НП „ИТ КАРИЕРА “

Модул 8 - Разработка на софтуер

***Проект BLACKJACK***

Проект : <https://www.tinkercad.com/things/5Mb8hGeeCPl>

Github :

Изготвил : Гани Темур

От ПМГ“Академик Боян Петканчин“

Съдържание

[1. Описание на проекта: 2](#_Toc69398436)

[1.1 Какво предствлява 2](#_Toc69398437)

[1.2 Компоненти и употреба 3](#_Toc69398438)

[LCD 3](#_Toc69398439)

[Keypad 4x4 5](#_Toc69398440)

[7 Segment Display(cards) 5](#_Toc69398441)

[2. Блокова схема 8](#_Toc69398442)

[3. Електрическа схема 9](#_Toc69398443)

[4. Списък съставни части 9](#_Toc69398444)

[5. Сорс код – описание на функционалността 9](#_Toc69398445)

[6. Заключение 68](#_Toc69398446)

[6.1. Употреби 68](#_Toc69398447)

[6.2. Заключение 69](#_Toc69398448)

# 

# 1. Описание на проекта:

## 1.1 Какво предствлява

Проектът е разработен в webapp-а Tinkercad, който представлява симулатор на електрически вериги.

Проекта е симулация на играта blackjack. [Правила на blackjack.](http://www.hitorstand.net/strategy.php) Играча започва с 500 резултат и неговата задача е да увеличи този резултат колкото може. Ползваме 7 сегментен дисплей за изписването на различните изтеглени карти. Ползваме лсиди за изписване на сумата и залога или залога и резултатите. Лсидито има потенциометър за контраста. А за инпут ползваме 4x4 кей пад, но са ни нужни само 4x3. И няма как да забравим използваме Arduino Uno R3 за всичката логика и код.

Цялата функционалност се упрвлява от компютъра на Arduino-то, което ползва [кода написан от мене](#_5._Сорс_код). Този код се занимава с регулирането на баланса и залога на играча. Той не позволява да се салага повече от колкото има. Кода също създава тесте и го разбърква. После тегли крати за играча и компютъра. След това чака команда от играча и я изпълнява. След това компютъра се опитва да победи играча. След това според резултата се увеличава или намаля баланса на играча. Накрая се връща в началото и чака играча за нов залог.

## 1.2 Компоненти и употреба

|  |  |
| --- | --- |
| LCD | |
| Diagram, schematic  Description automatically generated | От ляво се намира потенциометъра, който служи за контраста. Дисплея е сетнат на 16х2. При победа, равенство или загуба ще се изпиши определеното съобщение. |
| A picture containing text, electronics  Description automatically generated | При начало на програмата на дисплея ще се изпиши 500. Това е баланса с който разполага играча. А долу ще изпиши залога. Залога не може да бъде по-голям от баланса. Баланса и залога са лонг. Максималния лонг е 2,147,483,647. Ако се надвиши става отрицателно число. Но ако стигнете до това число означава че сте професионалист на бляк джак. |
| Diagram  Description automatically generated | При вече въведен залог се изписва този дисплей. Отгоре се изписва залога който е въвел играча. Отлъво се изписва ресултата на играча, а отдясно се изписва резултата на компютъра. Резултата е разделен на две. Отляво на точката и запетаята се изписва резултата ако смятаме асака за 11, а от дясно ако смятаме аска за 1. При случай че лявата сума премине 21, тогава лявата сума ще се приравни на дясната. |

|  |  |
| --- | --- |
| Keypad 4x4 | |
| A picture containing text, electronics, calculator  Description automatically generated | Заради недостига на пинове трябва да ползваме кейпада 4х3 вместо 4х4. Използваните клавиши са 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, \*, #. Докато играча още решава колко да залага цифрените клавиши са за съответните цифри. А клавиша \* се използва за да се изчисти каквато и да е въведена сума. А клавиша # се използва за да се потвърди сумата и за започване на играта.  При започната игра цифровите клавиши не извършват никаква функция. Клавиша \* се ползва за теглене на карта. А клавиша # се ползва за холд(играча казва че няма да тегли повече карти). |

|  |  |  |  |
| --- | --- | --- | --- |
| 7 Segment Display(cards) | | | |
| Name | 7 Segment  Display | Card | Points |
| Ace | A picture containing text, clock  Description automatically generated | Logo, icon, company name  Description automatically generated | 1 or 11 |
| Two | A picture containing text, clock, gauge  Description automatically generated | Icon  Description automatically generated with medium confidence | 2 |
| Three | A picture containing text, clock, gauge  Description automatically generated | Icon  Description automatically generated | 3 |
| Four | A picture containing text, clock  Description automatically generated | Icon  Description automatically generated with medium confidence | 4 |
| Five | A picture containing text, clock, gauge  Description automatically generated | Shape, arrow  Description automatically generated | 5 |
| Six | A picture containing text, clock, gauge  Description automatically generated | A picture containing text, queen, gallery, vector graphics  Description automatically generated | 6 |
| Seven | A picture containing text, clock, gauge  Description automatically generated | Icon  Description automatically generated | 7 |
| Eight | A picture containing text, clock  Description automatically generated | Shape, arrow  Description automatically generated | 8 |
| Nine | A picture containing text, clock, gauge  Description automatically generated | Shape  Description automatically generated | 9 |
| Ten | A picture containing text, clock, gauge  Description automatically generated | Shape  Description automatically generated | 10 |
| Jack | A picture containing text, clock  Description automatically generated | A picture containing text, queen  Description automatically generated | 10 |
| Queen | A picture containing text, clock  Description automatically generated | A picture containing text, queen  Description automatically generated | 10 |
| King | A picture containing text, clock, gauge  Description automatically generated | A picture containing text, queen  Description automatically generated | 10 |

# 2. Блокова схема

Diagram

Description automatically generated

# 3. Електрическа схема

Chart, diagram

Description automatically generated with medium confidence

# 4. Списък съставни части

Table

Description automatically generated

# 5. Сорс код – описание на функционалността

#include <LiquidCrystal.h>

//Cteate the the lcd

LiquidCrystal lcd(13,12,11,10,9,8);

#include <Keypad.h>

//Create the rows and columns of the keypad

const byte rows = 4;

const byte cols = 3;

//create an array in which every element responds to keys from the keypad

char hexkeypad[rows][cols] =

{

{'1','2','3'},

{'4','5','6'},

{'7','8','9'},

{'\*','0','#'}

};

//create two arrays with the supposed pins for the keypad

byte rowspins[rows] = {0,A5,A4,A3};

byte colspins[cols] = {A2,A1,A0};

//create a keypad with all the above created veriable

Keypad kpd = Keypad(makeKeymap(hexkeypad), rowspins, colspins, rows, cols);

//create an array with all the playing cards

char\* deck[]= { "ACE", "TWO", "THREE", "FOUR", "FIVE", "SIX", "SEVEN", "EIGHT", "NINE", "TEN", "JACK", "QUEEN", "KING", "ACE", "TWO", "THREE", "FOUR", "FIVE", "SIX", "SEVEN", "EIGHT", "NINE", "TEN", "JACK", "QUEEN", "KING", "ACE", "TWO", "THREE", "FOUR", "FIVE", "SIX", "SEVEN", "EIGHT", "NINE", "TEN", "JACK", "QUEEN", "KING", "ACE", "TWO", "THREE", "FOUR", "FIVE", "SIX", "SEVEN", "EIGHT", "NINE", "TEN", "JACK", "QUEEN", "KING"};

// shuffle the cards array into random order

void shuffle()

{

// Start from the last element and swap one by one. We don't

// need to run for the first element that's why i > 0

for (int i = 51; i > 0; i--)

{

// Pick a random index from 0 to i

int j = random(i+1);

// Swap cards[i] with the element at random index

char\* temp = deck[i];

deck[i] = deck[j];

deck[j] = temp;

}

}

void setup()

{

//set the pis for the 7 segment display, which will display our cards, as output

pinMode(1,OUTPUT);

pinMode(2,OUTPUT);

pinMode(3,OUTPUT);

pinMode(4,OUTPUT);

pinMode(5,OUTPUT);

pinMode(6,OUTPUT);

pinMode(7,OUTPUT);

//set the lcd to be 16x2

lcd.begin(16,2);

}

//create the amount the person will have

long balance = 500;

void loop()

{

//reset the 7 segment display at the begining

digitalWrite(1,LOW);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,LOW);

digitalWrite(7,LOW);

//reset the lcd and write the balance of the player

lcd.clear();

lcd.print(balance);

//reset the scores and betting

int playerScoreWithAces = 0;

int playerScoreWithoutAces = 0;

int computerScoreWithAces = 0;

int computerScoreWithoutAces = 0;

int computerHiddenScoreWithAces = 0;

int computerHiddenScoreWithoutAces = 0;

long betting = 0;

bool gameStart = false;

//creates a loop until the player is ready with betting

while(!gameStart)

{

//get the pressed key

char keypressed = kpd.getKey();

if(keypressed)

{

//when \* is pressed reset the bet

if(keypressed == '\*')

{

betting = 0;

}

//when # is pressed start the game, only if the bet is smaller or equal to the balance

else if(keypressed == '#' && betting <= balance)

{

gameStart = true;

}

else if(keypressed == '#'){}

//increases the bet with the pressed key

else

{

betting = betting \* 10;

betting = betting + keypressed - 48;

}

//print the balance and the current bet

lcd.clear();

lcd.print(balance);

lcd.setCursor(0,1);

lcd.print(betting);

}

}

//shuffles the deck of cards

shuffle();

//print the first card of the player and add it to the score

//checks if the card is ace

if(deck[0] == "ACE")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = 11;

playerScoreWithoutAces = 1;

}

//checks if the card is two

if(deck[0] == "TWO")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = 2;

playerScoreWithoutAces = 2;

}

//checks if the card is three

if(deck[0] == "THREE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = 3;

playerScoreWithoutAces = 3;

}

//checks if the card is four

if(deck[0] == "FOUR")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

playerScoreWithAces = 4;

playerScoreWithoutAces = 4;

}

//checks if the card is five

if(deck[0] == "FIVE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

playerScoreWithAces = 5;

playerScoreWithoutAces = 5;

}

//checks if the card is six

if(deck[0] == "SIX")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

playerScoreWithAces = 6;

playerScoreWithoutAces = 6;

}

//checks if the card is seven

if(deck[0] == "SEVEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = 7;

playerScoreWithoutAces = 7;

}

//checks if the card is eight

if(deck[0] == "EIGHT")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = 8;

playerScoreWithoutAces = 8;

}

//checks if the card is nine

if(deck[0] == "NINE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = 9;

playerScoreWithoutAces = 9;

}

//checks if the card is ten

if(deck[0] == "TEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = 10;

playerScoreWithoutAces = 10;

}

//checks if the card is jack

if(deck[0] == "JACK")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = 10;

playerScoreWithoutAces = 10;

}

//checks if the card is queen

if(deck[0] == "QUEEN")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

playerScoreWithAces = 10;

playerScoreWithoutAces = 10;

}

//checks if the card is king

if(deck[0] == "KING")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

playerScoreWithAces = 10;

playerScoreWithoutAces = 10;

}

//print the amount that has been bet as well as the score of the player

lcd.clear();

lcd.print(betting);

lcd.setCursor(0,1);

lcd.print(playerScoreWithAces);

lcd.print(";");

lcd.print(playerScoreWithoutAces);

delay(2000);

//prints the first card of the computer and adds it to the scores

//checks if the card is ace

if(deck[1] == "ACE")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = 11;

computerScoreWithoutAces = 1;

computerHiddenScoreWithAces = 11;

computerHiddenScoreWithoutAces = 1;

}

//checks if the card is two

if(deck[1] == "TWO")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = 2;

computerScoreWithoutAces = 2;

computerHiddenScoreWithAces = 2;

computerHiddenScoreWithoutAces = 2;

}

//checks if the card is three

if(deck[1] == "THREE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = 3;

computerScoreWithoutAces = 3;

computerHiddenScoreWithAces = 3;

computerHiddenScoreWithoutAces = 3;

}

//checks if the card is four

if(deck[1] == "FOUR")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerScoreWithAces = 4;

computerScoreWithoutAces = 4;

computerHiddenScoreWithAces = 4;

computerHiddenScoreWithoutAces = 4;

}

//checks if the card is five

if(deck[1] == "FIVE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

computerScoreWithAces = 5;

computerScoreWithoutAces = 5;

computerHiddenScoreWithAces = 5;

computerHiddenScoreWithoutAces = 5;

}

//checks if the card is six

if(deck[1] == "SIX")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

computerScoreWithAces = 6;

computerScoreWithoutAces = 6;

computerHiddenScoreWithAces = 6;

computerHiddenScoreWithoutAces = 6;

}

//checks if the card is seven

if(deck[1] == "SEVEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = 7;

computerScoreWithoutAces = 7;

computerHiddenScoreWithAces = 7;

computerHiddenScoreWithoutAces = 7;

}

//checks if the card is eight

if(deck[1] == "EIGHT")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = 8;

computerScoreWithoutAces = 8;

computerHiddenScoreWithAces = 8;

computerHiddenScoreWithoutAces = 8;

}

//checks if the card is nine

if(deck[1] == "NINE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = 9;

computerScoreWithoutAces = 9;

computerHiddenScoreWithAces = 9;

computerHiddenScoreWithoutAces = 9;

}

//checks if the card is ten

if(deck[1] == "TEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = 10;

computerScoreWithoutAces = 10;

computerHiddenScoreWithAces = 10;

computerHiddenScoreWithoutAces = 10;

}

//checks if the card is jack

if(deck[1] == "JACK")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = 10;

computerScoreWithoutAces = 10;

computerHiddenScoreWithAces = 10;

computerHiddenScoreWithoutAces = 10;

}

//checks if the card is queen

if(deck[1] == "QUEEN")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

computerScoreWithAces = 10;

computerScoreWithoutAces = 10;

computerHiddenScoreWithAces = 10;

computerHiddenScoreWithoutAces = 10;

}

//checks if the card is king

if(deck[1] == "KING")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerScoreWithAces = 10;

computerScoreWithoutAces = 10;

computerHiddenScoreWithAces = 10;

computerHiddenScoreWithoutAces = 10;

}

//prints the score of the computer

lcd.setCursor(11,1);

lcd.print(computerScoreWithAces);

lcd.print(";");

lcd.print(computerHiddenScoreWithoutAces);

delay(2000);

//prints the second card of the player and adds it to the scores

//checks if the card is ace

if(deck[2] == "ACE")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

//depending if you already have ace adds different scores

//if you have ace both scores are increased by 1

if(playerScoreWithAces != playerScoreWithoutAces)

{

playerScoreWithAces = playerScoreWithAces + 1;

playerScoreWithoutAces = playerScoreWithoutAces + 1;

}

//if you don't have an ace scores are increased with 11 or 1 supposedto the score

else

{

playerScoreWithAces = playerScoreWithAces + 11;

playerScoreWithoutAces = playerScoreWithoutAces + 1;

}

}

//checks if the card is two

if(deck[2] == "TWO")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 2;

playerScoreWithoutAces = playerScoreWithoutAces + 2;

}

//checks if the card is three

if(deck[2] == "THREE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 3;

playerScoreWithoutAces = playerScoreWithoutAces + 3;

}

//checks if the card is four

if(deck[2] == "FOUR")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 4;

playerScoreWithoutAces = playerScoreWithoutAces + 4;

}

//checks if the card is five

if(deck[2] == "FIVE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

playerScoreWithAces = playerScoreWithAces + 5;

playerScoreWithoutAces = playerScoreWithoutAces + 5;

}

//checks if the card is six

if(deck[2] == "SIX")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

playerScoreWithAces = playerScoreWithAces + 6;

playerScoreWithoutAces = playerScoreWithoutAces + 6;

}

//checks if the card is seven

if(deck[2] == "SEVEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 7;

playerScoreWithoutAces = playerScoreWithoutAces + 7;

}

//checks if the card is eight

if(deck[2] == "EIGHT")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 8;

playerScoreWithoutAces = playerScoreWithoutAces + 8;

}

//checks if the card is nine

if(deck[2] == "NINE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 9;

playerScoreWithoutAces = playerScoreWithoutAces + 9;

}

//checks if the card is ten

if(deck[2] == "TEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 10;

playerScoreWithoutAces = playerScoreWithoutAces + 10;

}

//checks if the card is jack

if(deck[2] == "JACK")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 10;

playerScoreWithoutAces = playerScoreWithoutAces + 10;

}

//checks if the card is queen

if(deck[2] == "QUEEN")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

playerScoreWithAces = playerScoreWithAces + 10;

playerScoreWithoutAces = playerScoreWithoutAces + 10;

}

//checks if the card is king

if(deck[2] == "KING")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 10;

playerScoreWithoutAces = playerScoreWithoutAces + 10;

}

//prints the bet and the new scores

lcd.clear();

lcd.print(betting);

lcd.setCursor(0,1);

lcd.print(playerScoreWithAces);

lcd.print(";");

lcd.print(playerScoreWithoutAces);

lcd.setCursor(11,1);

lcd.print(computerScoreWithAces);

lcd.print(";");

lcd.print(computerScoreWithoutAces);

delay(2000);

//prints that the second card of the computer is hodden and adds it to the hidden score

//checks if the card is ace

if(deck[3] == "ACE")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

//depending if you already have ace adds different scores

//if you have ace both scores are increased by 1

if(computerScoreWithAces != computerScoreWithoutAces)

{

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 1;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 1;

}

//if you don't have an ace scores are increased with 11 or 1 supposed to the score

else

{

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 11;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 1;

}

}

//checks if the card is two

if(deck[3] == "TWO")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 2;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 2;

}

//checks if the card is three

if(deck[3] == "THREE")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 3;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 3;

}

//checks if the card is four

if(deck[3] == "FOUR")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 4;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 4;

}

//checks if the card is five

if(deck[3] == "FIVE")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 5;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 5;

}

//checks if the card is six

if(deck[3] == "SIX")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 6;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 6;

}

//checks if the card is seven

if(deck[3] == "SEVEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 7;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 7;

}

//checks if the card is eight

if(deck[3] == "EIGHT")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 8;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 8;

}

//checks if the card is nine

if(deck[3] == "NINE")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 9;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 9;

}

//checks if the card is ten

if(deck[3] == "TEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 10;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 10;

}

//checks if the card is jack

if(deck[3] == "JACK")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 10;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 10;

}

//checks if the card is queen

if(deck[3] == "QUEEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 10;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 10;

}

//checks if the card is king

if(deck[3] == "KING")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerHiddenScoreWithAces = computerHiddenScoreWithAces + 10;

computerHiddenScoreWithoutAces = computerHiddenScoreWithoutAces + 10;

}

delay(2000);

//creates a bool for when the player looses

bool playerLost = false;

//creates a bool fow when the player is done pulling new card

bool continueGame = false;

//int that keeps track of the current card

int currentCard = 4;

//loop that continues until the player is done with pulling cards or the player looses

while(!continueGame && !playerLost)

{

//reads the pressed key

char keypressed = kpd.getKey();

if(keypressed)

{

//if \* is pressed the player pulls a card, then that card is displayed and added to the score

if(keypressed == '\*')

{

//checks if the card is ace

if(deck[currentCard] == "ACE")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

//depending if you already have ace adds different scores

//if you have ace both scores are increased by 1

if(playerScoreWithAces != playerScoreWithoutAces)

{

playerScoreWithAces = playerScoreWithAces + 1;

playerScoreWithoutAces = playerScoreWithoutAces + 1;

}

//if you don't have an ace scores are increased with 11 or 1 supposed to the score

else

{

playerScoreWithAces = playerScoreWithAces + 11;

playerScoreWithoutAces = playerScoreWithoutAces + 1;

}

}

//checks if the card is two

if(deck[currentCard] == "TWO")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 2;

playerScoreWithoutAces = playerScoreWithoutAces + 2;

}

//checks if the card is three

if(deck[currentCard] == "THREE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 3;

playerScoreWithoutAces = playerScoreWithoutAces + 3;

}

//checks if the card is four

if(deck[currentCard] == "FOUR")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 4;

playerScoreWithoutAces = playerScoreWithoutAces + 4;

}

//checks if the card is five

if(deck[currentCard] == "FIVE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

playerScoreWithAces = playerScoreWithAces + 5;

playerScoreWithoutAces = playerScoreWithoutAces + 5;

}

//checks if the card is six

if(deck[currentCard] == "SIX")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

playerScoreWithAces = playerScoreWithAces + 6;

playerScoreWithoutAces = playerScoreWithoutAces + 6;

}

//checks if the card is seven

if(deck[currentCard] == "SEVEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 7;

playerScoreWithoutAces = playerScoreWithoutAces + 7;

}

//checks if the card is eight

if(deck[currentCard] == "EIGHT")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 8;

playerScoreWithoutAces = playerScoreWithoutAces + 8;

}

//checks if the card is nine

if(deck[currentCard] == "NINE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 9;

playerScoreWithoutAces = playerScoreWithoutAces + 9;

}

//checks if the card is ten

if(deck[currentCard] == "TEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 10;

playerScoreWithoutAces = playerScoreWithoutAces + 10;

}

//checks if the card is jack

if(deck[currentCard] == "JACK")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 10;

playerScoreWithoutAces = playerScoreWithoutAces + 10;

}

//checks if the card is queen

if(deck[currentCard] == "QUEEN")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

playerScoreWithAces = playerScoreWithAces + 10;

playerScoreWithoutAces = playerScoreWithoutAces + 10;

}

//checks if the card is king

if(deck[currentCard] == "KING")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

playerScoreWithAces = playerScoreWithAces + 10;

playerScoreWithoutAces = playerScoreWithoutAces + 10;

}

//clears the lcd, after that prints the bet and the player and computer scores

lcd.clear();

lcd.print(betting);

lcd.setCursor(0,1);

lcd.print(playerScoreWithAces);

lcd.print(";");

lcd.print(playerScoreWithoutAces);

lcd.setCursor(11,1);

lcd.print(computerScoreWithAces);

lcd.print(";");

lcd.print(computerScoreWithoutAces);

delay(2000);

//checks if player lost

if(playerScoreWithAces > 21 && playerScoreWithoutAces > 21)

{

playerLost = true;

}

//checks if ace can not be counted as 11 anymore

else if(playerScoreWithAces > 21)

{

playerScoreWithAces = playerScoreWithoutAces;

}

//goes to the next card

currentCard++;

}

//if # is pressed the game continues

else if(keypressed == '#')

{

continueGame = true;

}

}

}

//if player lost his balance is decreased by the bet and on the lcd its printed that the player has lost

if(playerLost == true)

{

balance = balance - betting;

lcd.clear();

lcd.print("Player lost!");

delay(2000);

}

//if the player has not lost, the computer will try to beat him

else

{

//the hidden card is revealed

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

delay(2000);

//checks if the card is ace

if(deck[3] == "ACE")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

}

//checks if the card is two

if(deck[3] == "TWO")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

}

//checks if the card is three

if(deck[3] == "THREE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

}

//checks if the card is four

if(deck[3] == "FOUR")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

}

//checks if the card is five

if(deck[3] == "FIVE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

}

//checks if the card is six

if(deck[3] == "SIX")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

}

//checks if the card is seven

if(deck[3] == "SEVEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

}

//checks if the card is eight

if(deck[3] == "EIGHT")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

}

//checks if the card is nine

if(deck[3] == "NINE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

}

//checks if the card is ten

if(deck[3] == "TEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

}

//checks if the card is jack

if(deck[3] == "JACK")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

}

//checks if the card is queen

if(deck[3] == "QUEEN")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

}

//checks if the card is king

if(deck[3] == "KING")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

}

//the nomal computer scores are changed to the hidden once

computerScoreWithAces = computerHiddenScoreWithAces;

computerScoreWithoutAces = computerHiddenScoreWithoutAces;

//lcd is cleared after that the bet is printed as well as the new scores

lcd.clear();

lcd.print(betting);

lcd.setCursor(0,1);

lcd.print(playerScoreWithAces);

lcd.print(";");

lcd.print(playerScoreWithoutAces);

lcd.setCursor(11,1);

lcd.print(computerScoreWithAces);

lcd.print(";");

lcd.print(computerScoreWithoutAces);

delay(2000);

//create a new bool if the computer has lost

bool ifComputerLost = false;

//loop that continues until the computer either defeats the play or the computer losses

while(playerScoreWithAces > computerScoreWithAces && !ifComputerLost)

{

//checks if the card is ace

if(deck[currentCard] == "ACE")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

//depending if you already have ace adds different scores

//if you have ace both scores are increased by 1

if(playerScoreWithAces != playerScoreWithoutAces)

{

computerScoreWithAces = computerScoreWithAces + 1;

playerScoreWithoutAces = playerScoreWithoutAces + 1;

}

//if you don't have an ace scores are increased with 11 or 1 supposed to the score

else

{

computerScoreWithAces = computerScoreWithAces + 11;

computerScoreWithoutAces = computerScoreWithoutAces + 1;

}

}

//checks if the card is two

if(deck[currentCard] == "TWO")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = computerScoreWithAces + 2;

computerScoreWithoutAces = computerScoreWithoutAces + 2;

}

//checks if the card is three

if(deck[currentCard] == "THREE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = computerScoreWithAces + 3;

computerScoreWithoutAces = computerScoreWithoutAces + 3;

}

//checks if the card is four

if(deck[currentCard] == "FOUR")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerScoreWithAces = computerScoreWithAces + 4;

computerScoreWithoutAces = computerScoreWithoutAces + 4;

}

//checks if the card is five

if(deck[currentCard] == "FIVE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

computerScoreWithAces = computerScoreWithAces + 5;

computerScoreWithoutAces = computerScoreWithoutAces + 5;

}

//checks if the card is six

if(deck[currentCard] == "SIX")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

computerScoreWithAces = computerScoreWithAces + 6;

computerScoreWithoutAces = computerScoreWithoutAces + 6;

}

//checks if the card is seven

if(deck[currentCard] == "SEVEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,LOW);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = computerScoreWithAces + 7;

computerScoreWithoutAces = computerScoreWithoutAces + 7;

}

//checks if the card is eight

if(deck[currentCard] == "EIGHT")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = computerScoreWithAces + 8;

computerScoreWithoutAces = computerScoreWithoutAces + 8;

}

//checks if the card is nine

if(deck[currentCard] == "NINE")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = computerScoreWithAces + 9;

computerScoreWithoutAces = computerScoreWithoutAces + 9;

}

//checks if the card is ten

if(deck[currentCard] == "TEN")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = computerScoreWithAces + 10;

computerScoreWithoutAces = computerScoreWithoutAces + 10;

}

//checks if the card is jack

if(deck[currentCard] == "JACK")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,LOW);

digitalWrite(4,LOW);

digitalWrite(5,LOW);

digitalWrite(6,HIGH);

digitalWrite(7,HIGH);

computerScoreWithAces = computerScoreWithAces + 10;

computerScoreWithoutAces = computerScoreWithoutAces + 10;

}

//checks if the card is queen

if(deck[currentCard] == "QUEEN")

{

digitalWrite(1,LOW);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,HIGH);

digitalWrite(5,HIGH);

digitalWrite(6,HIGH);

digitalWrite(7,LOW);

computerScoreWithAces = computerScoreWithAces + 10;

computerScoreWithoutAces = computerScoreWithoutAces + 10;

}

//checks if the card is king

if(deck[currentCard] == "KING")

{

digitalWrite(1,HIGH);

digitalWrite(2,HIGH);

digitalWrite(3,HIGH);

digitalWrite(4,LOW);

digitalWrite(5,HIGH);

digitalWrite(6,LOW);

digitalWrite(7,HIGH);

computerScoreWithAces = computerScoreWithAces + 10;

computerScoreWithoutAces = computerScoreWithoutAces + 10;

}

//clear the lcd and print the bet and the new scores

lcd.clear();

lcd.print(betting);

lcd.setCursor(0,1);

lcd.print(playerScoreWithAces);

lcd.print(";");

lcd.print(playerScoreWithoutAces);

lcd.setCursor(11,1);

lcd.print(computerScoreWithAces);

lcd.print(";");

lcd.print(computerScoreWithoutAces);

delay(2000);

//checks if the computer has lost

if(computerScoreWithAces > 21 && computerScoreWithoutAces > 21)

{

ifComputerLost = true;

}

//checks if the computer can no longer use ace as 11

else if(computerScoreWithAces > 21)

{

computerScoreWithAces = computerScoreWithoutAces;

}

}

//if the computer has lost adds the bet to the players balance and prints the supposed message

if(ifComputerLost)

{

balance = balance + betting;

lcd.clear();

lcd.print("Player won!!!");

delay(2000);

}

//if its a draw nothing happens to the players ballance and prints the supposed message

else if(computerScoreWithAces == playerScoreWithAces)

{

lcd.clear();

lcd.print("Player drew!");

delay(2000);

}

//if the computer won the players balance is decreased by the bet and prints the supposed message

else if(computerScoreWithAces > playerScoreWithAces)

{

balance = balance - betting;

lcd.clear();

lcd.print("Player lost!");

delay(2000);

}

}

}

# 6. Заключение

## 6.1. Употреби

Проекта може да се ползва за направата на малка преносима игра (подобна на геймбой) или за голяма машина в казино зависи целта.

## 6.2. Заключение

Проекта е игра изизкваща математика и логическо мислене. За да се играе не се изисква нищо освен проекта и също не изисква втори играч защото играча играе срещу компютъра. Докато изисква умвствена дейност тя е и забавна и развлекателна. Всеки може да се забавлява!