



# Dice BlackJack

*Pervasive Computing Project*

**Team members:**

*Milo Lurati (2702919)*

*Tahsin Kamil Simsek (2695517)*

	<b>DICE BLACKJACK</b>	
<input type="button" value="Throw"/>		Money <input type="text" value="1250"/>
<input type="button" value="Start"/>		Bet <input type="text" value="100"/>
<input type="button" value="Quit"/>		Player status <input type="text" value="21"/>
		Dealer status <input type="text" value="18"/>

**You win this hand!**

## 1. Introduction

Everybody likes to play games, but we do not always have a companion to play them with! Think about all the elderly that love to play blackjack but do not have a dealer and partners to play with. It is kind of sad, isn't it? Think of all the casinos that have these gambling games where you play with a machine. But have you ever seen a machine in a casino to play dice BlackJack? We haven't!

Here is where our pervasive computing skills come in place. We decided to create a system where you can play with physical dice. The system recognizes the numbers and keeps count of the score, and of how much money you have left. The system will also have a virtual dealer that will follow you through the full experience of Dice BlackJack.

This system can be implemented in an elderly home to keep company to the residents that love BlackJack. Also, it could become a new category of gambling machine for casinos that have not yet implemented dice BlackJack in their selection of games.

In this document, we will go through all the different steps that we will take and that we will be needing to create such a pervasive system. From a literature background study to the testing and evaluation of the product.

## 2. Game rules

Dice BlackJack is very similar to BlackJack with cards. The end goal of this game is to get as closer to 21 as possible. There are two main roles: the players and the dealer. At each round, both the player and the dealer try to get closer to 21. Who gets the highest score, that does not go higher than 21, wins the round.

The player starts throwing the dice first. The first two dice rolls are summed together and multiplied by two. Then the player can decide to continue to roll or not. If the player decides to continue rolling, he rolls one dice at a time. Once he decides to stop, it is the turn of the dealer.

The dealer's turn has the same process as the player's turn, but with the difference that once the score passes 17, he stops and can not roll again.

## 3. System overview and general design

The system will be equipped with a camera, a microphone, and a visual interface to interact with the program. The program will start off with a description of the rules and an interactive feature to input the money/points that the player wants to start off with. Then it will give instructions on when and how the player has to play her/his dice. The program will calculate and interact with the player in the different steps of each turn, with speech recognition and image recognition. Once the player has finished his turn, the program will simulate the dealer's dice play, and calculate the outcome of the round. The program will then ask the player if he wants to continue playing, and if the player has finished her/his money or her/his points, the program will terminate the game.

## 4. System requirements

In this section, we elaborate on the different requirements to build such a system. We divide the requirements into functional and non-functional. Then we continue by describing the different constraints and conclude by prioritizing the requirements using the MoSCoW approach.

### 4.1 Functional requirements

- The system shall have the capability to record audio.
- The system shall have the capability to take photos.
- The system shall have the capability to process audio into 12 mfcc coefficients.
- The system shall have the capability to recognize a word through a trained neural network function with the 12 mfcc coefficients.
- The system shall have the capability to process images.
- The system shall have the capability to interact with the user. (GUI)
- 

### 4.2 Non-functional requirements

- The system shall determine the output of each step within 3 seconds.
- The system shall recognize the number of the rolled dice with an accuracy of at least 80%.
- The system shall recognize the voice commands with an accuracy of at least 80%.
- The system shall not compromise user privacy.

### 4.3 Constraints

- The system must be developed in MatLab with its toolboxes, and C++ if needed.
- Input must be audio and image.
- The system has to use pattern recognition and signal processing.
- The output has to be visual or audio.
- The system must be ready for the deadline date of the project.

### 4.4 Prioritization of requirements using the MoSCoW approach

**Must** have:

- The ability to take photos.
- The ability to record sound.
- The ability to process images.
- The ability to process audio.
- The ability to at least interact with the user visually through a terminal.

**Should** have:

- The ability to keep track of the user's money or score.
- The ability to interact with the user through a GUI.

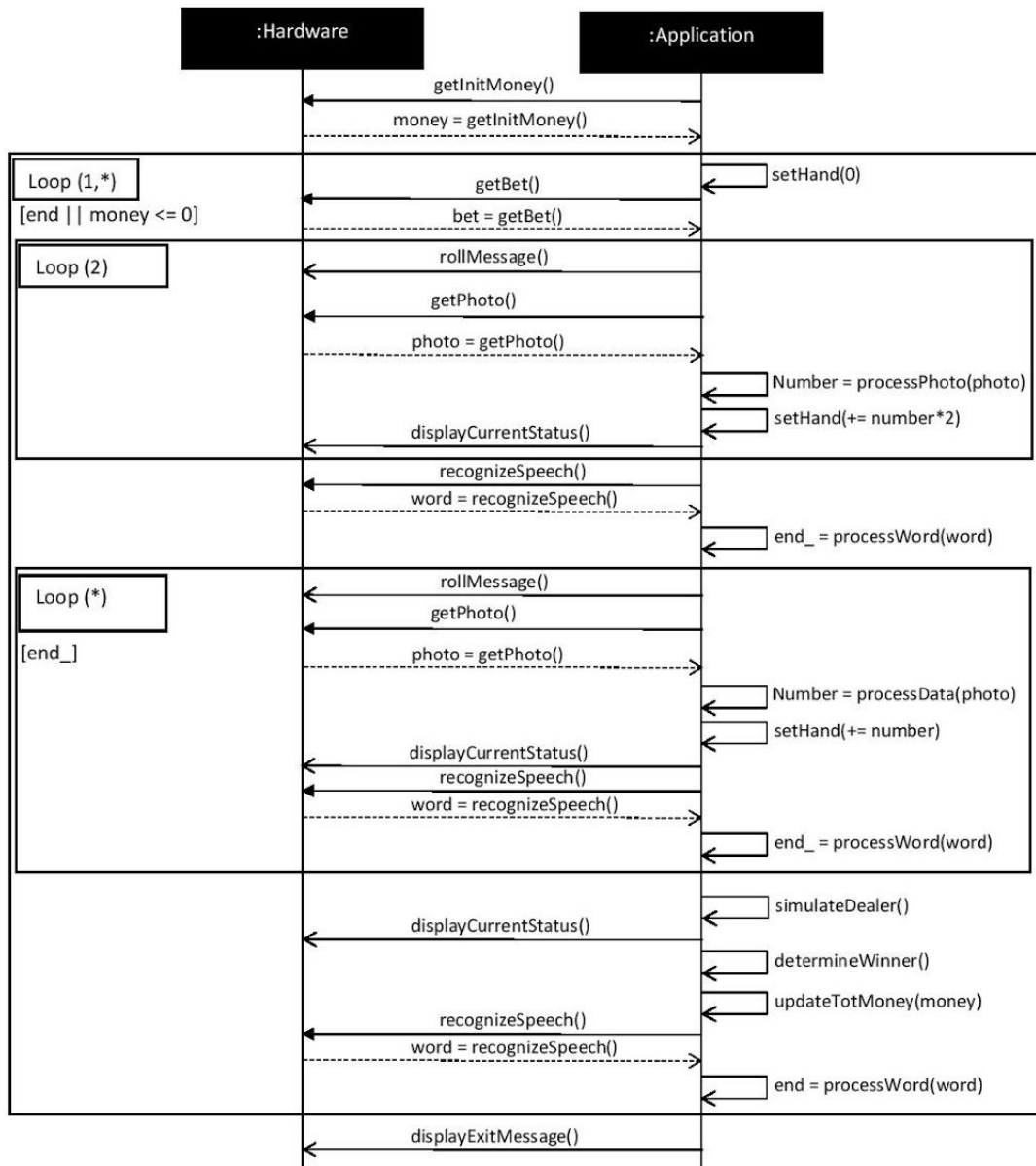
**Could** have:

- The ability to recognize more than two words.
- The ability to interact with the user with audio.
- The ability to have more than one player.

Won't have:

- The ability to recognize other objects like cards.

#### 4.5 UML sequential diagram



## 5. Components of the system

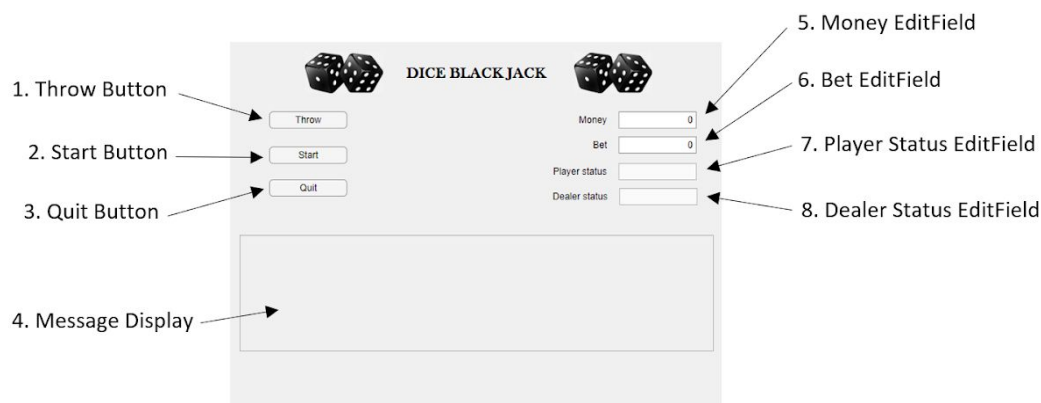
### 5.1 Hardware

- A camera to take photos.
- A computer that can run the code.
- A microphone to record speech.
- An adequate environment to capture photos and record audio.

### 5.2 Software

- MatLab to run our scripts and programs.
- Matlab image processing toolbox.
- Matlab deep learning toolbox.
- Classification algorithms for image processing and speech recognition.
- A trained neural network function to recognize the speech command.
- Matlab App Designer.

## 6. Graphical User Interface



For our system, we created the Graphical User Interface with Matlab's App Designer. Considering that the entirety of our code is with Matlab, choosing App Designer was the most convenient GUI choice. Here below you can find a description of each component, and what it does.

- 1. Throw Button:** This component of the GUI is a button. The user presses it after he has thrown the dice to activate the image processing.
- 2. Start Button:** This component of the GUI is a button. The user presses it when he wants to start the game, meaning that the main function of the code gets called inside the button callback function.
- 3. Quit Button:** This component of the GUI is a button. The user presses it when he wants to terminate the application.
- 4. Message Display:** This component is an Edit Field. Here the user can read the different commands of the system. For example: when he has to speak, who won the current hand, etc.

5. **Money EditField**: This component is an Edit Field. Here the user can insert at the beginning, the amount of money he wants to play with, and during the game he can read how much money he has left. This edit field is editable only at the beginning of the game.
6. **Bet EditField**: This component is an Edit Field. Here the user can insert his bet at the beginning of the hand. During the hand, the user can only read his bet.
7. **Player Status EditField**: This component is an Edit Field. Here the system displays the status of the current hand for the player. This Edit Field can not be modified by the user.
8. **Dealer Status EditField**: This component is an Edit Field. Here the system displays the status of the current hand for the dealer. This Edit Field can not be modified by the user.

## 7. How the system processes image

To count the dots on the dice we process the image in six different steps. (1) First we transform the image to a grayscale one with the matlab function “`rgb2gray(...)`”. (2) From here we are able to select a threshold to binarize the image with the matlab function “`imbinarize(...)`”. In our system we chose a threshold of 0.3. From here we can start to refine the image to then be able to count the objects. (3) We apply to the image an open disk-shaped morphological structuring element with radius of 1, with the matlab function “`imopen(...)`”. (4) Then we retrieve its complement (“`imcomplement(...)`”). (5) We apply once more an open disk-shaped morphological structuring element with radius of 3. (6) To conclude we are now able to count the objects in the image with the function “`bwlabel(...)`”.

```
im = rgb2gray(im);
im = imbinarize(im, 0.3);
im = imopen(im, strel('disk', 1));
im = imcomplement(im);
im = imopen(im, strel('disk', 3));
[labels, numDots] = bwlabel(im);
```



## 8. How the system processes audio

For the speech recognition feature of the system we decided to use neural networks and mel frequency cepstral coefficients (MFCC). For our version, we used only two voice commands, "NEXT" and "STOP", and trained the neural network to also distinguish noise. We first recorded 2.5 seconds of audio, and then transformed it to an array with the matlab function "getaudiodata(...)". After that we extract the first 12 mfcc with the combination of functions "ifft(log(abs(fft(...)))". From here on, the classification is all done by the neural network we trained.

```
recordblocking (rec, 2.5);
y = getaudiodata (rec);
cep_vector = ifft(log(abs(fft(y))));
cep_vector = cep_vector(1:12).';
outputNN = myNeuralNetworkFunction(cep_vector);
```

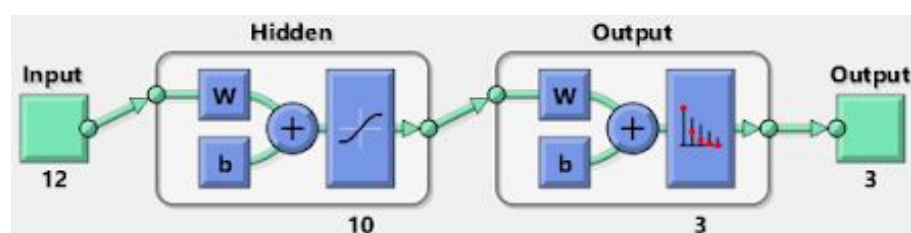
### 8.1 Neural Network

For the neural network we used the Deep learning toolbox of matlab. It is structured with 12 inputs, 10 hidden layers, and 3 outputs.

- 12 inputs: the 12 mfcc we retrieved from the earlier processing.
- 10 hidden layers: based on some research, we learned that the number of hidden layers should be less than the number of inputs and more than the number of outputs. With different tests we arrived at the conclusion that 10 hidden layers was the best choice.
- 3 outputs:  $[1,0,0] \rightarrow \text{"NEXT"}$ ,  $[0,1,0] \rightarrow \text{"STOP"}$ ,  $[0,0,1] \rightarrow \text{noise}$ .

We recorded and extracted the 12 mfcc for 80 different samples of "NEXT", "STOP", and random noises. We arranged an array with these coefficients and a target array with its corresponding outputs. With these arrays, we trained the neural network, with 70% of the samples for training, 15% for testing, and 15% for validation. To apply this neural network to the code, we generated a neural network function where we input 12 mfcc, and it determines the output.

Depending on the output, our code determines the outcome of the speech recognition. The code determines which is the highest element of the output array, and if it is bigger than 0.8 it determines the final outcome. If this is not the case or it detects noise, the code asks the user to repeat his command and records and classifies again.





## 9. Code structure

The code is divided in four different files:

- **recognize\_word.m** : in this function we process the audio for the speech recognition feature of the system.
- **countDots.m** : in this function we process the images for the counting of the dice.
- **myNeuralNetworkFunction.m** : this is the automatic function created after the training of the neural network with the Deep learning Matlab Toolbox.
- **DiceBlackJack\_GUI.m** : this is the Matlab app Designer file where all the GUI elements are coded, where the main calculations and the structure of the game are coded, and where the other functions are called. This file is the main script to run to execute the application.

You can find the description of “recognize\_word.m”, contDots.m” and “myNeuralNetworkFunction.m” in sections 7 and 8. The main structure of the game that is coded in the “DiceBlackJack\_GUI.m”, you can find it represented in the UML sequential diagram in section 4.5.

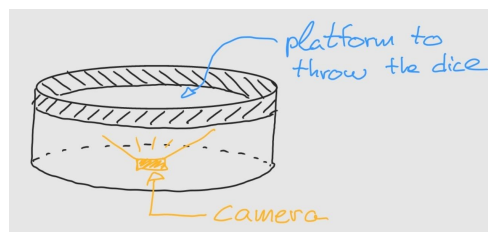


## 10. Possible future implementation and elaboration of the system

By thinking further about this project/idea, we could elaborate this system to a different level. It can be implemented as a home game for elderly people, or as a new gambling machine in a casino.

### 9.1 Home Game

To implement this idea for an elderly home we have to create a system that is straightforward and easy to use. We can not ask the user to attach the camera to the computer, launch the software, and make sure that the software reads the camera and the microphone correctly. The main idea is to have a singular system that you turn on and it is ready to go. We thought of a cylindrical object (as shown in the image below), where the camera reads the numbers not from the top but from underneath. This way, the system is much more compact and neat. Then the system will have a platform made of a material where the camera will be able to see through to process the image. This platform will also have a border, so the dice will not fall off when you throw them. The only way you will be able to interact with the system is through voice commands. This means that the system will have to have a microphone installed somewhere, and have a built-in speaker system so the users are kept up to date with the status of the game. To process and elaborate all the information, the system will have installed a microchip where all the signal processing will happen. To conclude, the user will switch on the device and have the camera, microphone, speakers, and the interaction system ready to throw the first dice.



### 9.2 Casino machine:

The idea for the casino machine is very similar to the home game idea for elderly homes. It has a built-in camera, microphone, and speaker to interact with the player. The system can be bigger, decorated to look better in a casino, and have a system to acquire money and give potential winnings to the user, depending on the casino's money flow system. An added feature that can be implemented is a physical system that "throws" the dice for the virtual dealer. A secured capsule with a mechanical component that simulates the dice throw of the dealer. This feature could be also implemented in the home game system, but it would make it more expensive and less compact.

Due to money, material, and skill limitation, we are not able to create such a system, but these are good ideas that could be developed in a potential future project! We would need a larger team that has experience in electronic and mechanical engineering to implement all the different features that these systems require.

## 11. Project planning

After we decided on the outlines of our project we made a backlog of the tasks we should follow in a Trello table. During the project term we used this backlog and updated it regularly according to our requirements. We also followed the timeplan below.

*(Our development method was agile so we modified our backlog and time table during the process.)*

Week 1
Background research (Done before Lab session)
Decision on project and it's viability (Done before Lab session)
*Decided on project, it's capabilities, functions and constraints
Research: Voice recognition with MatLab and C++ (Done before Lab session)
Research: Image recognition with MatLab and C++ (Done before Lab session)
Research: Is it viable or even beneficial to combine MatLab and C++ for our purposes
Decision: Which way is better to use : MatLab / C++ / MatLab and C++ combined
*Decided on tools to use (MatLab and it's toolboxes + GitHub, GoogleDocs, Trello Charts)
Setting up project/development environments: Github, Trello Chart (for backlog), GoogleDocs (for project documentation)
Planning on Construction of Project
Duty Separation
Week 2
*Started to implement parts of project with respect to priority
Coding image recognition function
Find or prepare Speech Commands Data Set for Neural Network
Construct and Train Neural Network
Code Speech Recognition Function: combine Neural Network function to work with the function that gets voice input and calculates MFCCs
Week 3 & 4
Coding main function that uses Image and Speech Recognition Functions
Testing and Evaluating
*Our program has worked in terminal environment
Planning GUI
Coding GUI

## 12. Testing

Here you can find two confusion matrices, one for the speech recognition and one for the image recognition. We tested each class 15 times, and we can state that the outcome is pretty accurate. Unfortunately we did not have enough time in the lab with the correct hardware to have a more detailed testing process.

### 10.1 Confusion matrix for speech recognition

	"NEXT"	"STOP"	NOISE	Total
"NEXT"	13	1	0	14
"STOP"	0	12	1	13
NOISE	2	2	14	18
Total	15	15	15	

From this confusion matrix we can see that for the command "NEXT" and "STOP", the system recognized two times for each noise. This is better than other error outcomes, because the system will ask the user to repeat the command instead of activating the wrong command.

### 10.2 Confusion matrix for image recognition

	1	2	3	4	5	6	Total
0	1	0	0	0	0	0	1
1	14	0	1	0	0	0	15
2	0	13	0	0	0	0	13
3	0	0	14	1	0	1	16
4	0	1	0	13	0	0	14
5	0	0	0	1	15	0	16
6	0	0	0	0	0	14	14
Higher	0	1	0	0	0	0	1
Total	15	15	15	15	15	15	

## 13. Evaluation

During the process of development, we encountered a couple of problems in processing images and training the neural network. Here, we will illustrate these problems, and give a quick conclusion on what we have learned from this project.

### 13.1 Problems with image processing

In the first version of “countDots.m” we had the following operations:

1. Transform image to grayscale
2. Binarize with threshold
3. Open disk-shaped morphological structuring
4. Retrieve its complement
5. Count objects in the image

This order was working when counting other objects like coins but in our situation, it did not work. The dots on the sides of the dice were also visible in some images. This was making our calculations wrong. The solution we came up with was using two “Open disk-shaped morphological structuring” with different parameters. We put the second operation after “Retrieving the complement of image”.

The new order became:

1. Transform image to grayscale
2. Binarize with threshold
3. Open disk-shaped morphological structuring (strel(‘disk’, 1))
4. Retrieve its complement
5. Open disk-shaped morphological structuring (strel(‘disk’, 3))
6. Count objects in the image

This method eliminated the side dots and gave a better outcome for recognition.

### 13.2 Problems with neural network

The problem we encountered in the training the neural network was dealing with a big database for the neural network. We initially created a neural network with 80 training samples. This worked, but the neural network recognized only two words, and it was not completely accurate. We decided to search for a bigger database that made our neural network more accurate and capable of recognizing more than two words. Unfortunately we had different problems in dealing with this big amount of data and did not manage to train the new neural network. In the end our final system has the first neural network we trained.

### 13.3 Conclusion

As a team, we learned that we do not have to underestimate the management of a big quantity of data. Downloading, merging, moving, etc. takes time and can create a lot of problems. In future projects, we will add dedicated time in our schedule, for the management of a big database.

As individuals, we expanded our knowledge in the field of managing/processing signals, and machine learning. This was one of the main objectives of this project following the success of a functioning system. The final result achieves the must requirements we decided on at the beginning of the project. There is still a big margin of improvement, but the final product

delivers a good base for future systems that one day could be sold as home games or machines for casinos.