Group leader - SAMARAWEERA D.T. - 200564J - 0705505337

- If our controller takes too much time to build, copy and paste the code into a new C++ controller.
- Our robot can stay without doing anything for some time. Therefore wait for some time for it to take the relevant actions (max 15 seconds).
- Robot's starting position should be parallel to the starting white square.

(We have made the robot in webots and haven't imported the CAD files. (Our robot arm mechanism can be practically implemented as in the links below.)
Robot Arm Mechanism
https://youtu.be/z3HNBt9s5ic
https://youtu.be/FNFmIbqV8nc)

```
#include <webots/Robot.hpp>
#include <webots/Motor.hpp>
#include <webots/Brake.hpp>
#include <webots/gyro.hpp>
#include <webots/InertialUnit.hpp>
#include <webots/DistanceSensor.hpp>
#include <webots/PositionSensor.hpp>
#include <iostream>
#include <ctime>
#include <thread>
#include <string>

#include <chrono>
#include <cstdlib>
#include <unistd.h>

#define TIME_STEP 16
#define MAX_SPEED 6.18
// All the webots classes are defined in the "webots" namespace
using namespace webots;
using namespace std;
using namespace std::this_thread;
using namespace std::chrono;
DistanceSensor* ir[8]; //ir_panle
DistanceSensor* ds[2];
DistanceSensor* fds;
DistanceSensor* fds_angle;
DistanceSensor* ds_top;
DistanceSensor* irf;
```

```
PositionSensor* pos_left;
PositionSensor* pos_right;
Motor* leftMotor;
Motor* rightMotor;
Brake* left_brk;
Brake* right_brk;
Motor* left_linear_motor1;
Motor* left_linear_motor2;
Motor* left_linear_motor3;
Motor* right_linear_motor1;
Motor* right_linear_motor2;
Motor* right_linear_motor3;
InertialUnit* gyroSens;

double junValues[8];

float kp = 0.08;
float ki = 0.005;
float kd = 0.0001;
double p = 0;
double i = 0;
double d = 0;
double lastError = 0;
bool pidOn=true;
double M_SPEED=MAX_SPEED*0.5;
double last_left_speed;
double last_right_speed;
double leftSpeed;
double rightSpeed;
double dc = 0; //damping coeficient
bool leftWall;
bool rightWall;
bool rook;
double leftDsValue;
double rightDsValue;
double hardLength;
double right_pos;
double left_pos;
double chessStart;
bool chessActivate = false;
bool x =true;
bool y =false;
bool z =true;
bool t =true;
```

```
bool king=false;
string kingPosition;
bool drop_activate =true;
bool wayOut1=true;
bool playingChess=false;
bool checkMate=false;
double front_colour;
float sqre=11.8;
float sqrered=13;
std::time_t initial_time;
std::time_t initial_time2;
std::time_t drop_initial_time;
std::time_t chamber_initial_time;
std::time_t chamber_initial_time2;
std::time_t onBridge_initial_time;
std::time_t onBridge_initial_time2;

//secret chamber
bool chamber_active=false;
float start_right;
float start_left;
float gap;
float n;
float iteration_starting_left;
float iteration_starting_right;
bool step1=true;
bool step2=false;
bool r_true=true;
bool step1_initial=true;
int iterations;
int r=0;
//turn
double temper=0;

bool dashFound=false;
int climb=0;

bool bridgeArrival1=false;
bool bridgeArrival2=false;

bool onBridge1=false;
bool onBridge2=false;
```

```
// This is the main program of your controller.
// It creates an instance of your Robot instance, launches its
// function(s) and destroys it at the end of the execution.
// Note that only one instance of Robot should be created in
// a controller program.
// The arguments of the main function can be specified by the
// "controllerArgs" field of the Robot node


bool piece(){
    double front_distance = fds->getValue();
    //check whethere there's any piece using ultrasonic
    if (front_distance<450){
        return 1;
    }
    else{
        return 0;
    }
}

int isClimb(){
  const double* gyroVal = gyroSens->getRollPitchYaw();
  std::cout << "jgyro val "<< gyroVal[0]  <<std::endl;


  if (gyroVal[0]<-0.1){
    climb=1;
    dashFound=false;
  }
  else if(gyroVal[0]>0.1){
    climb=2;
    dashFound=false;
  }
  else{
    climb=0;
  }
  return climb;
}

bool whiteking(){
    front_colour = irf->getValue();
    double king_check = ds_top->getValue();
    std::cout<<"called"<<std::endl;
```

```
    //check whether the piece is king using ultrasonic2
    if (king_check<950 && front_colour<990){
        //check whether the king is white using ir

        return 1;

    }
    else{
        return 0;
    }
}
void pickup(){

    if (x){
        x=false;
        initial_time = time(NULL);
        right_pos=pos_right->getValue();
        left_pos=pos_left->getValue();
        chessStart=left_pos+sqre;


    }
    std::cout << "Initial time : " << initial_time << std::endl;
    std::time_t current_time;
    current_time = time(NULL);

    std::time_t temp_time = current_time - initial_time;
    std::cout << "temp_time :" << temp_time << std::endl;

    if(0<temp_time && temp_time<=3){
        left_linear_motor1->setVelocity(0.08);
        left_linear_motor1->setPosition(0.20);
        right_linear_motor1->setVelocity(0.08);
        right_linear_motor1->setPosition(0.20);
    }
    else if(8<temp_time && temp_time<=10){
        left_linear_motor2->setVelocity(0.01);
        left_linear_motor2->setPosition(-0.015);
        right_linear_motor2->setVelocity(0.01);
        right_linear_motor2->setPosition(0.015);
    }
    else if(3<temp_time && temp_time<=8){
        left_linear_motor3->setVelocity(0.01);
        left_linear_motor3->setPosition(-0.05);
```

```
        right_linear_motor3->setVelocity(0.01);
        right_linear_motor3->setPosition(-0.05);

    }
    else if(10<temp_time && temp_time<=15){
        left_linear_motor3->setVelocity(0.01);
        left_linear_motor3->setPosition(0);
        right_linear_motor3->setVelocity(0.01);
        right_linear_motor3->setPosition(0);

    }
    else if(15<temp_time && temp_time<=18){
        left_linear_motor1->setVelocity(0.08);
        left_linear_motor1->setPosition(0);
        right_linear_motor1->setVelocity(0.08);
        right_linear_motor1->setPosition(0);
        leftMotor->setVelocity(0.7*MAX_SPEED);
        leftMotor->setPosition(left_pos+sqre);
        rightMotor->setVelocity(0.7*MAX_SPEED);
        rightMotor->setPosition(right_pos+sqre);
        z=false;

    }

}

void drop(){
    if (drop_activate){
        drop_activate=false;
        drop_initial_time = time(NULL);
    }
    std::cout << "Initial time : " << drop_initial_time << std::endl;
    std::time_t current_time;
    current_time = time(NULL);

    std::time_t temp_time = current_time - drop_initial_time;
    std::cout << "temp_time_drop :" << temp_time << std::endl;

    if(0<temp_time && temp_time<=3){
        left_linear_motor1->setVelocity(0.08);
        left_linear_motor1->setPosition(0.20);
        right_linear_motor1->setVelocity(0.08);
        right_linear_motor1->setPosition(0.20);
    }
```

```
    else if(8<temp_time && temp_time<=10){
        left_linear_motor2->setVelocity(0.01);
        left_linear_motor2->setPosition(0.0);
        right_linear_motor2->setVelocity(0.01);
        right_linear_motor2->setPosition(0.0);
    }
    else if(3<temp_time && temp_time<=8){
        left_linear_motor3->setVelocity(0.01);
        left_linear_motor3->setPosition(-0.05);
        right_linear_motor3->setVelocity(0.01);
        right_linear_motor3->setPosition(-0.05);

    }
    else if(10<temp_time && temp_time<=15){
        left_linear_motor3->setVelocity(0.01);
        left_linear_motor3->setPosition(0);
        right_linear_motor3->setVelocity(0.01);
        right_linear_motor3->setPosition(0);

    }
    else if(15<temp_time && temp_time<=18){
        left_linear_motor1->setVelocity(0.08);
        left_linear_motor1->setPosition(0);
        right_linear_motor1->setVelocity(0.08);
        right_linear_motor1->setPosition(0);

    }

}
void dropLeft(){
    if (drop_activate){
        drop_activate=false;
        drop_initial_time = time(NULL);
    }
    std::cout << "Initial time : " << drop_initial_time << std::endl;
    std::time_t current_time;
    current_time = time(NULL);

    std::time_t temp_time = current_time - drop_initial_time;
    std::cout << "temp_time_drop :" << temp_time << std::endl;

    if(0<temp_time && temp_time<=3){
        left_linear_motor1->setVelocity(0.08);
        left_linear_motor1->setPosition(0.20);
```

```
        right_linear_motor1->setVelocity(0.08);
        right_linear_motor1->setPosition(0.20);
    }
    else if(8<temp_time && temp_time<=10){
        left_linear_motor2->setVelocity(0.01);
        left_linear_motor2->setPosition(0.0);
        right_linear_motor2->setVelocity(0.01);
        right_linear_motor2->setPosition(0.015);
    }
    else if(3<temp_time && temp_time<=8){
        left_linear_motor3->setVelocity(0.01);
        left_linear_motor3->setPosition(-0.03);
        right_linear_motor3->setVelocity(0.01);
        right_linear_motor3->setPosition(-0.03);

    }
    else if(10<temp_time && temp_time<=15){
        left_linear_motor3->setVelocity(0.01);
        left_linear_motor3->setPosition(0);
        right_linear_motor3->setVelocity(0.01);
        right_linear_motor3->setPosition(0);

    }
    else if(15<temp_time && temp_time<=18){
        left_linear_motor1->setVelocity(0.08);
        left_linear_motor1->setPosition(0);
        right_linear_motor1->setVelocity(0.08);
        right_linear_motor1->setPosition(0);

    }

}


 void sharpTurn(int turn) {
    double right_pos=pos_right->getValue();
    std::cout<<right_pos;
    double left_pos=pos_left->getValue();
    std::cout<<left_pos;
    if (turn == 0 && temper==0) {
        temper=right_pos;
        hardLength = 110.0;
        std::cout << "turning left"<<std::endl;
        leftSpeed = -0.5 * MAX_SPEED;
```

```
    rightSpeed = 0.5 * MAX_SPEED;
}
else if (turn == 0 && right_pos>temper+6.18){
    std::cout << "going forward"<<std::endl;
    leftSpeed = 0.5 * MAX_SPEED;
    rightSpeed = 0.5 * MAX_SPEED;

}
else if (turn == 1) {
    hardLength = 18.0;
    std::cout << "going forward"<<std::endl;
    leftSpeed = 0.5 * MAX_SPEED;
    rightSpeed = 0.5 * MAX_SPEED;
}

else if (turn == 2) {
    hardLength = 110.0;
    std::cout << "turning right"<<std::endl;
    leftSpeed = 0.5 * MAX_SPEED;
    rightSpeed = -0.5 * MAX_SPEED;
}
else if (turn == 80) {
    hardLength = 170.0;
    std::cout << "circle left"<<std::endl;
    leftSpeed = 0 * MAX_SPEED;
    rightSpeed = 0.5 * MAX_SPEED;
}
else if (turn == 82) {
    hardLength = 170.0;
    std::cout << "circle right"<<std::endl;
    leftSpeed = 0.5 * MAX_SPEED;
    rightSpeed = 0 * MAX_SPEED;
}
else if (turn == -1){
    hardLength = 142.0;
    std::cout << "turning back"<<std::endl;
    leftSpeed = -0.5 * MAX_SPEED;
    rightSpeed = 0.5 * MAX_SPEED;
}
else if (turn == -2){
    hardLength = 142.0;
    std::cout << "turning back 2"<<std::endl;
    leftSpeed = 0.5 * MAX_SPEED;
    rightSpeed = -0.5 * MAX_SPEED;
```

```
}
else if (turn == 22){
    hardLength = 133.0;
    std::cout << "ramp right"<<std::endl;
    leftSpeed = 0.25 * MAX_SPEED;
    rightSpeed = -0.25 * MAX_SPEED;
}
else if (turn == 20){
    hardLength = 133.0;
    std::cout << "ramp left"<<std::endl;
    leftSpeed = -0.25 * MAX_SPEED;
    rightSpeed = 0.25 * MAX_SPEED;
}
else if (turn == 21){
    hardLength = 60.0;
    std::cout << "ramp adjust"<<std::endl;
    leftSpeed = 0.25 * MAX_SPEED;
    rightSpeed = 0.25 * MAX_SPEED;
}
else if (turn == -10){
    hardLength = 120.0;
    std::cout << "reverseC"<<std::endl;
    leftSpeed = -0.25 * MAX_SPEED;
    rightSpeed = -0.25 * MAX_SPEED;
}
else if (turn == 52){
    hardLength = 150.0;
    std::cout << "mid right"<<std::endl;
    leftSpeed = 0.5 * MAX_SPEED;
    rightSpeed = 0 * MAX_SPEED;
}
else if (turn == 50){
    hardLength = 150.0;
    std::cout << "mid left"<<std::endl;
    leftSpeed = 0 * MAX_SPEED;
    rightSpeed = 0.5 * MAX_SPEED;;
}
else if (turn == 32){
    hardLength = 125.0;
    std::cout << "mid right"<<std::endl;
    leftSpeed = 0.5 * MAX_SPEED;
    rightSpeed = 0 * MAX_SPEED;
}
```

```
else if (turn == 30){
   hardLength = 125.0;
   std::cout << "mid left"<<std::endl;
   leftSpeed = 0 * MAX_SPEED;
   rightSpeed = 0.5 * MAX_SPEED;;
}
else if (turn == 41){
   hardLength = 16.0;
   std::cout << "quad 4 forward"<<std::endl;
   leftSpeed = 0.5 * MAX_SPEED;
   rightSpeed = 0.5 * MAX_SPEED;
}
else if (turn == 100){
   hardLength = 75.0;
   std::cout << "start forward"<<std::endl;
   leftSpeed = 0.5 * MAX_SPEED;
   rightSpeed = 0.5 * MAX_SPEED;
}
else {
   cout << "wrong input";
}


leftMotor->setVelocity(leftSpeed);
rightMotor->setVelocity(rightSpeed);

/**
double pos_val = pos_right->getValue();
if (turn == 2) {
   double pos_val = pos_left->getValue();
}
if (abs(pos_val) > 0) {
   pos_lst.push_back(pos_val);
   std::cout << "encorder"<< abs(pos_lst.begin() - pos_lst.end())<< std::endl;
   if (abs(pos_lst.begin() - pos_lst.end()) > hardLength) {
      junc = -1;
      pos_lst={};
      turn_command = false;
      direct_count += 1;
      if (safety){go=false;}
      canUpdateStates = true;
      if (state[direct_count]=="ramp adjust" || state[direct_count-1]=="foundRev"){
        junc = 9;
        std::cout<<"next com"<<endl;
```

```
            }
            p=0;
            i=0;
            d=0;
            leftSpeed=0;
            rightSpeed=0;
            //canUpdateLoc=true;
            //remove this at final stage. This only for safety
            //depends on final robot speed
        }
    }**/
}
int checkmate(int path){
    int count=1;
    for (int i = 0; i < path; i++) {
        if (piece()){
            if (whiteking()){
                rook=1;
                break;
            }
            else{
                break;
            }
        }
        //move step forward
        count++;
    }
    sharpTurn(-1);//turn back
    //move step*(count-1) //a8
    return rook;
}


void colorCheck(){
    double ir3 = ir[3]->getValue();
    double ir4 = ir[4]->getValue();
    if (ir3==653 && ir4==653){
        leftSpeed=0;
        rightSpeed=0;
    }

}
```

```
void pido() {
    // initializing the pid coefficients
    // 0.14 0.001 0.0001


    // initializing the array to store ir sensor values
    double ir_values[8];
    double ir_sum = 0;

    for (int i = 0; i < 8; i++) {
        double ir_val = ir[i]->getValue();
        ir_values[i] = ir_val;
        ir_sum += ir_val;
    }

    // standard deviation of the ir array
    double sd = 0;

    for (int i = 0; i < 8; i++) {
        sd += (ir_values[i] - ir_sum / 8)* (ir_values[i] - ir_sum / 8);
    }

    sd = pow(sd / 8, 0.5);

    for (int i = 0; i < 8; i++) {
        ir_values[i] = (ir_values[i] - ir_sum / 8) / (sd + 0.0001);
    }
    // now the ir readings are normalized to a mean of 0 and standard deviation of 1

    // variable for storing the position
    double pos = 0;

    for (int i = 0; i < 4; i++) {
        pos += ir_values[i] * (-i + 4) + ir_values[7 - i] * (-4 + i);
    }

    double error = 0.0 - pos;
    p = error;
    i = i + p;
    d = error - lastError;
    lastError = error;
    double motor_speed = kp * p + ki * i + kd * d;
    leftSpeed = 0.5 * MAX_SPEED - motor_speed;
    rightSpeed = 0.5 * MAX_SPEED + motor_speed;
```

```
}
//set motors
void setMotors() {
   for (int j = 0; j < 8; j++) {
      double irVal = ir[j]->getValue();
      junValues[j] = irVal;
   }

   bool temp = true;
   for (int j = 0; j < 8; j++) {

      if (junValues[j]==1000) {
         temp = false;
      }
   }
   if (temp){
      leftSpeed = MAX_SPEED *0.5;
      rightSpeed = MAX_SPEED *0.5;

   }
   //setting motor speeds
   leftMotor->setVelocity(leftSpeed);
   rightMotor->setVelocity(rightSpeed);

   //setting brakes
   left_brk->setDampingConstant(dc);
   right_brk->setDampingConstant(dc);

   //storing the speed for next loop
   last_left_speed = leftSpeed;
   last_right_speed = rightSpeed;

}

void pid() {
   // initializing the PID coefficients
   //float kp = 0.14;
   //float ki = 0.001;
   // 0.0001
   //float kp = 0.11;
   //float ki = 0.001;
   //float kd = 0.005;
   std::cout<<"pid called"<< std::endl;
   // initializing the array to store IR sensor values
```

```
double irValues[8];
double irSum = 0;

for (int j = 0; j < 8; j++) {
    double irVal = ir[j]->getValue();
    irValues[j] = irVal;
    junValues[j] = irVal;
    irSum += irVal;
}


//for (double item : junValues)
    //std::cout << item << ", ";
//cout << endl;

// standard deviation of the ir array
double sd = 0;

for (int j = 0; j < 8; j++) {
    sd += (irValues[j] - irSum / 8) * (irValues[j] - irSum / 8);
}

sd = pow(sd / 8, 0.5);

for (int j = 0; j < 8; j++) {
    irValues[j] = (irValues[j] - irSum / 8) / (sd + 0.0001);
}
//for (double item : irValues)
    //std::cout << item << ", ";
//cout << endl;
// now the ir readings are normalized to a mean of 0 and standard deviation of 1

// variable for storing the position
double pos = 0;

for (int j = 0; j < 4; j++) {
    pos += irValues[j] * (-j + 4) + irValues[7 - j] * (-4 + j);
}

double error = 0.0 - pos;
p = error;
i = i + p;

if (i > 200) {
```

```
        i = 200;
    }
    else if (i < -200) {
        i = -200;
    }

    d = error - lastError;
    lastError = error;
    double motorSpeed = kp * p + ki * i + kd * d;
    //cout << "motor speed: " << motorSpeed << endl;

    leftSpeed =  M_SPEED - motorSpeed;
    rightSpeed = M_SPEED + motorSpeed;
    //std::cout <<"left"<< leftSpeed<< std::endl;
};
void wallFollowing() {
    std::cout << "wall following"<<std::endl;
    if (leftWall) {
        cout << "left wall" << endl;
        if (leftDsValue > 750) {
            //cout << "turn left" << endl;
            leftSpeed = MAX_SPEED * 0.3;
            rightSpeed = MAX_SPEED * 0.5;
        }
        else if (leftDsValue < 650) {
            //cout << "turn right" << endl;
            rightSpeed = MAX_SPEED * 0.3;
            leftSpeed = MAX_SPEED * 0.5;
        }
        else {
            leftSpeed = MAX_SPEED * 0.5;
            rightSpeed = MAX_SPEED * 0.5;
        }
    }

    else if (rightWall) {
        cout << "right wall" << endl;
        if (rightDsValue > 750) {
            //cout << "turn right" << endl;
            rightSpeed = MAX_SPEED * 0.3;
            leftSpeed = MAX_SPEED * 0.5;
        }
        else if (rightDsValue < 650) {
            //cout << "turn left" << endl;
```

```
            leftSpeed = MAX_SPEED * 0.3;
            rightSpeed = MAX_SPEED * 0.5;
        }
        else {
            leftSpeed = MAX_SPEED * 0.5;
            rightSpeed = MAX_SPEED * 0.5;
        }
    }

}
//No line
void noLine() {
    //cout << "no line" << endl;
    leftSpeed = MAX_SPEED * 0.5;
    rightSpeed = MAX_SPEED * 0.5;
}
void chessBoardEntrance(){

    std::cout << "At the entrance"<< std::endl;

    double front_distance = fds->getValue();
    if (front_distance<350 || y){
        y=true;
        front_distance=250;

        leftSpeed = 0;
        rightSpeed = 0;
        leftMotor->setVelocity(leftSpeed);
        rightMotor->setVelocity(rightSpeed);
        pickup();
    }

}
void wall() {
    leftDsValue = ds[0]->getValue();
    rightDsValue = ds[1]->getValue();
    for (int j = 0; j < 8; j++) {
        double irVal = ir[j]->getValue();
        junValues[j] = irVal;
    }


    cout << "left ds: " << leftDsValue << endl;
    cout << "right ds: " << rightDsValue << endl;
```

```
    //cout << "hey" << endl;

    leftWall = leftDsValue < 1000;
    rightWall = rightDsValue < 1000;

    ////////////////////////////////
    //for checking whether there is a line
    bool cond = false;
    for (int j = 0; j < 8; j++) {
        cout << junValues[j] << endl;
        if (junValues[j] < 1000) {
            cond = true;
            //cout << "hey" << endl;
        }
    }

    cout << "cond" << cond << endl;

    //condition for wall following
    if ((leftWall or rightWall) && !cond) {
        wallFollowing();
        pidOn=false;
    }
    else if (!cond) {
        noLine();
        pidOn=false;
    }
    //cout << leftSpeed << endl;
    //cout << rightSpeed << endl;


    //cout << leftSpeed << endl;
    //cout << rightSpeed << endl;
}

void chessBoard(){




    if (t){
      initial_time2 = time(NULL);
```

```
  t=false;
  right_pos=pos_right->getValue();
  left_pos=pos_left->getValue();
}
std::cout <<left_pos<<" "<<right_pos<< std::endl;
std::time_t current_time;
current_time = time(NULL);

std::time_t temp_time = current_time - initial_time2;
std::cout << "temp time : " << temp_time << std::endl;
std::cout << "king position : " << kingPosition << std::endl;

std::cout << king;

std::cout<<right_pos;

std::cout<<left_pos<< std::endl;

double front_colour = irf->getValue();
double king_check = ds_top->getValue();
std::cout << "king ir ultrsonic " << front_colour <<" "<<king_check << std::endl;
if(0<temp_time && temp_time<=3){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18);

}

else if(3<temp_time && temp_time<=8){
    playingChess=true;
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18+11.7809);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18+11.7809);
    if (piece()){
      king=whiteking();
      initial_time2=initial_time2-76;
    }

}
else if(8<temp_time && temp_time<=11){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18+11+6.18);
```

```
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18+11.7809-6.18);
    if (piece() && temp_time==11){
      king=whiteking();
      initial_time2=initial_time2-60;
    }
  }
  else if(11<temp_time && temp_time<=16){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18+11.7809+6.18+11.7809);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18+11.7809-6.18+11.7809);
    if (piece()){
      king=whiteking();
      initial_time2=initial_time2-50;

    }
  }

  else if(16<temp_time && temp_time<=21){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18+11.7809+6.18+11.7809+11.7809);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18+11.7809-6.18+11.7809+11.7809);
    if (piece()){
      king=whiteking();
      initial_time2=initial_time2-40;

    }
  }
  else if(21<temp_time && temp_time<=26){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18+11.7809+6.18+11.7809*3);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18+11.7809-6.18+11.7809*3);
    if (piece()){
      king=whiteking();
      initial_time2=initial_time2-30;

    }
  }
  else if(26<temp_time && temp_time<=31){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18+11.7809+6.18+11.7809*4);
```

```
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18+11.7809-6.18+11.7809*4);
    if (piece()){
      king=whiteking();
      initial_time2=initial_time2-20;

    }
}
else if(31<temp_time && temp_time<=36){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18+11.7809+6.18+11.7809*5);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18+11.7809-6.18+11.7809*5);
    if (piece()){
      king=whiteking();
      initial_time2=initial_time2-10;

    }

}
else if(36<temp_time && temp_time<=41){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18+11.7809+6.18+11.7809*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18+11.7809-6.18+11.7809*6);
    if (piece()){
      king=whiteking();
    }
}

else if(41<temp_time && temp_time<=71){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+11.7809);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+11.7809);

}
else if(71<temp_time && temp_time<=74){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18+11.7809);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18+11.7809);
}
else if(74<temp_time && temp_time<=79){
```

```
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18);
    if (king){
        initial_time2=initial_time2-935;
        kingPosition="A8";
    }
}

else if(79<temp_time && temp_time<=82){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos);

}
//round2
else if(82<temp_time && temp_time<=85){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18);
    if (piece() && temp_time<=85){
      initial_time2=initial_time2-456;
      wayOut1=false;
      king=whiteking();
    }
}
else if(85<temp_time && temp_time<=90){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18+sqre);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18+sqre);
    if (piece()){
      initial_time2=initial_time2-380;
      king=whiteking();
    }
}
else if(90<temp_time && temp_time<=95){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18+sqre*2);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18+sqre*2);
```

```
    if (piece()){
      initial_time2=initial_time2-304;
      king=whiteking();
    }
  }
  else if(95<temp_time && temp_time<=100){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18+sqre*3);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18+sqre*3);
    if (piece()){
      initial_time2=initial_time2-228;
      king=whiteking();
    }
  }
  else if(100<temp_time && temp_time<=105){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18+sqre*4);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18+sqre*4);
    if (piece()){
      initial_time2=initial_time2-152;
      king=whiteking();
    }
  }
  else if(105<temp_time && temp_time<=110){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18+sqre*5);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18+sqre*5);
    if (piece()){
      initial_time2=initial_time2-76;
      king=whiteking();
    }
  }
  else if(110<temp_time && temp_time<=115){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18+sqre*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18+sqre*6);

  }
  else if(115<temp_time && temp_time<=118){
    leftMotor->setVelocity(0.5*MAX_SPEED);
```

```
    leftMotor->setPosition(left_pos+sqre*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6);
    if (piece() && temp_time==118){
      initial_time2=initial_time2-60;
      king=whiteking();
    }
  }
  else if(118<temp_time && temp_time<=123){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7);
    if (piece()){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
  }
  else if(123<temp_time && temp_time<=128){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
  }
  else if(128<temp_time && temp_time<=133){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*9);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*9);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
  }
  else if(133<temp_time && temp_time<=138){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*10);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*10);
    if (piece()){
```

```
      initial_time2=initial_time2-20;
      king=whiteking();
    }
}
else if(138<temp_time && temp_time<=142){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*11);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*11);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
}
else if(142<temp_time && temp_time<=148){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*12);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*12);
    if (piece()){
      king=whiteking();
    }
}

else if(148<temp_time && temp_time<=178){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6);
}
else if(178<temp_time && temp_time<=181){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6-6.18);
}
else if(181<temp_time && temp_time<=186){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5-6.18);
    if (king){
        initial_time2=initial_time2-854;
        kingPosition="A1";
```

```
    }
  }
  else if(186<temp_time && temp_time<=189){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5);
    if (piece() && temp_time==189){
      initial_time2=initial_time2-60;
      king=whiteking();
    }
  }
  else if(189<temp_time && temp_time<=194){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6);
    if (piece()){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
  }
  else if(194<temp_time && temp_time<=199){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
  }
  else if(119<temp_time && temp_time<=204){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
  }
  else if(204<temp_time && temp_time<=209){
    leftMotor->setVelocity(0.5*MAX_SPEED);
```

```
    leftMotor->setPosition(left_pos+sqre*9);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*9);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
}
else if(209<temp_time && temp_time<=214){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*10);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*10);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
}
else if(214<temp_time && temp_time<=219){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*11);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*11);
    if (piece()){

      king=whiteking();
    }
}
else if(219<temp_time && temp_time<=249){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5);

}

else if(249<temp_time && temp_time<=252){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5-6.18);
}
else if(252<temp_time && temp_time<=257){
    leftMotor->setVelocity(0.5*MAX_SPEED);
```

```
    leftMotor->setPosition(left_pos+sqre*4+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*4-6.18);
    if (king){
        initial_time2=initial_time2-834;
        kingPosition="A2";
    }
}
else if(257<temp_time && temp_time<=260){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*4);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*4);
    if (piece() && temp_time==260){
      initial_time2=initial_time2-60;
      king=whiteking();
    }
}
else if(260<temp_time && temp_time<=265){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5);
    if (piece()){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
}
else if(265<temp_time && temp_time<=270){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
}
else if(270<temp_time && temp_time<=275){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7);
    if (piece()){
```

```
      initial_time2=initial_time2-30;
      king=whiteking();
    }
}
else if(275<temp_time && temp_time<=280){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
}
else if(280<temp_time && temp_time<=285){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*9);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*9);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
}
else if(285<temp_time && temp_time<=290){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*10);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*10);
    if (piece()){
      king=whiteking();
    }
}
else if(290<temp_time && temp_time<=320){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*4);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*4);

}
else if(320<temp_time && temp_time<=323){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*4+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
```

```
    rightMotor->setPosition(right_pos+sqre*4-6.18);
}
else if(323<temp_time && temp_time<=328){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*3+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*3-6.18);
    if (king){
        initial_time2=initial_time2-809;
        kingPosition="A3";
    }
}
else if(328<temp_time && temp_time<=331){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*3);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*3);
    if (piece() && temp_time==331){
      initial_time2=initial_time2-60;
      king=whiteking();
    }
}
else if(331<temp_time && temp_time<=336){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*4);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*4);
    if (piece()){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
}
else if(336<temp_time && temp_time<=341){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
}
else if(341<temp_time && temp_time<=346){
    leftMotor->setVelocity(0.5*MAX_SPEED);
```

```
    leftMotor->setPosition(left_pos+sqre*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
  }
  else if(346<temp_time && temp_time<=351){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
  }
  else if(351<temp_time && temp_time<=356){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
  }
  else if(356<temp_time && temp_time<=361){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*9);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*9);
    if (piece()){
      king=whiteking();
    }
  }
  else if(361<temp_time && temp_time<=391){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*3);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*3);

  }
```

```
else if(391<temp_time && temp_time<=394){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*3+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*3-6.18);
}
else if(394<temp_time && temp_time<=399){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*2+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*2-6.18);
   if (king){
      initial_time2=initial_time2-779;
      kingPosition="A4";
   }
}
else if(399<temp_time && temp_time<=402){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*2);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*2);
   if (piece() && temp_time==402){
     initial_time2=initial_time2-60;
     king=whiteking();
   }
}
else if(402<temp_time && temp_time<=407){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*3);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*3);
   if (piece()){
     initial_time2=initial_time2-50;
     king=whiteking();
   }
}
else if(407<temp_time && temp_time<=412){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*4);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*4);
   if (piece()){
     initial_time2=initial_time2-40;
     king=whiteking();
```

```
    }
  }
  else if(412<temp_time && temp_time<=417){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
  }
  else if(417<temp_time && temp_time<=422){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
  }
  else if(422<temp_time && temp_time<=427){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
  }
  else if(427<temp_time && temp_time<=432){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8);
    if (piece()){
      king=whiteking();
    }
  }
  else if(432<temp_time && temp_time<=462){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*2);
```

```
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*2);
}
else if(462<temp_time && temp_time<=465){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*2+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*2-6.18);
}
else if(465<temp_time && temp_time<=470){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre-6.18);
    if (king){
        initial_time2=initial_time2-744;
        kingPosition="A5";
    }
}
else if(470<temp_time && temp_time<=473){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre);
    if (piece() && temp_time<=473){
      initial_time2=initial_time2-60;
      king=whiteking();
    }
}
else if(473<temp_time && temp_time<=478){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*2);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*2);
    if (piece()){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
}
else if(478<temp_time && temp_time<=483){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*3);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*3);
```

```
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
}
else if(483<temp_time && temp_time<=488){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*4);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*4);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
}
else if(488<temp_time && temp_time<=493){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
}
else if(493<temp_time && temp_time<=498){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
}
else if(498<temp_time && temp_time<=503){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7);
    if (piece()){
      king=whiteking();
    }
}
```

```
else if(503<temp_time && temp_time<=533){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*1);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*1);
}
else if(533<temp_time && temp_time<=537){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre-6.18);
}
else if(537<temp_time && temp_time<=542){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos-6.18);
   if (king){
      initial_time2=initial_time2-703;
      kingPosition="A6";
   }

}
else if(542<temp_time && temp_time<=547){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos);


}//////////////////////////////////////////////////
//round3
//////////////////////////////////////////////////
else if(547<temp_time && temp_time<=552){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre);
   if (piece()){
     initial_time2=initial_time2-396;
     king=whiteking();
   }
}
```

```
else if(552<temp_time && temp_time<=557){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*2);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*2);
   if (piece()){
     initial_time2=initial_time2-330;
     king=whiteking();
   }
}
else if(557<temp_time && temp_time<=562){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*3);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*3);
   if (piece()){
     initial_time2=initial_time2-264;
     king=whiteking();
   }
}
else if(562<temp_time && temp_time<=567){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*4);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*4);
   if (piece()){
     initial_time2=initial_time2-198;
     king=whiteking();
   }
}
else if(567<temp_time && temp_time<=572){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*5);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*5);
   if (piece()){
     initial_time2=initial_time2-132;/////
     king=whiteking();
   }
}
else if(572<temp_time && temp_time<=577){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*6);
   rightMotor->setVelocity(0.5*MAX_SPEED);
```

```
    rightMotor->setPosition(right_pos+sqre*6);
    if (piece()){
      initial_time2=initial_time2-66;
      king=whiteking();
    }
  }
  else if(577<temp_time && temp_time<=582){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7);
  }
  else if(582<temp_time && temp_time<=585){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7-6.18);
    if (piece()){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
  }
  else if(585<temp_time && temp_time<=590){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8-6.18);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
  }
  else if(590<temp_time && temp_time<=595){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*9+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*9-6.18);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
  }
  else if(595<temp_time && temp_time<=600){
    leftMotor->setVelocity(0.5*MAX_SPEED);
```

```
    leftMotor->setPosition(left_pos+sqre*10+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*10-6.18);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
}
else if(600<temp_time && temp_time<=605){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*11+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*11-6.18);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
}
else if(605<temp_time && temp_time<=610){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*12+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*12-6.18);
    if (piece()){
      king=whiteking();
    }
}
else if(610<temp_time && temp_time<=635){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7-6.18);
}
else if(635<temp_time && temp_time<=638){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7);
}
else if(638<temp_time && temp_time<=643){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6);
```

```
    if (king){
        initial_time2=initial_time2-628;
        kingPosition="H7";
    }
}
else if(643<temp_time && temp_time<=646){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6-6.18);
    if (piece() && temp_time==646){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
}
else if(646<temp_time && temp_time<=651){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7-6.18);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
}
else if(651<temp_time && temp_time<=656){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8-6.18);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
}
else if(656<temp_time && temp_time<=661){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*9+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*9-6.18);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
```

```
}
else if(661<temp_time && temp_time<=666){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*10+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*10-6.18);
   if (piece()){
     initial_time2=initial_time2-10;
     king=whiteking();
   }
}
else if(666<temp_time && temp_time<=671){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*11+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*11-6.18);
   if (piece()){
     king=whiteking();
   }
}

else if(671<temp_time && temp_time<=696){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*6+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*6-6.18);
}
else if(696<temp_time && temp_time<=699){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*6);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*6);
}
else if(699<temp_time && temp_time<=704){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*5);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*5);
   if (king){
      initial_time2=initial_time2-620;
      kingPosition="G7";
   }
}
else if(704<temp_time && temp_time<=707){
```

```
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5-6.18);
    if (piece() && temp_time==707){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
  }
  else if(707<temp_time && temp_time<=712){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6-6.18);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
  }
  else if(712<temp_time && temp_time<=717){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7-6.18);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
  }
  else if(717<temp_time && temp_time<=722){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8-6.18);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
  }
  else if(722<temp_time && temp_time<=727){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*9+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*9-6.18);
```

```
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
}
else if(727<temp_time && temp_time<=732){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*10+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*10-6.18);
    if (piece()){
      king=whiteking();
    }
}


else if(732<temp_time && temp_time<=757){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5-6.18);

}
else if(757<temp_time && temp_time<=760){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5);
}
else if(760<temp_time && temp_time<=765){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*4);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*4);
    if (king){
        initial_time2=initial_time2-607;
        kingPosition="F7";
    }
}/////////
else if(765<temp_time && temp_time<=768){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*4+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*4-6.18);
```

```
    if (piece() && temp_time==768){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
  }
  else if(768<temp_time && temp_time<=773){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5-6.18);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
  }
  else if(773<temp_time && temp_time<=778){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6-6.18);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
  }
  else if(778<temp_time && temp_time<=783){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7-6.18);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
  }
  else if(783<temp_time && temp_time<=788){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8-6.18);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
```

```
}
else if(788<temp_time && temp_time<=793){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*9+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*9-6.18);
   if (piece()){
     king=whiteking();
   }
}
else if(793<temp_time && temp_time<=818){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*4+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*4-6.18);
}
else if(818<temp_time && temp_time<=823){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*4);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*4);
}
else if(823<temp_time && temp_time<=826){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*3);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*3);
   if (king){
      initial_time2=initial_time2-587;
      kingPosition="E7";
   }
}//////////////
else if(826<temp_time && temp_time<=829){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*3+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*3-6.18);
   if (piece() && temp_time==829){
     initial_time2=initial_time2-50;
     king=whiteking();
   }
}
else if(829<temp_time && temp_time<=834){
   leftMotor->setVelocity(0.5*MAX_SPEED);
```

```
    leftMotor->setPosition(left_pos+sqre*4+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*4-6.18);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
  }
  else if(834<temp_time && temp_time<=839){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5-6.18);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
  }
  else if(839<temp_time && temp_time<=844){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6-6.18);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
  }
  else if(844<temp_time && temp_time<=849){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7-6.18);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
  }
  else if(849<temp_time && temp_time<=854){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8-6.18);
    if (piece()){
```

```
      king=whiteking();
    }
}


else if(854<temp_time && temp_time<=879){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*3+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*3-6.18);
}
else if(879<temp_time && temp_time<=882){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*3);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*3);
}
else if(882<temp_time && temp_time<=887){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*2);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*2);
    if (king){
        initial_time2=initial_time2-566;
        kingPosition="D7";
    }
}/////////
else if(887<temp_time && temp_time<=890){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*2+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*2-6.18);
    if (piece() && temp_time==890){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
}
else if(890<temp_time && temp_time<=895){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*3+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*3-6.18);
    if (piece()){
      initial_time2=initial_time2-40;
```

```
      king=whiteking();
    }
  }
  else if(895<temp_time && temp_time<=900){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*4+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*4-6.18);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
  }
  else if(900<temp_time && temp_time<=905){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5-6.18);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
  }
  else if(905<temp_time && temp_time<=910){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6-6.18);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
  }
  else if(910<temp_time && temp_time<=915){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*7+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*7-6.18);
    if (piece()){
      king=whiteking();
    }
  }

  else if(915<temp_time && temp_time<=940){
```

```
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*2+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*2-6.18);
}
else if(940<temp_time && temp_time<=943){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*2);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*2);
}
else if(943<temp_time && temp_time<=948){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre);
    if (king){
        initial_time2=initial_time2-538;
        kingPosition="C7";
    }
}////////
else if(948<temp_time && temp_time<=951){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre-6.18);
    if (piece() && temp_time==951){
      initial_time2=initial_time2-50;
      king=whiteking();
    }
}
else if(951<temp_time && temp_time<=956){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*2+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*2-6.18);
    if (piece()){
      initial_time2=initial_time2-40;
      king=whiteking();
    }
}
else if(956<temp_time && temp_time<=961){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*3+6.18);
```

```
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*3-6.18);
    if (piece()){
      initial_time2=initial_time2-30;
      king=whiteking();
    }
  }
  else if(961<temp_time && temp_time<=966){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*4+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*4-6.18);
    if (piece()){
      initial_time2=initial_time2-20;
      king=whiteking();
    }
  }
  else if(966<temp_time && temp_time<=971){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5-6.18);
    if (piece()){
      initial_time2=initial_time2-10;
      king=whiteking();
    }
  }
  else if(971<temp_time && temp_time<=976){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6-6.18);
    if (piece()){
      king=whiteking();
    }
  }

  else if(976<temp_time && temp_time<=1001){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre-6.18);
    if (king){
       kingPosition="B7";
```

```
    }
}
else if(1001<temp_time && temp_time<=1004){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre);
}
else if(1004<temp_time && temp_time<=1009){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos);
    if (king){
        initial_time2=initial_time2-505;
        kingPosition="B7";
    }
}
//A8
else if(1009<temp_time && temp_time<=1014){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos-6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+6.18);
}
else if(1014<temp_time && temp_time<=1032){
    drop();
}
else if(1032<temp_time && temp_time<=1035){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos);
    if(temp_time==1035){
        initial_time2=initial_time2-497;
    }
}
//A1
else if(1035<temp_time && temp_time<=1040){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5-6.18);
}
```

```
else if(1040<temp_time && temp_time<=1058){
   drop();
}
else if(1058<temp_time && temp_time<=1083){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos-6.18);
}
else if(1083<temp_time && temp_time<=1086){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos);
   if(temp_time==1086){
      initial_time2=initial_time2-446;
   }
}
//A2
else if(1086<temp_time && temp_time<=1091){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*4+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*4-6.18);
}
else if(1091<temp_time && temp_time<=1109){
   drop();
}
else if(1109<temp_time && temp_time<=1129){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos-6.18);
}
else if(1129<temp_time && temp_time<=1132){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos);
   if(temp_time==1132){
      initial_time2=initial_time2-400;
   }
}
//A3
```

```
else if(1132<temp_time && temp_time<=1137){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*3+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*3-6.18);
}
else if(1137<temp_time && temp_time<=1155){
   drop();
}
else if(1155<temp_time && temp_time<=1170){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos-6.18);
}
else if(1170<temp_time && temp_time<=1173){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos);
   if(temp_time==1173){
      initial_time2=initial_time2-359;
   }
}
//A4
else if(1173<temp_time && temp_time<=1178){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*2+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*2-6.18);
}
else if(1178<temp_time && temp_time<=1196){
   drop();
}
else if(1196<temp_time && temp_time<=1206){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos-6.18);
}
else if(1206<temp_time && temp_time<=1209){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos);
   rightMotor->setVelocity(0.5*MAX_SPEED);
```

```
    rightMotor->setPosition(right_pos);
    if(temp_time==1209){
        initial_time2=initial_time2-323;
    }
}
//A5
else if(1209<temp_time && temp_time<=1214){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre-6.18);
}
else if(1214<temp_time && temp_time<=1232){
    drop();
}
else if(1232<temp_time && temp_time<=1237){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18);
}
else if(1237<temp_time && temp_time<=1240){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos);
    if(temp_time==1240){
        initial_time2=initial_time2-292;
    }
}
//A6
else if(1240<temp_time && temp_time<=1245){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18);
}
else if(1245<temp_time && temp_time<=1263){
    drop();
}
else if(1263<temp_time && temp_time<=1266){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos);
    rightMotor->setVelocity(0.5*MAX_SPEED);
```

```
    rightMotor->setPosition(right_pos);
    if(temp_time==1266){
        initial_time2=initial_time2-266;
    }
}
//H7
else if(1266<temp_time && temp_time<=1271){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*6);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*6);
}
else if(1271<temp_time && temp_time<=1289){
    drop();
}
else if(1289<temp_time && temp_time<=1319){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos);
    if(temp_time==1319){
        initial_time2=initial_time2-213;
    }
}
//G7
else if(1319<temp_time && temp_time<=1324){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*5);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*5);
}
else if(1324<temp_time && temp_time<=1342){
    drop();
}
else if(1342<temp_time && temp_time<=1367){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos);
    if(temp_time==1367){
        initial_time2=initial_time2-165;
    }
}
//F7
```

```
else if(1367<temp_time && temp_time<=1372){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*4);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*4);
}
else if(1372<temp_time && temp_time<=1390){
   drop();
}
else if(1390<temp_time && temp_time<=1410){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos);
   if(temp_time==1410){
      initial_time2=initial_time2-122;
   }
}
//E7
else if(1410<temp_time && temp_time<=1415){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*3);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*3);
}
else if(1415<temp_time && temp_time<=1433){
   drop();
}
else if(1433<temp_time && temp_time<=1448){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos);
   if(temp_time==1448){
      initial_time2=initial_time2-84;
   }
}
//D7
else if(1448<temp_time && temp_time<=1453){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre*2);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre*2);
}
```

```
else if(1453<temp_time && temp_time<=1471){
   drop();
}
else if(1471<temp_time && temp_time<=1481){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos);
   if(temp_time==1481){
      initial_time2=initial_time2-51;
   }
}
//C7
else if(1481<temp_time && temp_time<=1486){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos+sqre);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos+sqre);
}
else if(1486<temp_time && temp_time<=1504){
   drop();
}
else if(1504<temp_time && temp_time<=1509){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos);
   if(temp_time==1509){
      initial_time2=initial_time2-23;
   }
}
//B7
else if(1509<temp_time && temp_time<=1514){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(left_pos);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(right_pos);
}
else if(1514<temp_time && temp_time<=1532){
   drop();
}
else if(1532<temp_time && wayOut1){
   if(1532<temp_time && temp_time<=1535){
      leftMotor->setVelocity(0.5*MAX_SPEED);
```

```
    leftMotor->setPosition(left_pos+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18);
  }
  else if(1535<temp_time && temp_time<=1540){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18+sqrered);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18+sqrered);
  }
  else if(1540<temp_time && temp_time<=1543){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18*2+sqrered);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18*2+sqrered);
  }
  else if(1543<temp_time && temp_time<=1548){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18*2+sqre+sqrered-1);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18*2+sqre+sqrered-1);
  }
  else if(1548<temp_time && temp_time<=1551){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.28+sqre+sqrered-1);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.58+sqre+sqrered-1);
  }
  else if(1551<temp_time && temp_time<=1581){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+6.18+sqre*8-1);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos-6.18+sqre*8-1);
  }else if(1581<temp_time && temp_time<=1584){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*8-1);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*8-1);
  }
  else if(1584<temp_time && temp_time<=1634){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(left_pos+sqre*18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(right_pos+sqre*18);
```

```
        }

    }
    else if (1634<temp_time){
        checkMate=true;
    }



}
void onBridge(){
    if (!onBridge1){
        onBridge1=true;
        start_right=pos_right->getValue();
        start_left=pos_left->getValue();
        onBridge_initial_time=time(NULL);
        drop_activate=true;
    }
    double fd_angle = fds_angle->getValue();
    std::cout<<fd_angle<<std::endl;
    std::time_t current_time;
    current_time = time(NULL);
    std::time_t temp_time2 = current_time - onBridge_initial_time;
    std::cout<<temp_time2<<" "<<pos_right->getValue()<<" "<<pos_left->getValue()<<std::endl;
    if (0<temp_time2 && temp_time2<=3){
        leftMotor->setVelocity(0.5*MAX_SPEED);
        leftMotor->setPosition(start_left+6.18);
        rightMotor->setVelocity(0.5*MAX_SPEED);
        rightMotor->setPosition(start_right-6.18);
    }
    else if (3<temp_time2 && temp_time2<=6){
        leftMotor->setVelocity(0.5*MAX_SPEED);
        leftMotor->setPosition(start_left+6.18-2.512);
        rightMotor->setVelocity(0.5*MAX_SPEED);
        rightMotor->setPosition(start_right-6.18-2.512);
    }
    else if (6<temp_time2 && temp_time2<=9){
        leftMotor->setVelocity(0.5*MAX_SPEED);
        leftMotor->setPosition(start_left-2.512);
        rightMotor->setVelocity(0.5*MAX_SPEED);
        rightMotor->setPosition(start_right-2.512);
    }
    else if (9<temp_time2 && temp_time2<=12){
```

```
        leftMotor->setVelocity(0.5*MAX_SPEED);
        leftMotor->setPosition(start_left-2.512);
        rightMotor->setVelocity(0.5*MAX_SPEED);
        rightMotor->setPosition(start_right-2.512);
    }
    else if (12<temp_time2 && temp_time2<=13){
        leftMotor->setVelocity(0.5*MAX_SPEED);
        leftMotor->setPosition(start_left-2.512+1.75-0.35);
        rightMotor->setVelocity(0.5*MAX_SPEED);
        rightMotor->setPosition(start_right-2.512+1.75+0.35);
    }
    else if (13<temp_time2 && temp_time2<=31){
       rightMotor->setVelocity(0.0);
       leftMotor->setVelocity(0.0);
       dropLeft();
    }
    else if (31<temp_time2 && temp_time2<=37){
       leftMotor->setVelocity(0.5*MAX_SPEED);
       leftMotor->setPosition(start_left-2.512+1.6-0.35+11);
       rightMotor->setVelocity(0.5*MAX_SPEED);
       rightMotor->setPosition(start_right-2.512+1.6+0.35-11);
    }







}

void secretChamber(){


    int front_distance;
    if (!chamber_active){
        chamber_active=true;
        start_right=pos_right->getValue();
        start_left=pos_left->getValue();
    }
    if (step1){
        if (step1_initial){
            chamber_initial_time = time(NULL);
            step1_initial=false;
            iteration_starting_left=pos_left->getValue();
```

```
        iteration_starting_right=pos_right->getValue();
    }
    std::time_t current_time;
    current_time = time(NULL);
    std::time_t temp_time = current_time - chamber_initial_time;
    std::cout<<temp_time<<std::endl;
    if (0<temp_time && temp_time<=20){
        leftMotor->setVelocity(0.5*MAX_SPEED);
        leftMotor->setPosition(INFINITY);
        rightMotor->setVelocity(0.5*MAX_SPEED);
        rightMotor->setPosition(INFINITY);
        front_distance = fds->getValue();
        if (front_distance<100){
            leftMotor->setVelocity(0.0);
            rightMotor->setVelocity(0.0);
            gap=((pos_right->getValue())-start_right+(pos_left->getValue())-start_left)/2;
            n=gap/4.712388;
            std::cout <<gap<<" "<<n<< std::endl;
            iterations=(int)n;
            std::cout<<iterations<<std::endl;
        }
    }
    else if (20<temp_time && temp_time<=40){
        leftMotor->setVelocity(0.5*MAX_SPEED);
        leftMotor->setPosition(start_left);
        rightMotor->setVelocity(0.5*MAX_SPEED);
        rightMotor->setPosition(start_right);
        if (temp_time==40){
            step2=true;
            step1=false;
        }
    }
}

if (step2){

    if (r<iterations){

        if (r_true){
            chamber_initial_time2 = time(NULL);
            std::cout<<r<<std::endl;
            r_true=false;
            iteration_starting_left=pos_left->getValue();
            iteration_starting_right=pos_right->getValue();
```

```
}
std::time_t current_time;
current_time = time(NULL);

std::time_t temp_time = current_time - chamber_initial_time2;
std::cout<<temp_time<<std::endl;
if (0<temp_time && temp_time<=4){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(iteration_starting_left+4.712388);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(iteration_starting_right+4.712388);
}
else if(4<temp_time && temp_time<=7){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(iteration_starting_left+4.712388-6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(iteration_starting_right+4.712388+6.18);
}
else if(7<temp_time && temp_time<=15){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(INFINITY);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(INFINITY);

   front_distance = fds->getValue();
   if (front_distance<250){
      leftMotor->setVelocity(0.0);
      rightMotor->setVelocity(0.0);
      front_colour = irf->getValue();
      if (front_colour<1000){
         x=true;
         chamber_initial_time2=chamber_initial_time2-20;
      }
   }
}
else if(15<temp_time && temp_time<=23){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(iteration_starting_left+4.712388-6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(iteration_starting_right+4.712388+6.18);
}
else if(23<temp_time && temp_time<=26){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(iteration_starting_left+4.712388);
```

```
            rightMotor->setVelocity(0.5*MAX_SPEED);
            rightMotor->setPosition(iteration_starting_right+4.712388);
        }
        else if(26<temp_time && temp_time<=27){
            r++;
            r_true=true;
        }
        else if(27<temp_time){
            if (27<temp_time && temp_time<=48){
                pickup();
            }
            else if (48<temp_time && temp_time<=55){
                leftMotor->setVelocity(0.5*MAX_SPEED);
                leftMotor->setPosition(iteration_starting_left+4.712388-6.18);
                rightMotor->setVelocity(0.5*MAX_SPEED);
                rightMotor->setPosition(iteration_starting_right+4.712388+6.18);
            }
            else if (55<temp_time && temp_time<=58){
                leftMotor->setVelocity(0.5*MAX_SPEED);
                leftMotor->setPosition(iteration_starting_left+4.712388);
                rightMotor->setVelocity(0.5*MAX_SPEED);
                rightMotor->setPosition(iteration_starting_right+4.712388);
            }
            else if (58<temp_time && temp_time<=68){
                leftMotor->setVelocity(0.5*MAX_SPEED);
                leftMotor->setPosition(iteration_starting_left+(n-r)*4.712388);//n waarayak gen da
adu karanne gap eken da
                rightMotor->setVelocity(0.5*MAX_SPEED);
                rightMotor->setPosition(iteration_starting_right+(n-r)*4.712388);
            }
            else if (68<temp_time && temp_time<=71){
                leftMotor->setVelocity(0.5*MAX_SPEED);
                leftMotor->setPosition(iteration_starting_left+(n-r)*4.712388+6.18);//n waarayak
gen da adu karanne gap eken da
                rightMotor->setVelocity(0.5*MAX_SPEED);
                rightMotor->setPosition(iteration_starting_right+(n-r)*4.712388-6.18);
            }
            else if (71<temp_time && temp_time<=1000){
                int T;
                leftMotor->setPosition(INFINITY);
                rightMotor->setPosition(INFINITY);
                dc = 0;

                const double* gyroVal = gyroSens->getRollPitchYaw();
```

```
    std::cout << "jgyro val "<< gyroVal[0]  <<std::endl;
    std::cout << "jgyro val "<< gyroVal[1]  <<std::endl;
    std::cout << "jgyro val "<< gyroVal[2]  <<std::endl;

    if (pidOn){

      kp = 0.07;
      ki = 0.005;
      kd = 0.0001;
      M_SPEED=MAX_SPEED*0.4;
      pid();
      }
    pidOn=true;

    // Process sensor data here.
    setMotors();
    if (gyroVal[1]<0){
        bridgeArrival1=true;
    }
    if (bridgeArrival1 && gyroVal[1]>0){
        T=temp_time;
        chamber_initial_time2=chamber_initial_time2-(1000-T);
        start_right=pos_right->getValue();
        start_left=pos_left->getValue();
    }

}
else if (1000<=temp_time && temp_time<=1008){

    leftMotor->setPosition(start_left+2.7);
    rightMotor->setPosition(start_right+2.7);
    dc = 0;
    pidOn=true;
    if (pidOn){

        kp = 0.07;
        ki = 0.005;
        kd = 0.0001;
        M_SPEED=MAX_SPEED*0.4;
        pid();
    }

    setMotors();
}
```

```
else if (1008<temp_time && temp_time<=1045){

   onBridge();
}
else if (1045<temp_time && temp_time<=2000){
   std::cout<<"retrunig"<<std::endl;
   leftMotor->setPosition(INFINITY);
   rightMotor->setPosition(INFINITY);
   dc = 0;
   pidOn=true;
   if (pidOn){

      kp = 0.07;
      ki = 0.005;
      kd = 0.0001;
      M_SPEED=MAX_SPEED*0.4;
      pid();
   }

   setMotors();
   rightDsValue = ds[1]->getValue();
   double colourtest1 = ir[4]->getValue();
   double colourtest2 = ir[3]->getValue();
   if (rightDsValue<1000 &&  colourtest1<1000 && colourtest2<1000){
      chamber_initial_time2=chamber_initial_time2-(1000-temp_time);
      start_right=pos_right->getValue();
      start_left=pos_left->getValue();
   }
}
else if (2000<temp_time && temp_time<=4){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(iteration_starting_left+4.712388);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(iteration_starting_right+4.712388);
}
else if(4<temp_time && temp_time<=7){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(iteration_starting_left+4.712388-6.18);
   rightMotor->setVelocity(0.5*MAX_SPEED);
   rightMotor->setPosition(iteration_starting_right+4.712388+6.18);
}
else if(7<temp_time && temp_time<=15){
   leftMotor->setVelocity(0.5*MAX_SPEED);
   leftMotor->setPosition(INFINITY);
```

```
            rightMotor->setVelocity(0.5*MAX_SPEED);
            rightMotor->setPosition(INFINITY);

            front_distance = fds->getValue();
            if (front_distance<250){
               leftMotor->setVelocity(0.0);
               rightMotor->setVelocity(0.0);
               front_colour = irf->getValue();
               if (front_colour<1000){
                  x=true;
                  chamber_initial_time2=chamber_initial_time2-20;
               }
            }
         }
         else if(15<temp_time && temp_time<=23){
            leftMotor->setVelocity(0.5*MAX_SPEED);
            leftMotor->setPosition(iteration_starting_left+4.712388-6.18);
            rightMotor->setVelocity(0.5*MAX_SPEED);
            rightMotor->setPosition(iteration_starting_right+4.712388+6.18);
         }
         else if(23<temp_time && temp_time<=26){
            leftMotor->setVelocity(0.5*MAX_SPEED);
            leftMotor->setPosition(iteration_starting_left+4.712388);
            rightMotor->setVelocity(0.5*MAX_SPEED);
            rightMotor->setPosition(iteration_starting_right+4.712388);
         }









      }
   }
   /***
   if (step2){
      if (r<iterations){
         if (r_true){
            chamber_initial_time = time(NULL);
            r_true=false;
            iteration_starting_left=pos_left->getValue();
```

```
    iteration_starting_right=pos_right->getValue();

}
std::time_t current_time;
current_time = time(NULL);

std::time_t temp_time = current_time - initial_time2;

if (0<temp_time && temp_time<=4){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(iteration_starting_left+4.712388);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(iteration_starting_right+4.712388);
}
else if(4<temp_time && temp_time<=7){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(iteration_starting_left+4.712388+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(iteration_starting_right+4.712388-6.18);
}
else if(7<temp_time && temp_time<=15){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(INFINITY);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(INFINITY);

    if (front_distance<100){
        leftMotor->setVelocity(0.0);
        rightMotor->setVelocity(0.0);
    }
}
else if(15<temp_time && temp_time<=23){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(iteration_starting_left+4.712388+6.18);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(iteration_starting_right+4.712388-6.18);
}
else if(23<temp_time && temp_time<=26){
    leftMotor->setVelocity(0.5*MAX_SPEED);
    leftMotor->setPosition(iteration_starting_left+4.712388);
    rightMotor->setVelocity(0.5*MAX_SPEED);
    rightMotor->setPosition(iteration_starting_right+4.712388);
}
else if(26<temp_time){
```

```
            r++;
            r_true=true;
          }




      }
    }
    ***/
    }
}
//1m = 31.4pos value
int main(int argc, char **argv) {
  // create the Robot instance.
  Robot *robot = new Robot();

  // get the time step of the current world.
  int timeStep = (int)robot->getBasicTimeStep();



  // You should insert a getDevice-like function in order to get the
  // instance of a device of the robot. Something like:


  for (int i = 0; i < 8; i++) {
    ir[i] = robot->getDistanceSensor("ir" + to_string(i));
    ir[i]->enable(TIME_STEP);
  }
  for (int i = 0; i < 2; i++) {
      ds[i] = robot->getDistanceSensor("ds" + to_string(i));
      ds[i]->enable(TIME_STEP);
  }
  fds = robot->getDistanceSensor("fs");
  fds->enable(TIME_STEP);
  fds_angle = robot->getDistanceSensor("fs_angle");
  fds_angle->enable(TIME_STEP);
  irf = robot->getDistanceSensor("irf");
  irf->enable(TIME_STEP);
  ds_top = robot->getDistanceSensor("ds_top");
  ds_top->enable(TIME_STEP);
  left_brk = robot->getBrake("brake_left");
```

```
right_brk = robot->getBrake("brake_right");
left_brk->setDampingConstant(0);
right_brk->setDampingConstant(0);
pos_left = robot->getPositionSensor("pos_left");
pos_right = robot->getPositionSensor("pos_right");
pos_left->enable(TIME_STEP);
pos_right->enable(TIME_STEP);
gyroSens = robot->getInertialUnit("imu");
gyroSens->enable(TIME_STEP);
leftMotor = robot->getMotor("motor_left");
rightMotor = robot->getMotor("motor_right");
left_linear_motor1 = robot->getMotor("linear_motor1");
left_linear_motor2 = robot->getMotor("linear_motor2");
left_linear_motor3 = robot->getMotor("linear_motor3");
right_linear_motor1 = robot->getMotor("linear_motor4");
right_linear_motor2 = robot->getMotor("linear_motor5");
right_linear_motor3 = robot->getMotor("linear_motor6");
leftMotor->setPosition(INFINITY);
rightMotor->setPosition(INFINITY);
leftMotor->setVelocity(0.0);
rightMotor->setVelocity(0.0);
std::cout << "Motor state = line follow"<< std::endl;
//leftMotor->setVelocity(0.1 * MAX_SPEED);
//rightMotor->setVelocity(0.1 * MAX_SPEED);
//  DistanceSensor *ds = robot->getDistanceSensor("dsname");
//  ds->enable(timeStep);

// Main loop:
// - perform simulation steps until Webots is stopping the controller
while (robot->step(timeStep) != -1) {


  double front_distance = fds->getValue();
  double front_colour = irf->getValue();
  double left_distance = ds[0]->getValue();
  double right_distance = ds[1]->getValue();
  double colourtest1 = ir[4]->getValue();
  double colourtest2 = ir[7]->getValue();


  std::cout<<"fds"<<front_distance<< std::endl;
  std::cout<<"irf"<<front_colour<< std::endl;
  if (!chessActivate){
     right_pos=pos_right->getValue();
```

```
    left_pos=pos_left->getValue();
  }
  std::cout << "left_pos :" << left_pos << std::endl;
  std::cout << "right_pos :" << right_pos << std::endl;
  std::cout << "chessStart :" << chessStart<< std::endl;
  std::cout << "chessActivate :" << chessActivate<< std::endl;
  if (left_pos>=chessStart-0.0001 && right_pos>=chessStart-0.0001 && !z){
     chessActivate=true;
  }
  if ((left_distance<1000) && (right_distance<1000) && z){

     std::cout<<"ir4"<<colourtest1<< std::endl;
     if ((colourtest1>950) && (colourtest1<990)){


        chessBoardEntrance();
     }
  }
  else if (chessActivate && ((colourtest1<1000 && colourtest2<1000)||playingChess)){
     chessBoard();
  }

  // Read the sensors:
  // Enter here functions to read sensor data, like:
  else{
     dc = 0;

     if (pidOn){

       kp = 0.07;
       ki = 0.005;
       kd = 0.0001;
       M_SPEED=MAX_SPEED*0.4;
       pid();
       }
     pidOn=true;
     wall();
     // Process sensor data here.
     setMotors();
     // Enter here functions to send actuator commands, like:
     //  motor->setPosition(10.0);
  }

};
```

```
  // Enter here exit cleanup code.

  delete robot;
  return 0;
}
//A8 eka paththen check mate karoth yana paara hadanna
```