphics processing unit (GPU). Compared to deep neural network-based evaluations requiring dedicated GPUs, NNUE avoids idle times during the substantial data transfer operations between GPU and CPU required before and after each evaluation.

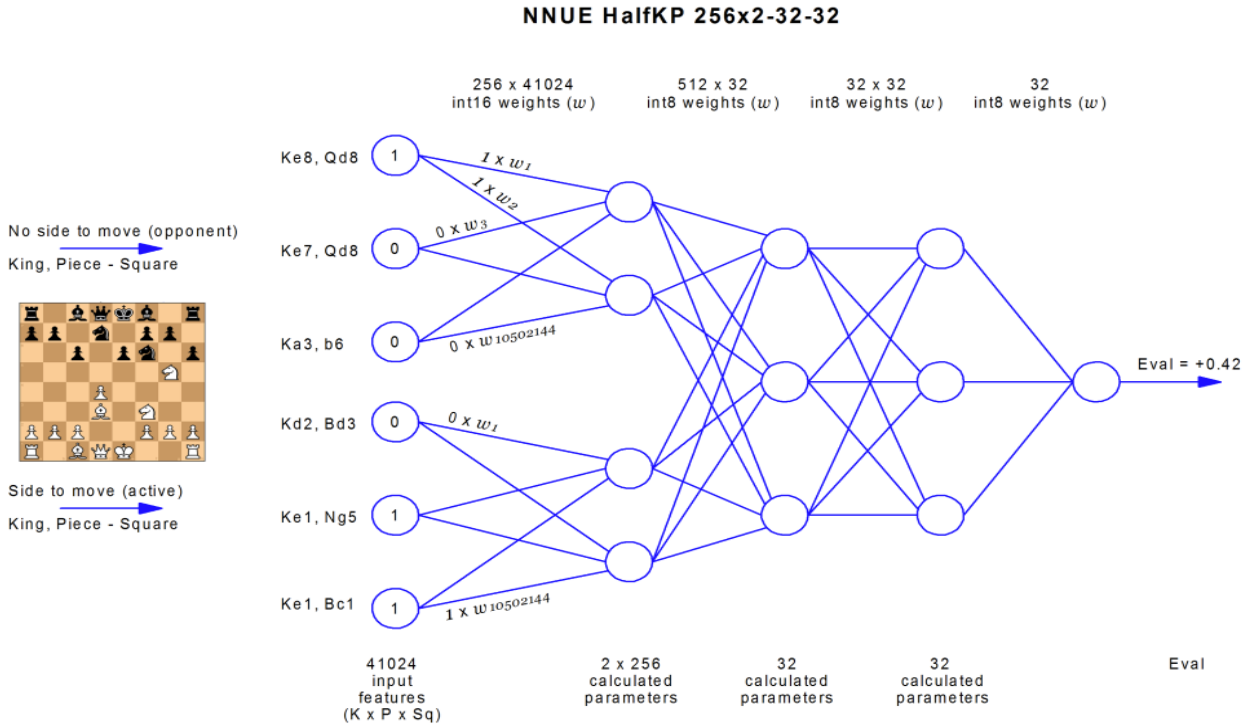


Figure (58) Shows the architecture of NNUE, it takes a chess position as input and gives an evaluation as an output

Luckily Stockfish's NNUE evaluation function is available as an open-source, all we need to do is compile it for the raspberry pi and get the .os file and we are ready to go.

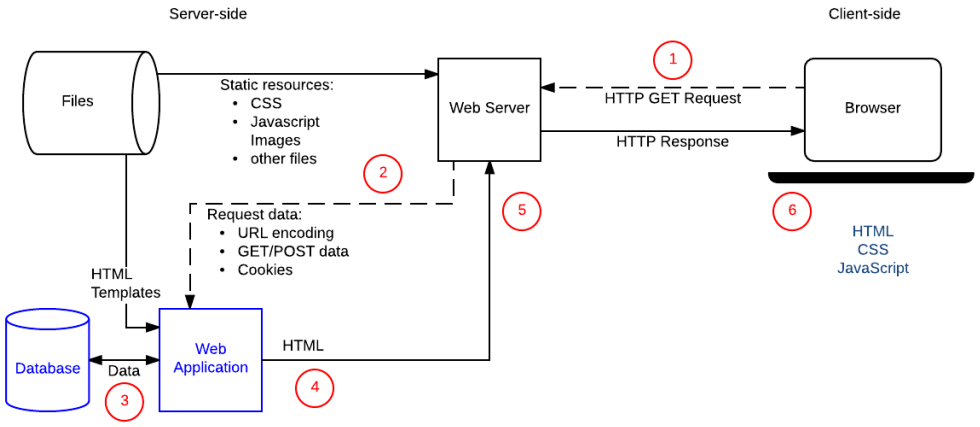
10. Website

At first, we had difficulties just understanding the rules and plans of chess and then turning it into web programming methods, but at first let's talk about how to build a website

10.1. The Road to Build a Website

In order to build a website, we have to understand what web structure looks like.

Website
|
|-- // Front-End (Client Side)
|
|--- HTML // structure of the web page and its content
|
|--- CSS // code that styles web content
|
|--- JavaScript // adds interactivity to your website.
|
|-- // Back-End (Server Side)
|
|--- WebServer // handle the requests from the client-side
|
|--- DataBase // storage for the data we deal with



10.2. Front-End (Client-Side)

Everything you see on your screen, on this website, is considered front-end. It's what the user can see when visiting a page or opening a mobile app. This layer can also be called the "Client-side".
Imagine a wall clock, where you have the hands, the time numbers, the artwork of the clock background, the frame, etc. All this would represent the front end of the clock. The gears, the springs, the battery, and all the internal mechanisms we can't see would be the back-end.
Bringing this same reasoning to software development, all the visible parts or all that affects the visible part of an application is part of the front-end. For example Menus, Buttons, Images, Fonts, and Forms, all this is part of the layer that can be seen, therefore the front-end.
A website can be built using only the front end. Sites like this are also called static websites. If you only need to display the content, without any kind of dynamic interaction, like a query to a database, you can achieve that by only using the front-end layer.

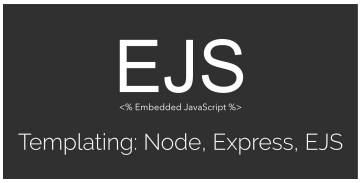
10.2.1. Front-End technologies

In general, three fundamental languages are used in front-end web development.

- HTML;
- CSS;
- JavaScript;

And we will specify which Technology we used for each part of them.

10.3. HTML (EJS Template)

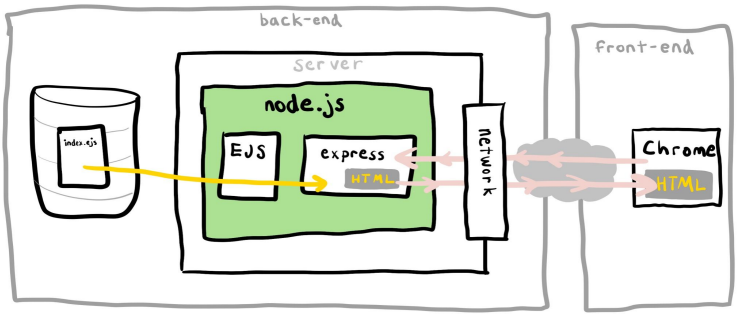


For HTML we've used something called EJS, EJS is a template system. You define HTML pages in the EJS syntax and you specify where various data will go on the page.
Then, your app combines data with the template and "renders" a complete HTML page where EJS takes your data and inserts it into the web page according to how you've defined the template.
For example, you could have a table of dynamic data from a database and you want EJS to generate the table of data according to your display rules.
It saves you from the drudgery of writing code to dynamically generate HTML based on data.
EJS also supports shared templates that can be inserted into other templates (e.g. for a common header/footer in your web pages) so you can specify that layout once and then use it in all your other templates rather than repeating that HTML over and over in all your pages. This vastly simplifies maintenance and modifications since common content is defined in one place and then used on many other pages.

10.3.1. Reasons to Use EJS instead of regular HTML syntax

The answer is that the coming era of WEB is called WEB.3, and the main target for us in this project is to consider it as an example of WEB.3, For that, I choose EJS for many reasons :

- This will make the website load faster because the EJS template is preloaded in the Back-end not as the HTML does, in regular HTML the syntax got executed at the front-end that's why sometimes it takes a while to load the website, but in EJS this won't happen because it will be already loaded at the Back-End.
- We used other technologies at the Back-End like Express.js Node.js “which we will explain later” EJS is so compatible with both of them which made it easier to use while building the website.



10.4. CSS (Bootstrap)

Bootstrap.css is a CSS framework that arranges and manages the layout of a website. While HTML works with the content and structure of a web page, CSS deals with the layout itself. For that reason, both structures need to coexist together to perform a particular action.
Bootstrap.css and its functions help a developer create a uniform look on as many pages as they need. As a result, the web developer won't have to spend hours on manual editing.
Instead of coding from scratch, you only need to refer a web page to a CSS file. Any necessary alteration can be done in that file alone.
CSS functions are not limited to text styles only as you can use them to format other aspects within a website, such as tables and image layouts.
Since CSS has a lot of declarations and selectors, memorizing all of them can take some time. Refer to our CSS cheat sheet to ease your learning process.

10.4.1. Ease of Use

First and foremost, Bootstrap is easy to learn. Due to its popularity, plenty of tutorials and online forums are available to help you get started.
One of the reasons why Bootstrap is so popular among web developers and web designers is that it has a simple file structure. Its files are compiled for easy access, and it only requires basic knowledge of HTML, CSS, and JS to modify them.
You can also use themes for popular content management systems as learning tools. For example, most WordPress themes were developed using Bootstrap, which any beginner web developer can access.
To increase the site's page load time, Bootstrap minifies the CSS and JavaScript files. Additionally, Bootstrap maintains consistency across the syntax between websites and developers, which is ideal for team-based projects.

10.4.2. Browser Compatibility

Making your website accessible via different browsers helps reduce the bounce rate and rank higher in search results. Bootstrap fulfills that requirement by being compatible with the latest versions of popular browsers.
Despite not supporting lesser-known browsers like WebKit and Gecko, websites with Bootstrap should function correctly on them as well. However, there may be limitations regarding modals and drop-downs on smaller screens.

10.4.3. Responsive Grid

Bootstrap comes with a predefined grid system, saving you from creating one from scratch. The grid system consists of rows and columns, letting you make a grid inside the existing one instead of entering media queries within the CSS file.
Additionally, Bootstrap's grid system makes the data entry process more straightforward. It contains lots of media queries, allowing you to define each column's custom breakpoints based on your web project needs. The default settings are usually more than enough. After creating a grid, you only need to add content to the containers.

The Bootstrap grid system has two container classes to better accommodate both desktop and mobile-based projects – a fixed container (.container) and a fluid container (.container-fluid).
The first container class provides a fixed-width container, while the latter offers a full-width container capable of adjusting your project to all screen sizes.

10.4.4. Bootstrap Image System

Bootstrap handles the image display and responsiveness with its predefined HTML and CSS rules. Adding the .img-responsive class will automatically resize images based on the users' screen size. This will benefit your website's performance, as reducing image sizes is part of the site optimization process. Bootstrap also provides additional classes like .img-circle and .img-rounded, which help to modify the images' shape.

10.4.5. Bootstrap Documentation

Bootstrap provides documentation for developers who want to learn to use this framework for the first time. Here are several topics you can find on the Bootstrap documentation page:

- Content – covers the precompiled Bootstrap source code.
- Browsers and devices – lists all the supported web and mobile browsers and mobile-based components.
- JavaScript – breaks down various JS plugins built on jQuery.
- Theming – explains built-in Sass variables for easier customization.
- Tools – teaches you how to use Bootstrap's npm scripts for various actions.
- Accessibility – covers Bootstrap's features and limitations regarding structural markup, components, color contrast, content visibility, and transition effects.

10.5. JavaScript (Vanilla JS)

JavaScript is a scripting or programming language that allows you to implement complex features on web pages — every time a web page does more than just sit there and display static information for you to look at — displaying timely content updates, interactive maps, animated 2D/3D graphics, scrolling video jukeboxes, etc. — you can bet that JavaScript is probably involved. It is the third layer of the layer cake of standard web technologies, two of which (HTML and CSS) we have covered in much more detail in other parts of the Learning Area.

10.5.1. The Definition of Vanilla JavaScript

Vanilla JS is a name to refer to using plain JavaScript without any additional libraries like jQuery back in the day. People use it as a joke to remind other developers that many things can be done nowadays without the need for additional JavaScript libraries.

10.5.2. Reasons Why JS in a vanilla form

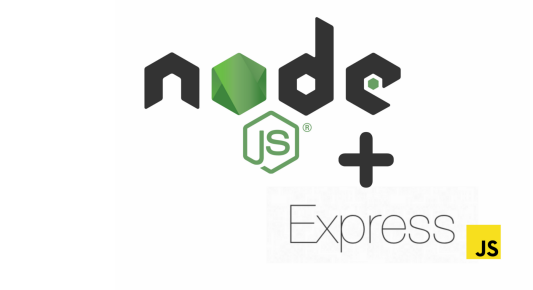
The Chess website is so complicated in a way that it would be so difficult for frameworks to handle, but in Vanilla JS it will give us more control over the syntax so we can write the chess rules & method in OOP without any difficulties.
In a moment we will explain in detail what we did in our structure

10.6. Back-End (Client-Side)

refers to the server-side development. It focuses on databases, scripting, and website architecture. It contains behind-the-scenes activities that occur when performing any action on a website. It can be an account login or making a purchase from an online store. Code written by back-end developers helps browsers communicate with database information.
Most modern websites are dynamic, meaning webpage content is generated on-the-fly. A dynamic page contains one or more scripts that run on the web server each time the page is accessed. These scripts generate the content of the page, which is sent to the user's web browser. Everything that happens before the page is displayed in a web browser is part of the backend.
The back-end is all of the technology required to process the incoming request and generate and send the response to the client. This typically includes three major parts:

- The server: This is the computer that receives requests.
- The app: This is the application running on the server that listens for requests, retrieves information from the database, and sends a response.
- The database: Databases are used to organize and persist data

10.7. The App (Node.js, Express.js):



The server runs an app that contains logic about how to respond to various requests based on the HTTP verb and the Uniform Resource Identifier (URI). The pair of an HTTP verb and a URI is called a route and matching them based on a request is called routing.
Some of these handler functions will be middleware. In this context, middleware is any code that executes between the server receiving a request and sending a response. These middleware functions might modify the request object, query the database, or otherwise process the incoming request. Middleware functions typically end by passing control to the next middleware function, rather than by sending a response.
Eventually, a middleware function will be called that ends the request-response cycle by sending an HTTP response back to the client.
Often, programmers will use a framework like Express or Ruby on Rails to simplify the logic of routing. For now, just think that each route can have one or many handler functions that are executed whenever a request to that route (HTTP verb and URI) is matched.

11. Node.js

Node.js is a platform built on Chrome's JavaScript runtime for easily building fast and scalable network applications. Node.js uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices.
Node.js is an open-source, cross-platform runtime environment for developing server-side and networking applications. Node.js applications are written in JavaScript and can be run within the Node.js runtime on OS X, Microsoft Windows, and Linux.
Node.js also provides a rich library of various JavaScript modules which simplifies the development of web applications using Node.js to a great extent.

11.0.1. Motivation to Use Node.js for Back-end

There are many languages that we can build a backend with like PHP, js, and even python for that when i got to choose I found that I have good experience in js so why

11.1. The Server(Express.js)

A web server is designed to serve HTTP Content. App Server can also serve HTTP Content but is not limited to just HTTP. It can be provided with other protocol support such as RMI/RPC.
Web Server is mostly designed to serve static content, though most Web Servers have plugins to support scripting languages like Perl, PHP, ASP, JSP, etc. through which these servers can generate dynamic HTTP content.
For this part, we didn’t have to do anything because the host of our website did all the work to build the web server for us.

11.1.1. What is Express.js and Why did we choose it to handle our requests?

Express is a node js web application framework that provides broad features for building web and mobile applications. It is used to build a single page, multipage, and hybrid web application.
It's a layer built on the top of the Node js that helps manage servers and routes.
The reasons choose it is

- Express was created to make APIs and web applications with ease,
- It saves a lot of coding time almost by half and still makes web and
- Mobile applications are efficient.
- Another reason for using express is that it is written in javascript as javascript is an easy language even if you didn't have it before.
- knowledge of any language. Express lets so many new developers enter the field of web development.

11.1.2. The reasons behind creating an express framework for node js.

- Time-efficient
- Fast
- Economical
- Easy to learn
- Asynchronous Advantages of Using Express With Node.js
- Express is Opinionated, and we can customize it.
- For request handling, we can use Middleware.
- A single language is used for frontend and backend development.
- Express is fast to link it with databases like MySQL, MongoDB, etc.
- Express allows dynamic rendering of HTML Pages based on passing arguments to templates (like EJS that is used for HTML).

11.2. The DataBase(MongoDB)

11.2.1. Web database definition

A web database is essentially a database that can be accessed from a local network or the internet instead of one that has its data stored on a desktop or its attached storage. Used for both professional and personal use, they are hosted on websites and are software as service (SaaS) products, which means that access is provided via a web browser.
One of the types of web databases that you may be more familiar with is a relational database. Relational databases allow you to store data in groups (known as tables), through their ability to link records together. It uses indexes and keys, which are added to data, to locate information fields stored in the database, enabling you to retrieve information quickly.
To paint a picture, just think about when you shop online and want to have a look at a specific product. Typing in keywords such as “black dress” enables all the black dresses stored on the website to appear right on the very browser you are looking on because the information “black” and “dress” are stored in their database entries.

11.2.2. Some advantages of using a web database include:

1. Web database applications can be free or require payment, usually through monthly subscriptions. Because of this, you pay for the amount you use. So whether your business shrinks or expands, your needs can be accommodated by the amount of server space. You also don’t have to fork out the cost of installing an entire software program.
2. The information is accessible from almost any device. Having things stored in a cloud means that it is not stuck to one computer. As long as you are granted access, you can technically get a hold of the data from just about any compatible device.
3. Web database programs usually come with their own technical support team so your IT department folks can focus on other pressing company matters.
4. It’s convenient: web databases allow users to update the information so all you have to do is to create simple web forms.

11.3. Types of Databases

There are two types of DataBases:

- SQL
- NoSQL

11.4. SQL vs NoSQL: Five Main Differences

SQL is the programming language used to interface with relational databases. (Relational databases model data as records in rows and tables with logical links between them). NoSQL is a class of DBMS that is non-relational and generally does not use SQL.
There are five practical differences between SQL and NoSQL:

1. Language
2. Scalability
3. Structure
4. Properties
5. Support and communities

11.4.1. Language

SQL has been around for over 40 years, so it is recognizable, documented, and widely used. Safe and versatile, it's particularly well suited for complex queries. However, SQL restricts the user to work within a predefined tabular schema, and more care must be taken to organize and understand the data before it is used.
The dynamic schemata of NoSQL databases allow the representation of alternative structures, often alongside each other, encouraging greater flexibility. There is less emphasis on planning, greater freedom when adding new attributes or fields, and the possibility of varied syntax across databases. As a group, however, NoSQL languages lack the standard interface which SQL provides, so more complex queries can be difficult to execute.
Though there are many dialects of SQL, all share a common syntax and almost-identical grammar. When querying relational databases, fluency in one language translates to proficiency in most others. On the other hand, there is very little consistency between NoSQL languages, as they concern a diverse set of unrelated technologies. Many NoSQL databases have a unique data manipulation language constrained by particular structures and capabilities.

11.4.2. Scalability

Most SQL databases can be scaled vertically, by increasing the processing power of existing hardware. NoSQL databases use a master-slave architecture that scales better horizontally, with additional servers or nodes. These are useful generalizations, but it's important to note:

- SQL databases can be scaled horizontally as well, though sharding or partitioning logic is often the user's onus and not well supported.
- NoSQL technologies are diverse and while many rely on the master-slave architecture, options for scaling vertically also exist.
- Savings made using more efficient data structures can overwhelm differences in scalability; the most important is to understand the use case and plan accordingly.

11.4.3. Structure

SQL database schemata always represent relational, tabular data, with rules about consistency and integrity. They contain tables with columns (attributes) and rows (records), and keys have constrained logical relationships.
NoSQL databases need not stick to this format, but generally fit into one of four broad categories:

- Column-oriented databases transpose row-oriented RDBMSs, allowing efficient storage of high-dimensional data and individual records with varying attributes.
- Key-Value stores are dictionaries that access diverse objects with a key unique to each.
- Document stores hold semi-structured data: objects which contain all of their own relevant information, and which can be completely different from each other.
- Graph databases add the concept of relationships (direct links between objects) to documents, allowing rapid traversal of greatly connected data sets.

11.4.4. Properties

At a high level, SQL and NoSQL comply with separate rules for resolving transactions. RDBMSs must exhibit four “ACID” properties:

- Atomicity means all transactions must succeed or fail completely. They cannot be partially complete, even in the case of system failure.
- Consistency means that at each step the database follows invariants: rules which validate and prevent corruption.
- Isolation prevents concurrent transactions from affecting each other. Transactions must result in the same final state as if they were run sequentially, even if they were run in parallel.
- The durability makes transactions final. Even system failure cannot roll back the effects of a successful transaction.

NoSQL technologies adhere to the “CAP” theorem, which says that in any distributed database, only two of the following properties can be guaranteed at once:

- Consistency: Every request receives the most recent result or an error. (Note this is different than in ACID)
- Availability: Every request has a non-error result, regardless of how recent that result is.
- Partition tolerance: Any delays or losses between nodes will not interrupt the system’s operation.

11.4.5. Support and communities

SQL databases represent massive communities, stable codebases, and proven standards. Multitudes of examples are posted online and experts are available to support those new to programming relational data.
NoSQL technologies are being adopted quickly, but communities remain smaller and more fractured. However, many SQL languages are proprietary or associated with large single-vendors, while NoSQL communities benefit from open systems and concerted commitment to onboarding users.
SQL is available to most major platforms, from operating systems to architectures and programming languages. Compatibility varies more widely for NoSQL, and dependencies need to be investigated more carefully.
SQL is the programming language used to interface with relational databases. (Relational databases model data as records in rows and tables with logical links between them). NoSQL is a class of DBMs that are non-relational and generally do not use SQL.

11.5. MongoDB

MongoDB is a document-oriented NoSQL database used for high-volume data storage. Instead of using tables and rows as in the traditional relational databases, MongoDB makes use of collections and documents. Documents consist of key-value pairs which are the basic unit of data in MongoDB. Collections contain sets of documents and functions which are the equivalent of relational database tables. MongoDB is a database that came to light around the mid-2000s.

11.5.1. MongoDB Features

1. Each database contains collections which in turn contain documents. Each document can be different with a varying number of fields. The size and content of each document can be different from each other.
2. The document structure is more in line with how developers construct their classes and objects in their respective programming languages. Developers will often say that their classes are not rows and columns but have a clear structure with key-value pairs.
3. The rows (or documents as called in MongoDB) don't need to have a schema defined beforehand. Instead, the fields can be created on the fly.
4. The data model available within MongoDB allows you to represent hierarchical relationships, to store arrays, and other more complex structures more easily.
5. Scalability – MongoDB environments are very scalable. Companies across the world have defined clusters with some of them running 100+ nodes with around millions of documents within the database.

11.5.2. Reasons for Choosing MongoDB

- In our project, we had to deal with NoSQL data which is the chess notation.
- We have to have more than one path to the same cluster in the DataBase one for the website and one for the Robot, and MongoDB gives you this option.
- MongoDB is so integrated with express and node.js and this was so helpful while writing the back-end app
- Mongo is a live DataBase which means it’s always running even if the website wasn’t on the domain yet which was so helpful during the building process.

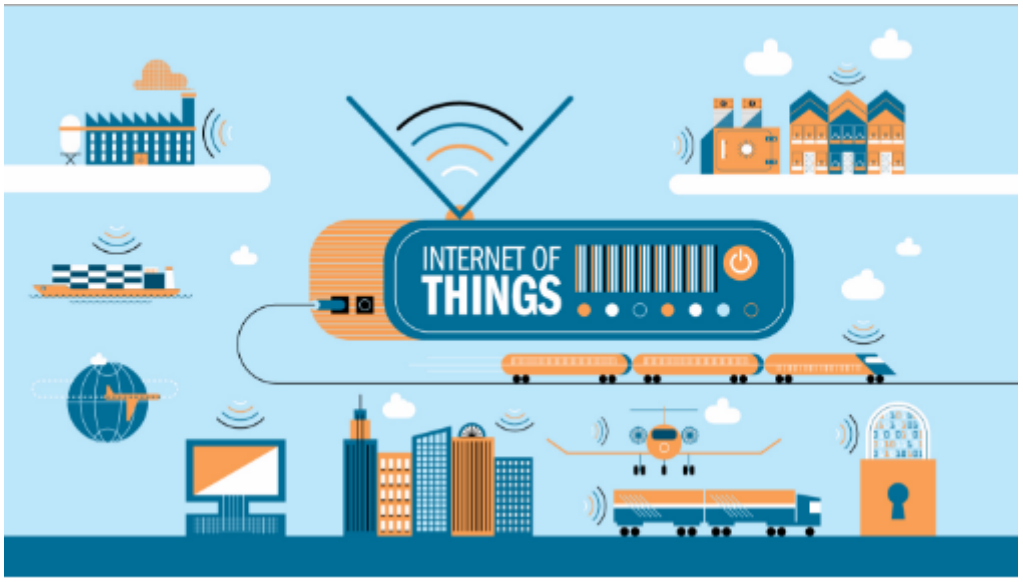
12. IoT

12.1. Introduction to IoT

Internet of Things has been in development for decades and just in 1999, it is proposed at a conference. The first Internet appliance, for example, was a Coke machine at Carnegie Melon University in the early 1980s. The programmers might connect to the machine more to the internet, check the status of the machine and conclude whether or not there would be a cold drink pending on them, should they decide to make the trip down to the machine.

12.2. Defining the Internet of Things

The Internet of Things (IoT) is a scenario in which objects, animals or people are provided with single identifiers and the capability to automatically transfer data more to a network without requiring human-to-human or human-to-computer communication. IoT has evolved from the meeting of wireless technologies, micro-electromechanical systems (MEMS) and the internet



12.3. Reasons to choose Raspberry Pi

The Raspberry Pi is a popular choice when developing IoT products. It offers a complete Linux server with a tiny platform at an incredibly low price
The Raspberry Pi Foundation provides Debian distributions for download, and promotes Python as the main programming language, with support for Java, C, PHP, C++, etc.

12.4. IOT Design Methodology

All web application is developed natively in Java Programming Language. It includes java technologies similar to web services. Additional technologies like bootstrap, java script, jQuery, etc are used to handle UI and client-side validations.
Five steps are used in web applications
· Installing Apache Web server
· Create My SQL or database system
· Developed web application For the GUI (Graphical User Interface)
· Write lots of PHP, JAVA script, CSS, and Python Programs for the Web Application

· Host Web application on our Web server

12.5. Raspberry pi and Web application connection

We use python language with pymongo library to connect to MongoDB database

12.5.1. Elaboration on python language

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985-1990. Like Perl, Python source code is also available under the GNU General Public License (GPL).

12.5.2. Motivation for the MongoDB database

MongoDB is a NoSQL cross-platform document-oriented database. It is one of the most popular databases available. MongoDB is developed by MongoDB Inc. and is published as free and open-source software
A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents. MongoDB stores documents in collections. Collections are analogous to tables in relational databases and documents to rows.

In our project, we use the Internet of Things to connect the Raspberry Pi with the web application.
The camera acts as a sensor that takes pictures of the bord and compares them to extract the data (in chess notation) and Raspberry Pi sends this data to the web application. The data is stored in a database called MongoDB and can return data from the database and send it to the robotic arm to detect the motion.
In Raspberry Pi using python language with PyMongo library to connect to MongoDB database.

12.5.3. Raspberry to Web Communication

PyMongo is a Python module for working with MongoDB in Python.
In PyMongo library
To insert data create a dictionary object and insert data into the database.The method used to insert data

insert_one() or insert_many()

```
def AddNewMove(HumanMove,moves):
    moves.insert_one({'move':{"from":HumanMove[0]+HumanMove[1],"to":HumanMove[2]+HumanMove[3]}})
```

```
client=SetUp()
moves,MP = AcessDB(client)
AddNewMove('a2a4',moves)
print('data is set......')
```

OutputOutput :

```
Connected successfully!!!
this is db : =====> Database(MongoClient(host=['moves-shard-00-02.yxjpv.mongodb.net:27017', 'moves-shard-00-01.yxjpv.mongodb.net:27017', 'moves-shard-00-00.yxjpv.mongodb.net:27017'], document_class=dict, tz_aware=False, connect=True, retrywrites=True, w='majority', authsource='admin', replicaset='atlas-r983fz-shard-0', tls=True), 'myFirstDatabase')
data is set......
```

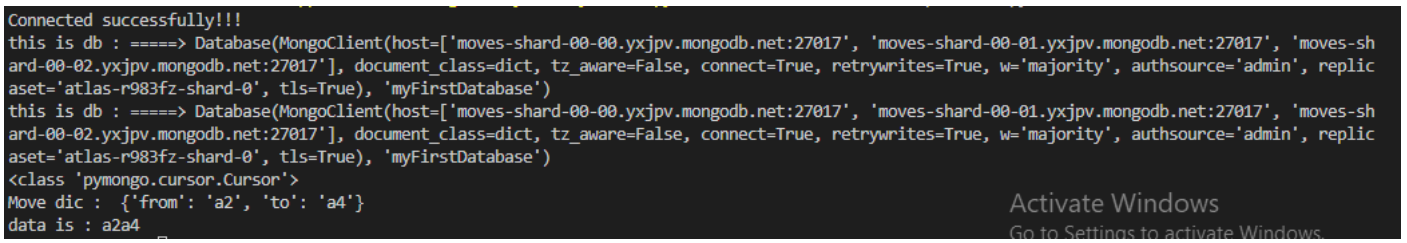

After insert data in database ,To find and return all document inside the collection.Method used is:
find() and can return the last record in database by method:

find().limit(1).sort([('$natural',-1)

```
def ReadLastMove(lastAddedmoveByRaspberry,client):
    while True :
        moves , _ =AcessDB(client)
        name=''
        name=moves.find().limit(1).sort([('$natural',-1)])
```

```
lastmove=ReadAnyLastMove()
```

OutputOutput:



12.6. IoT Future Improvements

The main purpose of the Internet of Things(IoT) is to collect and transmit data. According to Business Insider, there will be more than 64 billion IoT devices by 2025, Up from about 9 billion in 2017. All these IoT devices generate a lot of data that needs to be collected and mined for actionable results. This is where Artificial Intelligence comes into the picture. The Internet of Things is used to collect and handle the huge amount of data that is required by Artificial Intelligence algorithms. In turn, these algorithms convert the data into useful actionable results that can be implemented by IoT devices. The Internet of Things (IoT) is a concept that connects physical or virtual objects to the internet. The technology very often used is the sensor (a camera in our case), allowing us to link a physical object such as a watch, a drone or a robot arm as in our case, to the internet. If for a long time the few objects connected to the Internet were the telephone and the computer, this is no longer the case today and every year new types of objects incorporating IOT technology are born. and with the power of wireless communication technology and web applications, (IoT) becomes more prevalent than ever before. Wireless communication between Electronic devices and modules is very important, to make them ‘Fit’ in the World of the Internet of Things. The HTTP protocol and HTML language have made it possible to transfer the Data anywhere in the world, over the web. Raspberry pi powered us with the tools needed to transmit data.
According to several studies, the use of IoT is expected to generate 4.4 trillion GBytes in 2020, and this figure is expected to increase in subsequent years. In addition, this data must be read, exploited, and transmitted within specific time slots, so, as you might have guessed, the major challenge in the field of the Internet of Things is to be able to exploit a huge amount of data, hence the use of big data. This data can be used to improve IoT solutions and enable real-time analysis. After Collecting data generated by IoT sensors, Storing this data in files within the database, analyzing data, and putting it in a suitable form. Here's the role of AI came, to make use of these collected data and train AI models.
Since Web application simulates arm-to-arm play. If there are many robot arms, a server can be used to store data (make a database) collected by the WCRA chess engine and data collected by different cameras to train models for evaluating chess positions giving better performance and training camera models to also detect chess piece type, not the color only.

13. References

1. Google *TensorFlow* [online] may 2022 https://www.tensorflow.org/
2. Google *TensorFlow* [online] may 2022 https://www.tensorflow.org/install
3. Google *TensorFlow* [online] may 2022 https://www.tensorflow.org/learn
4. Mathworks *Mathworks help* [online] jan, 2022https://www.mathworks.com/help/
5. Alexlenail *NN-SVG* [online] Jan 15 2019, june 2022 https://github.com/alexlenail/NN-SVG.git
6. MDN (Mozilla Developer Network)
7. stack overflow
8. Figma.com
9. Npmjs.com
10. Dominik Klein, September 28, 2021, “Neural Networks for Chess”.
11. Andreas C. Müller, Sarah Guido, “Introduction to Machine Learning with Python”
12. Javatpoint [online] Mini-Max Algorithm https://www..com/mini-max-algorithm-in-ai
13. Researchgate.net