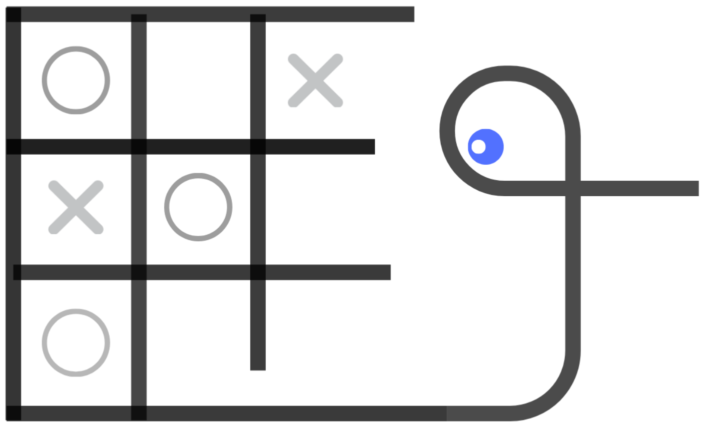
# Project Report of Tic-Tac-Toe robot



Li ZiHe

March 2025

kimi.li2008@outlook.com

# Contents

**Abstract**

This report aims to summarize the work I have done in the last month. I built a PVE system that can play against humans. The system contains a carefully designed container for the development board and the wiring, a robotic arm , a camera, and a user interface.

**keywords**: arduino, opencv, python, tkinter

## 1 Introduction

These pictures show how the tic-tac-toe robot looks like.

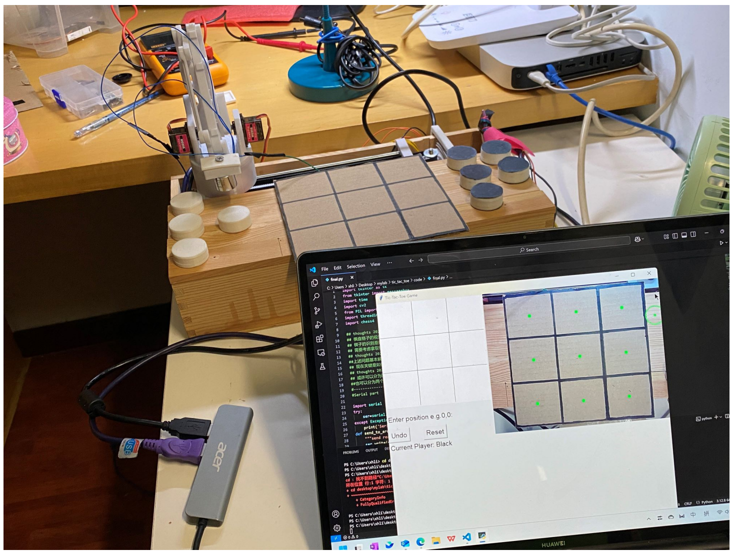


Figure 1: Robot in operation

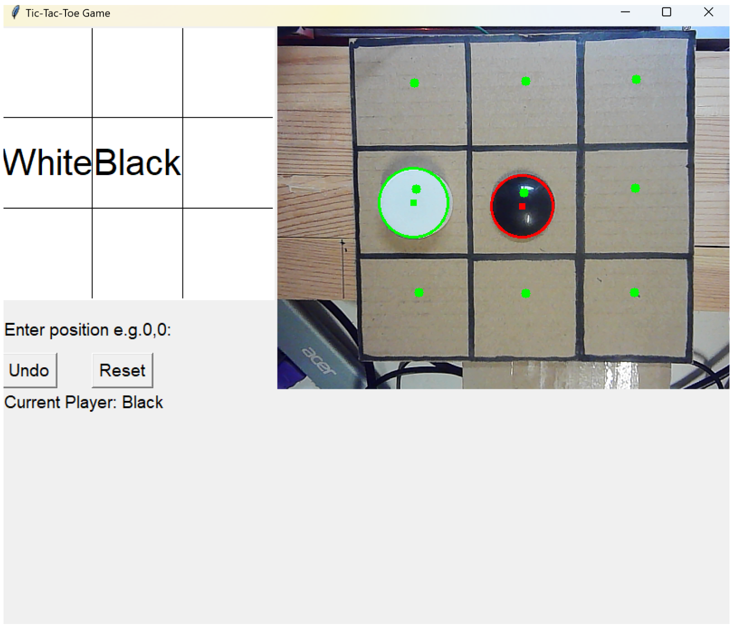


Figure 2: The GUI demo

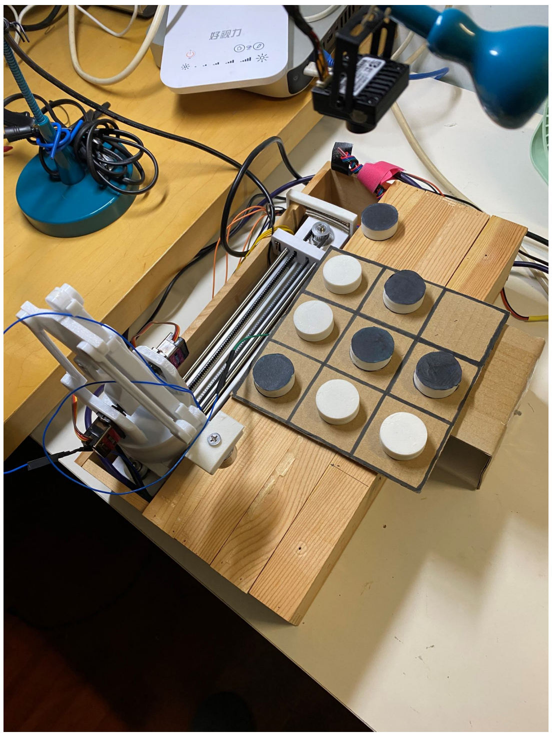


Figure 3: A view of the system

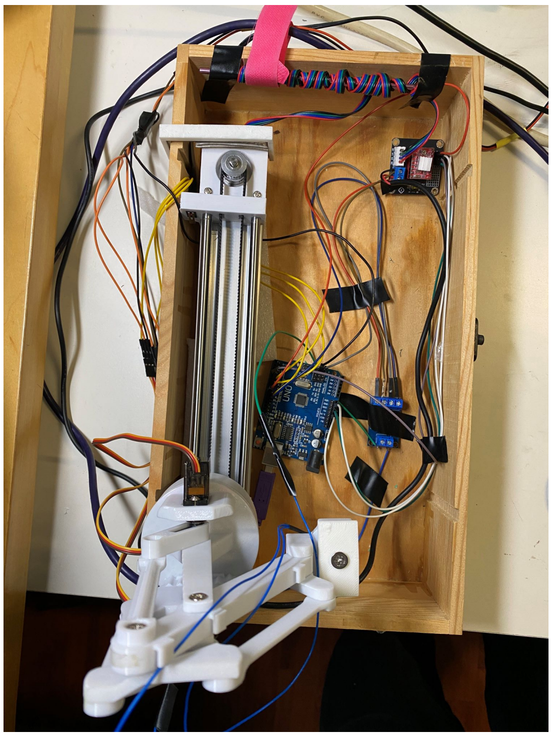


Figure 4: Inside of the box

**Here is the demo video**
**https://github.com/Kimili2008/Mylab/Tic_tac_toe-robot/display_video.mp4**

**This is the link to my project on github,including source code, development record, chessbot design stl files, and detailed explanations of the problems and solutions to them.**
https://github.com/Kimili2008/project

I started to have the idea of building a chess robot on April 1, 2023, during the time when I was rambling in a shopping mall. Then I saw the Chinese chess robot in the mall. I was very excited and made up my mind to develop one of my own, but at that time my knowledge in coding was not sufficient enough to support this idea, so I put off the plan.

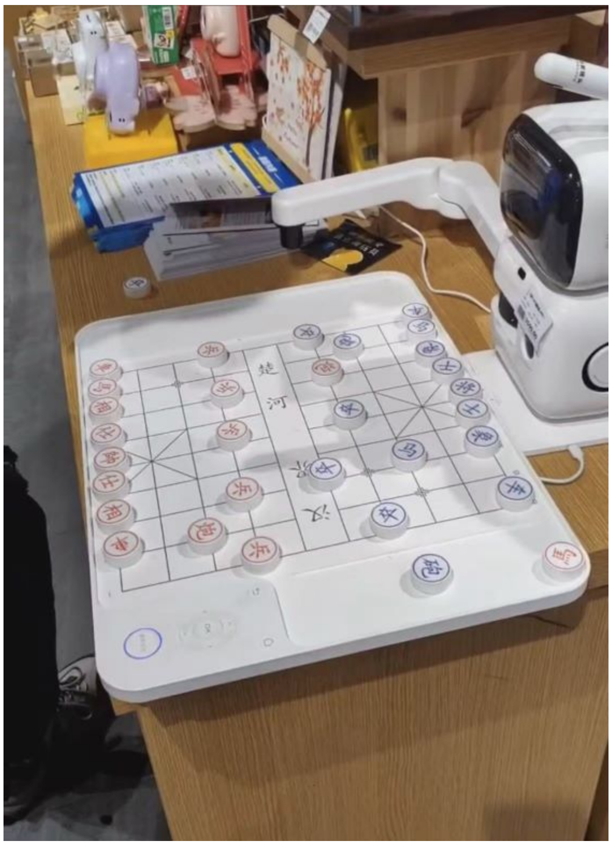


Figure 5: The picture of the robot taken on April 1st, 2023

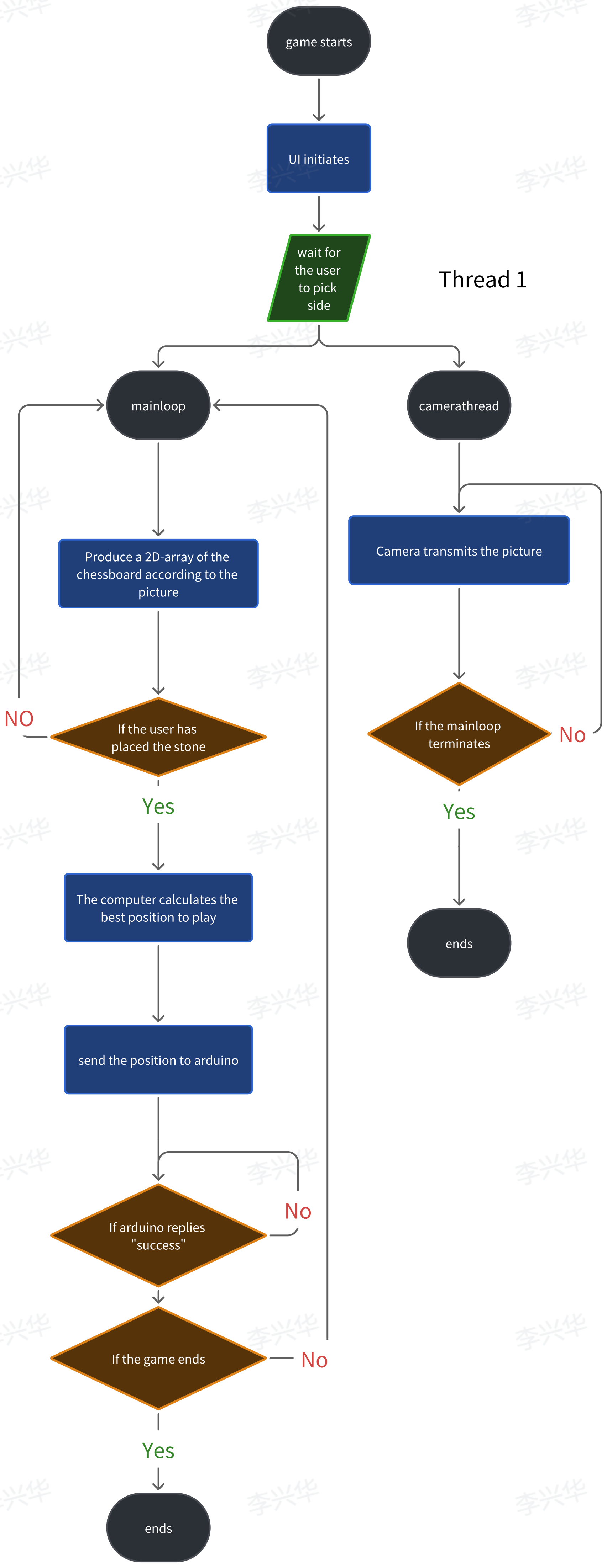
Having enough time in 2025, the development begins. And after researching, I accidentally found out that the present project happened to be E task in *CNUEC* 2024 (Chinese National Undergraduate Electronics Design Contest)

Link to this contest: https://baike.baidu.com/item/CNUEC

Although, as a high school student, I cannot directly participate in this contest, It is still full of fun to try something new. Everyday, I spent approximately two and a half hours on this project. The development,starting from February the fifth to the last commit on Marth the second, took me a long time to finish.

## 2 The System Diagram

This is the workflow of the robot and the flowchart of the whole system.


game starts
UI initiates
wait for the user to pick side
Thread 1
mainloop
camerathread
Produce a 2D-array of the chessboard according to the picture
Camera transmits the picture
NO
If the user has placed the stone
If the mainloop terminates
No
Yes
Yes
The computer calculates the best position to play
ends
send the position to arduino
If arduino replies "success"
No
If the game ends
No
Yes
ends

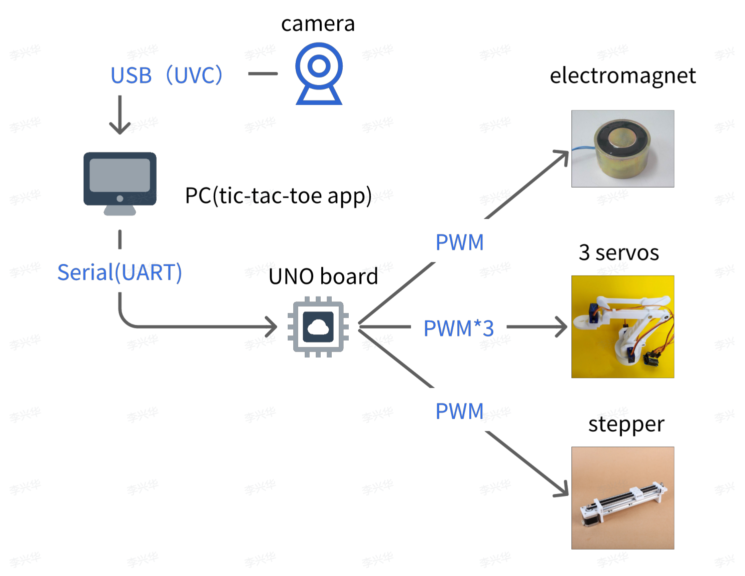


Figure 6: The hardware system

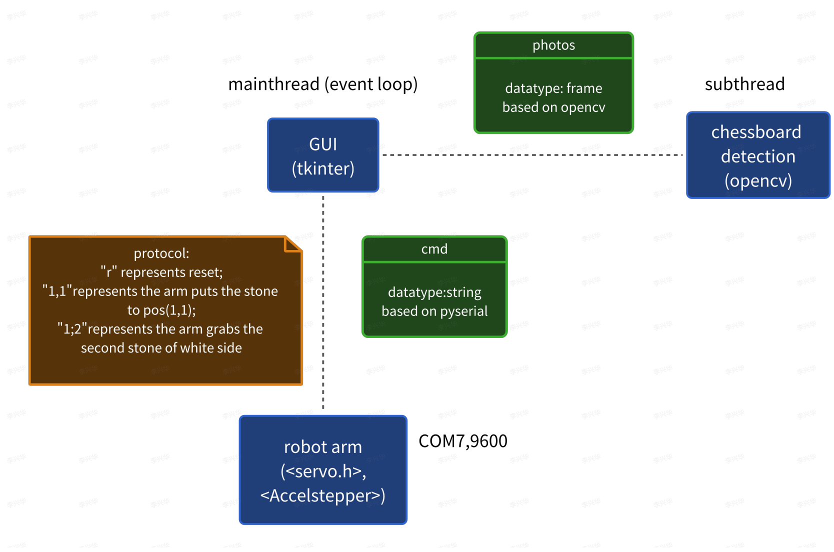

photos
datatype: frame based on opencv
mainthread (event loop)
subthread
GUI (tkinter)
chessboard detection (opencv)
protocol:
"r" represents reset;
"1,1"represents the arm puts the stone to pos(1,1);
"1;2"represents the arm grabs the second stone of white side
cmd
datatype:string based on pyserial
robot arm (<servo.h>, <Accelstepper>)
COM7,9600

Figure 7: the software block diagram

## 3 Design walkthrough

### 3.1 UNO board

#### 3.1.1 Introduction of the UNO board

Info of the board: https://docs.arduino.cc/hardware/uno-rev3

The UNO board features:
-14 GPIOs (digital input/output pins)
6 of which can be used as PWM outputs. digital input is the sensor input which only gives 1 or 0, marked by high voltage or low voltage.
PWM(Pulse width Modulation) is a controller which can change the frequency of the outputted waves without altering the frequency of the inputted wave. For instance, a 1kHZ output can be modulated to a 1 MHZ output, and the servo can run faster without

altering the frequency from the program.
- 6 analog inputs
analog input is the sensor input which gives a wave, interpreting the status of the sensor .
-It has a 16 MHZ ceramic resonator.
A 16 MHz ceramic resonator is a type of passive component used in electronic circuits to generate or stabilize oscillation frequencies. It functions similarly to a quartz crystal but uses ceramic materials to achieve piezoelectric properties, which allow it to oscillate at a specific frequency when an electrical signal is applied.
**In my design, PWM is used to control the servos and stepper.**

#### 3.1.2 Arduino SW

The stepper and servo control libraries in Arduino are well encapsulated.
- servo lib https://docs.arduino.cc/libraries/servo/
- stepper lib https://reference.arduino.cc/reference/en/libraries/accelstepper/
- pwm function(embedded) https://reference.arduino.cc/reference/en/language/functions/analog-io/analogwrite
- serial lib(embedded)
https://reference.arduino.cc/reference/en/language/functions/communication/serial/

#### 3.1.3 How does UNO communicate with PC?

The default transmission protocol in Arduino is UART(Universal Asynchronous Receiver/ Transmitter)(interrupt-driven). The data is transmitted and received at different stages in this protocol.
The data is first put into the TDR(transmission data register) by CPU, then fetched into the TSR(transmission shift register), translated into electrical signal, and sent out through the TX pin. When it is finished, register will send out a interrupt message to CPU. The CPU writes data again. The receive mode works in a similar way.

### 3.2 GUI software design

The GUI design's structure is based on tkinter. The tkinter is an event-driven lib. The Language model helps me to produce the structure of the program.
**These are the important methods of the class (Tic-tac-toe app)**

- __init__ (initiates the board)
- drawboard(self) (update the GUI)
- undo_move(self) (undo the move)
- reset_board(self) (reset the board)
- ai_turn(self) (the ai checks the camera, decides where to play, and sends signal to the robotic arm)
- select_black(self) (when the user clicks the black button event, the event triggers this method)
- select_white(self) (when the user clicks the black button event, the event triggers this method)

- update_camera_feed(self) (fetch the picture, and get the 2D-array chess board)

**These are the important attributes of the class.**

- self.board : list (records the current board)
- self.appstatus : str (FSM) (e.g. "app-waiting")
- self.whitelastmove : tuple (record the white's last move)(used in undo operation)
- self.blacklastmove : tuple (same as above)
- self.quenelist : str (events of cam stored in this attribute)(when =="cam_send",the camera sends the data to computer)
- self.currentplayer : int (records the current player)(1 represents white)(black represents 2)

## 3.3 Important algorithms

### 3.3.1 chessboard algorithm

To play the tic-tac-toe game, I designed a simple algorithm.

```
def ai_move(board:list):
        """
        Give the AI best response to the next location

        parameters:
            board (list of list of int): a 3x3 list represents the board,
            0 represents empty blocks,
            1 represents the white,
            2 represents the black.

        return:
            tuple: the AI choice (row, col)
        """
        # calculate the number of chess currently in board
        white_count = sum(row.count(1) for row in board)
        black_count = sum(row.count(2) for row in board)

        # decide who plays
        current_player = 1 if white_count < black_count else 2

        # check if there's a winning move
        def check_win_move(player):
            for row in range(3):
                for col in range(3):
                    if board[row][col] == 0:
                        # simulate to place the stones
                        board[row][col] = player
                        if check_winner(board, player)!=False:
                            board[row][col] = 0
                            return (row, col)
                        board[row][col] = 0
            return None

        # check the win move
        win_move = check_win_move(current_player)
        if win_move:
            return win_move

        # check if the opponent has the winning move
        opponent = 1 if current_player == 2 else 2
        block_move = check_win_move(opponent)
        if block_move:
            return block_move

        # choose a random space
        from random import choice
        empty_spots = [(r, c) for r in range(3) for c in range(3)
        if board[r][c] == 0]
        return choice(empty_spots)
```

I've thought about the min-max algorithm, but eventually chose a simpler algorithm because for easy game like tic-tac-toe, simple algorithm can produces the same result at a faster speed.

- checks who is playing right now.
- checks if the player has a chance of winning.If do, play it.
- checks if the opponent has a chance of winning. If do, play it.
- else, choose a random place to play.

### 3.3.2 Grayscaling

In the program "chess4.py", line 23, I used the function"cvtcolor" to convert the coloured picture to the gray version.
The algorithm follows this formula

$$Y = 0.299R + 0.587G + 0.114B$$

This formula is called NTSC grayscale formula. Note that the Green color shares a higher weight because the human eye is more sensitive to green color. This happens because human's eye contains three kinds of photorecepter cells:
L-cones:(sensitive to light with longer wavelength(e.g. red))
M-cones:(sensitive to light with mediate wavelength(e.g. green))
S-cones:(sensitive to light with shorter wavelength(e.g. blue))
Our eyes contain more M-cones because in the past, it's important to recognize green color(crops).

### 3.3.3 Gaussblur

Opencv has a well sealed pattern recognition function. The details will be explained here. In the program "chess4.py", line 24,the Gaussblur function is used. The Gaussblur function is a function used to smooth the picture and reduces noises.
In brief, the gauss function calculates the color of a pixel according to the rgb value of pixels nearby. The further the pixel is to the pixel being calculated, the smaller the weight of pixel is given. The formula is given by:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

1.x,y presents the difference between the pixel to the pixel being calculated now.
2.$\sigma$ represents the standard deviation of the gauss function.

### 3.3.4 edge recognition algorithm: Canny Edge Detection

Canny Edge detection is a multi-stage algorithm used to detect edges. Canny Edge Detection process involves several stages:

- convert the image to grayscale
- reduce noise with a Gaussian filter
- compute the gradient intensity and direction
- apply non-maximum suppression
- use double thresholding to detect strong and weak edges.

The first stage and second stage have already been introduced. Let's start with the third stage. The algorithm uses the convolution kernel(usually 3*3 or 5*5) and calculates a pixel's gradient at the x axis and the y axis. Then, the angle and magnitude can be calculated using

$$\theta = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

$$magnitude = \sqrt{G_x^2 + G_y^2}$$

non-maximum suppression transverse every kernal and saves the value with the highest magnitude , while other values with the same direction are disregarded.
Double thresholding contains two values: minval and maxval. The value which exceeds the maxval is considered as a real strong edge, while the value between minval and maxval is considered as a potential weak edge, others are discarded.
The weak edges which is attached to the strong edge is conserved, and those which is no attached to the strong edge is considered as a edge inside the body and tossed.

## 4 Issues and solutions

In the development process, I met some interesting problems, and the following contents will be a summary of problems and solutions.

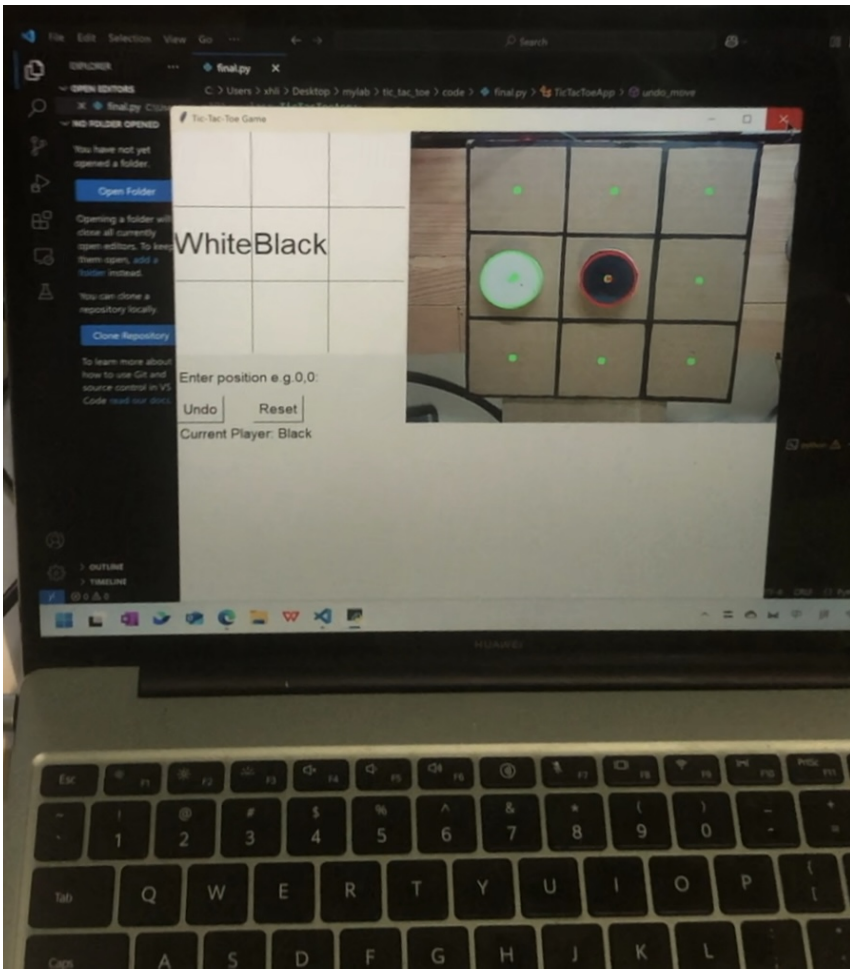


Figure 8: The final result

## 4.1 The choice of pattern recognition algorithm

At first, the blob algorithm is introduced, in which the pixels are separated into binary values in terms of the given threshold value.
link to the algorithm:blob detection-opencv
https://docs.opencv.org/3.4/d0/d7a/classcv_1_1SimpleBlobDetector.html
However, this didn't work because it is very susceptible to the environment.
Then, an addition algorithm called adaptive blob detection is used, with an adaptive parameter according to the environment. This still didn't work because reflected light affected this drastically.
Eventually, the more complicated algorithm called Canny Edge Detection was used and received great results.

## 4.2 The overheated electromagnet

A 12V voltage is applied to the electromagnet, and a 15 seconds opening will make it overheated.My fingers are even scalded during the development. I was using the I/O Pin to control it at that time.

Later, I soon found out that the PWM function(analogwrite) can address the problem and a 70%duty cycle can last the time to 30 seconds, which is enough.

## 4.3 The blocked UI

When testing the program, I found out that the camera display seemed slow and jammed. Later, I found out that the pyserial function "serial.write(data)" can jam the loop, and the camera is also jammed. Also, it requires quite a long time to repeat the whole loop, which means the FPS of camera display will decrease.

To deal with this problem, I imported an embedded lib in python called "Thread". Hence, the whole process of camera display is sealed into a thread and becomes much quicker.

## 5 3D-object design

I designed the pieces and some parts to reinforce the arm. The software I used is called free CAD https://www.freecad.org/, which is a free and open-source 3D-design software. I first designed the chess stone.

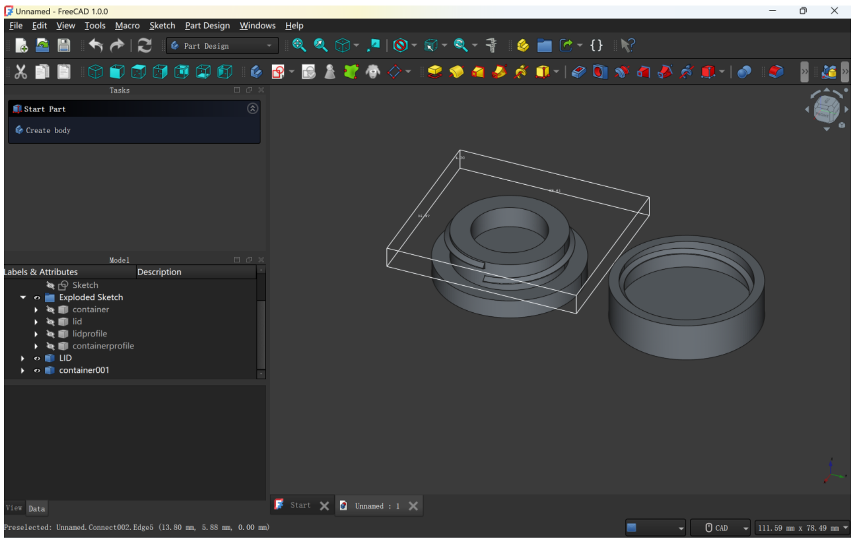


Figure 9: The chess stone

The stone is printed using PLA eco-friendly material. It's important to notice that when designing a lid. It's important to reserve a suitable thickness of material, according to my experience, a stickness at 2mm will be enough robust.
And when you have finished, remember to use the boolean operation "Union" in free CAD to check if there are collided parts.

## 6 Hardware requirement

This is the hardware used in the project. If you happen to see this and are interested to replicate or optimize this, you can check this list.

- robotic arm(servo*3)
  **https://e.tb.cn/h.TA6TEWTQkq1XTVt?tk=6mZYeprHWsb**
- stepper and sliding table
  **https://e.tb.cn/h.TzAFy1E8iAvDRZN?tk=agvmepruMKa**
- electromagnet
  **https://e.tb.cn/h.TAcW9pFKJPsERjF?tk=xYgbepry1mV**

- **USB*2(UART supported)**
- **Dupont line*25**
- Pieces white*4, black*5(stl file designed by me)
  **https://github.com/Kimili2008/Mylab/tree/main/design**
- **Arduino development board(e.g. UNO)*1**