International University of Sarajevo

Coffee machine simulator in MATLAB

Introduction to Engineering

Professor Tarik Namas

Students: Sara Avdić

Asja Bašović

Sarajevo, January 6, 2024

Table of Contents

1. INTRODUCTION

Coffee machines are a commonplace feature in our homes, offices, and schools. A coffee machine is essentially one of the earliest modern coffee brewing methods. The water boiler, brewer, coffee dispenser, control system, and ingredient motor are the fundamental components of a coffee machine (Roberts – Digital Marketing Manager, Connect Vending, 2023).

1.1 Milestones in coffee machine history

In modern times, coffee machines are very popular tools used to make coffee. There are two main types of coffee machines which are coffee vending machines and coffee machines that do not require money to be entered. As the project picked is a coffee vending machine, they will be the focus of the report. However, as the main functionality of the two types of machines is the same, the term coffee machine will just be used.

Coffee has been around for many centuries, dating back to ancient Ethiopia (Superior Sensor Technology, 2023). Ever since it was discovered, it has been liked and popular among people. As it spread across the globe, from Africa to Europe and Asia, and from Europe to the Americas, and then finally to Australia, many techniques to get the best cup of coffee were developed. With time, people strived to automate and standardize the process as much as possible, for more consistent results. Consequently, coffee machines started to be developed. The oldest known one dates to 1865 in the USA and it was made by James Nason, later improved by Hanson Goodrich in 1869 (Figure 1) (WMF Marketing Team, 2018).

Figure 1, The oldest known coffee machine

(https://blog.wmf-coffeemachines.uk.com/the-history-of-the-coffee-machine)

In 1906, the world famous La Pavoni started producing their machines commercially and the popularity of coffee machines increased. Other notable milestones include the invention of the French press by Calimani in 1929, and the invention of first pump driven coffee machine by Faema in 1961 (WMF Marketing Team, 2018). Faema's machine did not use manually operated levers but rather a motor driven pump to produce the pressure needed to make the coffee (Morris, 2020). This design is now the standard for coffee machines globally.

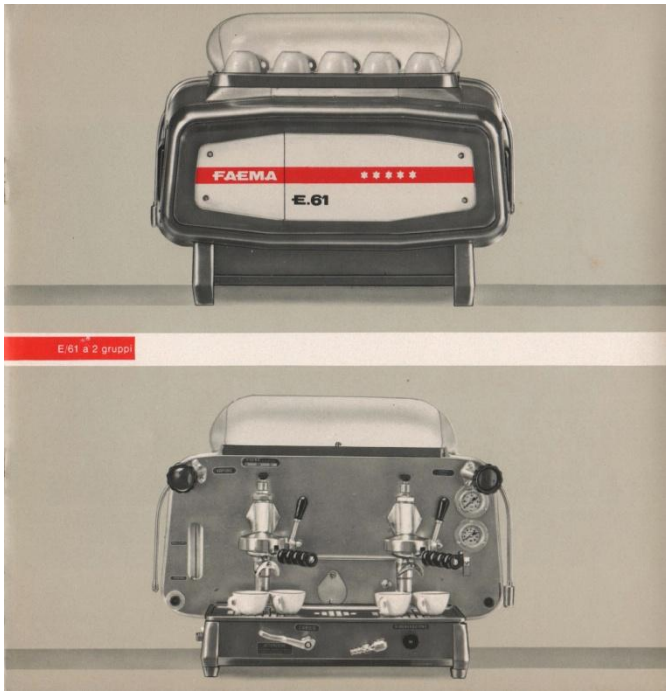

Figure 2, The first pump driven coffee machine by Faema in 1961

(https://www.historians.org/research-and-publications/perspectives-on-history/january-2021/the-faema-e61-espresso-machine)

One of the most impactful coffee machines developed was the electric drip coffee maker in 1972 in the USA (Koehler, 2016) . It introduced a new way of handling ground coffee and hot water by using a filter (Mione, 2023). It quickly became one of the most popular ways to make coffee at home due to its affordability and great results, making it still usable today.

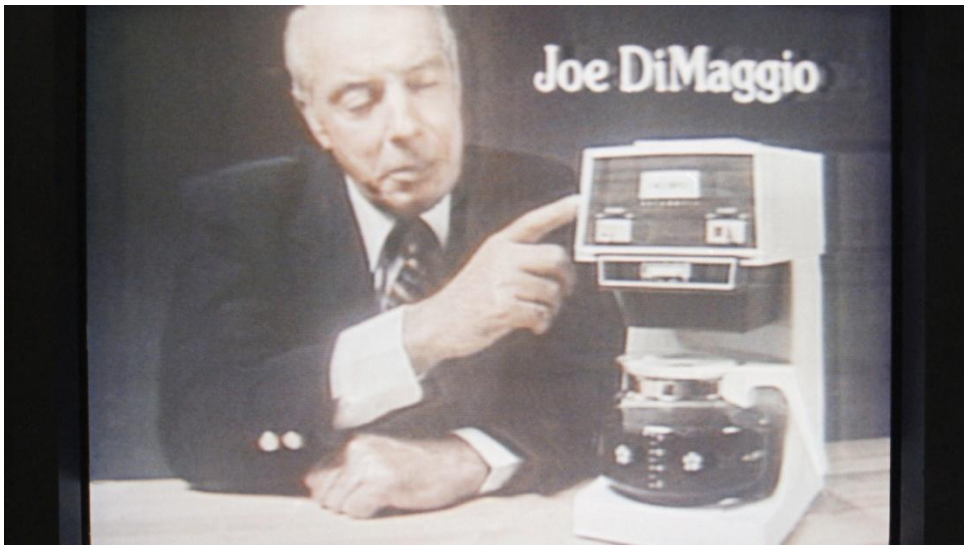

Figure 3, the inventor of the electric drip coffee maker – Joe DiMaggio

(https://www.npr.org/sections/thesalt/2016/12/16/505464932/when-mr-coffee-was-the-must-have-christmas-gift-for-java-snobs)

Currently, coffee machines are very advanced and automated. They can produce high quality coffee at the press of a button. They are quick and consistent, making them the ultimate tool for making a delicious cup of coffee.

1.2 Algorithm used in coffee machines

With its numerous settings, UI languages, and service options, a professional coffee maker requires a graphical user interface.

- Reservoir: Water is stored in a reservoir. The cycle starts when water is poured into the pot.
- Shower Head: Coffee beans are being sprayed over by the hot-water tube.
- Drip Area: Water in some coffee makers seeps into the coffee grinds (Raper, 2017).

The resistive heating element is only a coil of wire that conducts electrical current, much like the filament in a light bulb or toaster element. Plaster is applied to the coil to increase its durability.

Apart from producing warmth, the heating element also maintains the coffee warm and instantly warms the water when it is inserted (BYJU'S FutureSchool, 2022).

1.3 Sensors in coffee machines

The project did not use any sensors and only a simulation of the software part of the machine was made. However, as sensors are an important aspect in modern coffee machines, they were included in the report for a more complete explanation.

Generally, the most important aspects of a good cup of coffee are the taste and the aroma. By using sensors, the temperature and pressure can be closely monitored and controlled so that the coffee is consistent and made in optimal conditions (Superior Sensor Technology, 2023). Sensors that measure the temperature can help keep the best heat conditions while making the coffee, and pressure sensors can measure the water pressure passing through the coffee grounds, which is needed to extract the coffee flavors. Using the data from the pressure sensor, the machine can be made to control the amount of water, temperature, and flow rate. The sensor can also be very useful during the cleaning cycle as it can ensure the correct amount of air, steam, and water pressure is used (Superior Sensor Technology, 2023).

1.4 Inspiration for our project

The project was aimed to be as close to a simulation of a real-life coffee machine as possible. That is why the main source of inspiration was real-life models, such as the ones used on campus in A and B buildings or the coffee vending machine at a local food place Panera. Reference photos used for the design can be seen below in Figure 4. The main question asked while designing the machine was “What would happen if …?” Some of the questions that we addressed were “What would happen if the machine ran out of resources?”, “What would happen if a user entered a large sum of money?”, “What would happen if a user did not enter enough money?”, etc. The ways the machine can be used and the way the user errors are handled are discussed in section 5 of the report.

Figure 4, Sources of inspiration for the design

(https://www.linkedin.com/pulse/coffee-vending-machine-market-set-fly-high-years-come-lal-singh-bisht, https://necta.evocagroup.com/en/products/coffee-vending-machines, https://levending.en.made-in-china.com/product/CtYpRhEdaGDS/China-High-Quality-restaurant-Office-Maker-Table-Type-Automatic-Intelligent-Coffee-Vending-Machine.html)

2. PROPERTIES USED

The properties are the variables that are used within the functions in the continuation of the code. The properties *machineState* stores the machine's resources in the following order *[coffee, water, milk, sugar, price]* and each resource is set to 1000 (milliliters or grams, as needed), and the money is set to 100 KM. The property *coffeeResourcesNeeded* uses the same logic to store the resources needed to make each coffee, and each row represents a different coffee, as seen in the comments of the code.

The machine's status is monitored using the attributes *stock, coffeeMade,* and *paid*. The property is false if they are set to 0 (zero), for example, if *paid*=0 indicates that the consumer did not pay for their coffee.

When it comes to the logic used regarding the money, the machine can run out of money (the payment is not processed in that case) but it cannot be overfilled. The maximum amount of money that the machine can earn is earned by selling 10 flat white coffees (30 KM). Once the machine runs out of resources, the maintenance button resets the resources and the amount of money in the machine to initial values stated above.

The property *coffeeSelected* tracks which coffee was selected by the customer. For example, since espresso is saved in the first row of *coffeeResourcesNeeded*, when espresso is selected, *coffeeSelected=1*.

```
properties (Access = private)

  %[coffee water milk sugar price]; the order of the variables of one row
    machineState=[1000 1000 1000 1000 100]; %the resources the machine has initially

    coffeeResourcesNeeded =[10 30 0 0 1; %espresso

                            10 30 100 0 2; %cappuccino

                            10 30 20 0 2; %macchiato

                            20 60 0 0 2.5; %doppio

                            10 30 100 0 3; %flat white

                            10 30 30 0 3; ]; %cortado


    coffeeSelected = 0; % the index of the coffee selected
```

```
    stock=1; % if there are sufficient resources in the machine, stock=1
    coffeeMade=0; % if the coffee was made and the resources were deducted from the machine, made=1
    paid=0; % if the customer paid for the coffee selected, paid=1

end
```

3. METHODS USED

The following functions are declared in the methods section of the code, and they are set to private (they can only be used in this app/window).

3.1 *addSound* function

The following function takes the audio from the folder *sounds* as an argument when it is called and adds it to the pressed button (if the sound is './sounds/buttonSound.mp3'), to an image of a coin (if the sound is './sounds/coinSound.wav'), or to a video of a coffee being made (if the sound is './sounds/coffeeMakeAudio.mp3').

```
%the function adds a sound to a button/coin/video when the function
%is called
function addSound(app,sounds)
        [y,fs]=audioread(sounds);
        sound(y,fs);
end
```

3.2 *coffeeSelection* function

The following function takes an argument a, which is in the following format *'./coffeeImages/coffeeImage.png'* (the image shows what the coffee is made of). It shows the image on *app.UIAxes*, which is in the same position as display *app.Welcome*. This is a method of replacing images by setting the first image (*app.Welcome*) to not visible and only showing *app.UIAxes*. Finally, the text area (*app.EditField*) of a display shows the text *'Would you like to select this drink?'*.

```
%when a coffee button pressed, an image of the coffee is shown and the user is asked if they want to select it
  function coffeeSelection(app,a)

      img=imread(a);
      image(img);
      imshow(img,'parent', app.UIAxes);
      app.Welcome.Visible="off";
      app.EditField.Value='Would you like to select this drink?';

  end
```

3.3 *outOfStock* and *makeCoffee* functions

The function *makeCoffee* takes the argument *coffee* which is the *coffeeSelected*. It then calls the function *outOfStock* and passes the *coffeeSelected*. The function *outOfStoc*k checks if there are enough resources to make the selected coffee by checking if each resource in *coffeeMachine* is greater than the *coffeeResourcesNeeded* for that coffee. In other words, it checks the row in the matrix *coffeeResourcesNeeded* that corresponds to the coffee selected, and checks if the values of that row are greater than the resources that the machine has (*machineState*). If any machine resource is smaller than the resource needed to make the coffee, *stock*=0, and the *app.Lamp='r'* (the lamp light changes from green to red) indicates that the machine must be restocked.

The function *makeCoffee* checks if *stock*=0 and if it is, it displays a message *'OUT OF STOCK'*, returns the money entered, shows an appropriate animation, and adds a sound of coins to the animation. Otherwise, it deducts the resources needed to make the coffee from the machine. To track this change (that the coffee was made) *app.made=1*.

```
%checks if any resource is out of stock, turns the lamp red if it
%is, and makes the button for maintenance appear
function outOfStock(app,coffee)
      for i=1:4
         if app.machineState(1,i) < app.coffeeResourcesNeeded(coffee,i)
            app.stock=0;
            app.Lamp.Color='r';
             app.maintenanceButton.Visible='on';

          end
       end
end

%deducts the resources needed to make the coffee from the machine
function makeCoffee(app, coffee)

          outOfStock(app, coffee);

          if(app.stock==0)

                    app.EditField.Value='OUT OF STOCK';

                    app.change.Value=app.money.Value;

                    app.money.Value=0;

                    moneyOutputAnimation(app);

                    addCoinSound(app);

          else

                    for i=1:4
```

```
                app.machineState(1,i)=app.machineState(1,i) - app.coffeeResourcesNeeded(coffee,i);
            end
            app.made=1;
        end
end
```

3.4 *coffeePaid* function

The following function first checks if any money was entered into the machine. If not, it displays a message *'Please enter the money!'*. Otherwise, it checks if there is enough machine money to give change. If that is true, it checks if enough money was entered to buy the selected coffee. If less money was entered, the message *'Not enough money!'* is displayed. If enough money was entered *paid*=0, it takes money into the machine and returns change (only if more money than necessary was entered, appropriate sound and animation are shown). If there is not enough machine money to give the change, it returns the money entered and displays a message *'Cannot process the payment'* (appropriate sound and animation are shown).

```
%checks if enough money was entered and calculates the change
   function coffeePaid(app, coffee)
        if  app.money.Value==0 %checks if any money was entered
           app.EditField.Value='Please enter the money!';
        elseif app.money.Value<=app.machineState(5) %then checks if the machine can give change
                                                        (if someone enters a huge number the)
           if app.money.Value<app.coffeeResourcesNeeded(coffee,5)
              app.paid=0;
              app.EditField.Value='Not enough money!';
           else
              app.machineState(5)= app.machineState(5)+app.coffeeResourcesNeeded(app.coffeeSelected,5);
              app.paid=1;
             if app.money.Value>app.coffeeResourcesNeeded(coffee,5)
                        moneyOutputAnimation(app);
                        addSound(app,'./sounds/coinSound.wav');
             end
             app.change.Value=app.money.Value-app.coffeeResourcesNeeded(coffee,5);
           end
        elseif  app.money.Value>app.machineState(5)%special case: large sum of money was entered and the
machine cannot give the needed change, so it just returns the money
            app.change.Value=app.money.Value;
            app.money.Value=0;
            app.EditField.Value='Cannot process the payment';
           moneyOutputAnimation(app);
```

```
            addSound(app,'./sounds/coinSound.wav');
        end
    end
```

3.5 *restartMachine* function

After a coffee is made or when the function is called, the function *restartMachine* restarts the state of the machine. It returns everything to its original state, except for the machine resources.

```
%restarts the state (except the resources) when a coffee is made
    function restartMachine(app)
        app.coffeeSelected = 0;
        app.made=0;
        app.paid=0;
        app.money.Value=0;
        app.change.Value=0;
        app.Welcome.Visible="on";
        img=imread('./coffeeImages/background.png');
        image(img);
        imshow(img,'parent', app.UIAxes);
        app.EditField.Value='Welcome!';
        app.maintenanceButton.Visible='off';
        app.sugarCount.Value=0;

        app.MacchiatoButton.BackgroundColor=[0.42, 0.40, 0.40];
        app.MacchiatoButton.FontColor='w';

        app.CappuccinoButton.BackgroundColor=[0.42, 0.40, 0.40];
        app.CappuccinoButton.FontColor='w';

        app.EspressoButton.BackgroundColor=[0.42, 0.40, 0.40];
        app.EspressoButton.FontColor='w';

        app.DoppioButton.BackgroundColor=[0.42, 0.40, 0.40];
        app.DoppioButton.FontColor='w';

        app.FlatWhiteButton.BackgroundColor=[0.42, 0.40, 0.40];
        app.FlatWhiteButton.FontColor='w';

        app.CortadoButton.BackgroundColor=[0.42, 0.40, 0.40];
        app.CortadoButton.FontColor='w';
    end
```

3.6 *buttonPressed* function

The following function takes buttons as arguments and changes their appearances. When a button is pressed, it changes the color of its background to pink and the color of the font to black, and the rest of the buttons are changed to default colors.

```
%the button changes the color if it is selected and other buttons are
%changed to default
   function buttonPressed(app, button, button1, button2, button3, button4, button5, event)
        button.FontColor='k';
        button.BackgroundColor=[1.00, 0.00, 0.62];

        button1.FontColor='w';
        button1.BackgroundColor=[0.42, 0.40, 0.40];

        button2.FontColor='w';
        button2.BackgroundColor=[0.42, 0.40, 0.40];

        button3.FontColor='w';
        button3.BackgroundColor=[0.42, 0.40, 0.40];

        button4.FontColor='w';
        button4.BackgroundColor=[0.42, 0.40, 0.40];

        button5.FontColor='w';
        button5.BackgroundColor=[0.42, 0.40, 0.40];

   end
```

3.7 *moreSugar*/*lessSugar* functions

The amount of sugar is 0 by default and it can be added up to five times. When sugar is added, the function *moreSugar* displays the number of times sugar will be added, and that amount is removed from the machine resources. When sugar is removed, the function *lessSugar* does the same thing, except it returns the sugar that was previously selected to the machine.

```
%increments the amount of sugar by 1 if sugar<5 and the machine has
      %enough resources
      function moreSugar(app)
         if app.sugarCount.Value<5 && app.machineState(1,4)>=5
            app.sugarCount.Value=app.sugarCount.Value+1;
            app.machineState(1,4)=app.machineState(1,4)-5;
         end
      end

     %decrements the amount of sugar by 1 if sugar>0 and returns the
     %resources to the machine
      function lessSugar(app)
         if(app.sugarCount.Value>0)
            app.sugarCount.Value=app.sugarCount.Value-1;
            app.machineState(1,4)=app.machineState(1,4)+5;
         end
      end
```

3.8 *moneyInputAnimation*/*moneyOutputAnimation* functions

The following functions display animations for money input and money output. The animations are saved in the folder *moneyImages* and are GIFs.

```
%displaying an animation of a coin entering the machine
function moneyInputAnimation(app)
    app.inputSlot.Icon='./moneyImages/moneyAnimation.gif';
    pause(1);
    app.inputSlot.Icon='./moneyImages/moneyInput.gif';
end

%displaying an animation of change being given
function moneyOutputAnimation(app)
    app.outputSlot.Icon='./moneyImages/changeAnimation.gif';
    pause(1);
    app.outputSlot.Icon='./moneyImages/changeOutput.gif';
end
```

3.9 *coffeeMakeVideo* function

The function *coffeeMakeVideo* displays a video of a coffee being made. It first loads it and then reads it frame by frame two times from the beginning. The video is shown on *app.UIAxes2* which is in the same location as *app.coffeeSlot* (displays an image of an empty glass) and *app.coffeeSlot2* (displays an image of a full glass). a*pp.coffeeSlot* is visible by default, so its visibility is turned off to display the video. After the video is finished, *app.coffeeSlot2* is set to visible to show the coffee done. The video is cleared, so it can be used again when another coffee is supposed to be made.

```
%show a video of a coffee being made
function coffeeMakeVideo(app)
        v = VideoReader("coffeeVideo/coffeeVideo.mp4");
        app.coffeeSlot2.Visible='off';
        app.coffeeSlot.Visible='off';
        addSound(app,'./sounds/coffeeMakeAudio.mp3');
        for i=1:2
                v.CurrentTime = 0.0;
                while hasFrame(v)
                        vidFrame = readFrame(v);
                        image(vidFrame,"Parent",app.UIAxes2)
```

```
                pause(1/v.FrameRate)
            end
        end
        clear v;
        app.coffeeSlot2.Visible='on';
end
```

4. BUTTON PRESSED FUNCTIONS

The following functions are executed when the corresponding button is pressed.

4.1 Coffee buttons

The following functions are made for each button when each coffee is selected. They all follow the same logic. Firstly, a function for sound *addSound* is called. Then, the function for showing an image is called *coffeeSelection*, and then *coffeeSelected* is set to the corresponding coffee (ex. if espresso is pressed *coffeeSelected*=1). The last function that is called is *buttonPressed.*

```
% Button pushed function: EspressoButton
    function EspressoButtonPushed(app, event)
        addSound(app, './sounds/buttonSound.mp3');
        coffeeSelection(app,'./coffeeImages/espresso.png');
        app.coffeeSelected=1;
        buttonPressed(app, app.EspressoButton, app.CappuccinoButton, app.MacchiatoButton, app.DoppioButton,
app.FlatWhiteButton, app.CortadoButton);
    end

    % Button pushed function: CappuccinoButton
    function Cappuchino(app, event)
        addSound(app, './sounds/buttonSound.mp3');
        coffeeSelection(app,'./coffeeImages/cappuccino.png');
        app.coffeeSelected=2;
        buttonPressed(app, app.CappuccinoButton, app.EspressoButton, app.MacchiatoButton, app.DoppioButton,
app.FlatWhiteButton, app.CortadoButton);
    end

    % Button pushed function: MacchiatoButton
    function macchiato(app, event)
        addSound(app, './sounds/buttonSound.mp3');
        coffeeSelection(app,'./coffeeImages/macchiato.png');
        app.coffeeSelected=3;
        buttonPressed(app, app.MacchiatoButton, app.CappuccinoButton, app.EspressoButton, app.DoppioButton,
app.FlatWhiteButton, app.CortadoButton);
    end

    % Button pushed function: DoppioButton
    function doppio(app, event)
        addSound(app, './sounds/buttonSound.mp3');
        coffeeSelection(app,'./coffeeImages/doppio.png');
        app.coffeeSelected=4;
        buttonPressed(app, app.DoppioButton, app.CappuccinoButton, app.MacchiatoButton, app.EspressoButton,
app.FlatWhiteButton, app.CortadoButton);
    end

    % Button pushed function: FlatWhiteButton
    function flatwhite(app, event)
        addSound(app, './sounds/buttonSound.mp3');
        coffeeSelection(app,'./coffeeImages/flatWhite.png');
```

```
        app.coffeeSelected=5;
        buttonPressed(app, app.FlatWhiteButton, app.CappuccinoButton, app.MacchiatoButton, app.DoppioButton,
app.EspressoButton, app.CortadoButton);
    end

    % Button pushed function: CortadoButton
    function cortado(app, event)
        addSound(app, './sounds/buttonSound.mp3');
        coffeeSelection(app,'./coffeeImages/cortado.png');
        app.coffeeSelected=6;
        buttonPressed(app, app.CortadoButton, app.CappuccinoButton, app.MacchiatoButton, app.DoppioButton,
app.FlatWhiteButton, app.EspressoButton);
    end
```

4.2 Confirm button

Firstly, the sound function *addSound* is called. Afterward, it checks if any coffee was selected. If If no coffee was selected, it displays *'Select a coffee!'*. Otherwise, it then checks if the coffee was paid. If that statement is true, it displays a message *'Your coffee will be ready soon!'* and the process continues.

```
% Button pushed function: ConfirmButton
    function confirm(app, event)
        addSound(app, './sounds/buttonSound.mp3');
        if app.coffeeSelected==0
            app.EditField.Value='Select a coffee!';
        else
            coffeePaid(app, app.coffeeSelected);
        end
        if app.money.Value>0 && app.paid==1 && app.coffeeSelected~=0
            makeCoffee(app, app.coffeeSelected);
            if (app.made==1)
                app.EditField.Value='Your coffee will be ready soon!';
            end
        end
    end
```

4.3 addSugar/noSugar buttons

The following functions call the functions *addSound* and *moreSugar/lessSugar*.

```
    % Button pushed function: ButtonSugar
    function addSugar(app, event)
        addSound(app, './sounds/buttonSound.mp3');
        moreSugar(app);
    end
    % Button pushed function: ButtonNoSugar
    function noSugar(app, event)
```

```
    addSound(app, './sounds/buttonSound.mp3');
    lessSugar(app);
end
```

4.4 kf10/kf20/kf50/km1/km2/km5 buttons

The following functions are for adding money to the machine (paying). *appMoney.Value* shows the money entered the machine. When an image of the coin is clicked *appMoney.Value* is updated to show the total money entered. It is updated according to which coin is entered (for example, if the coin of 5 KM is clicked, 5 is added to *appMoney.Value).* The functions *addSound* and *moneyInputAnimation* are called.

```
% Image clicked function: KF10
function kf10(app, event)
    addCoinSound(app, './sounds/ coinSound.wav');
    app.money.Value=app.money.Value+0.10; %adding money to display
    moneyInputAnimation(app);

end

% Image clicked function: KF20
function kf20(app, event)
    addCoinSound(app, './sounds/ coinSound.wav');
    app.money.Value=app.money.Value+0.20; %adding money to display
    moneyInputAnimation(app);
end

% Image clicked function: KF50
function kf50(app, event)
    addCoinSound(app, './sounds/ coinSound.wav');
    app.money.Value=app.money.Value+0.50; %adding money to display
    moneyInputAnimation(app);
end

% Image clicked function: KM1
function km1(app, event)
    addCoinSound(app, './sounds/ coinSound.wav');
    app.money.Value=app.money.Value+1; %adding money to display
    moneyInputAnimation(app);
end

% Image clicked function: KM2
function km2(app, event)
    addCoinSound(app, './sounds/ coinSound.wav');
    app.money.Value=app.money.Value+2; %adding money to display
    moneyInputAnimation(app);
end

% Image clicked function: KM5
function km5(app, event)
```

```
        addCoinSound(app, './sounds/ coinSound.wav');
        app.money.Value=app.money.Value+5; %adding money to display
        moneyInputAnimation(app);
    end
```

4.5 Maintenance button

The maintenance button is only shown when the machine is out of stock. The functions *addSound* and *restartMachine* are called. Then, *app.stock* is set to one 1 and all resources in *app.machineState* are restocked as they were initially. Finally, the lamp color becomes green.

```
% Button pushed function: maintenanceButton
    function maintenance(app, event)
        addCoinSound(app);
        restartMachine(app);
        app.stock=1;
        for i=1:4
            app.machineState(i)=1000;
        end
        app.machineState(5)=100;
        app.Lamp.Color='g';
end
```

5. HOW TO USE THE COFFEE MACHINE

5.1 The best way to use the coffee machine

1. Input enough money in the machine by clicking an image of a coin.
2. Select wanted coffee by pressing a button on the left side.
3. Add sugar by pressing "+" button or remove sugar by pressing "–" button on the right side.
4. Press the confirm button below the display screen.
5. Wait for your coffee to be made and take your change (if there is any).
6. Enjoy your coffee! You can select another one if you would like to.

5.2 Handling user errors

5.2.1 No coffee selected

If confirm button is pressed before selecting a coffee, "Select a coffee" message will be displayed. Figure 5 shows what the coffee machine looks like in this state.

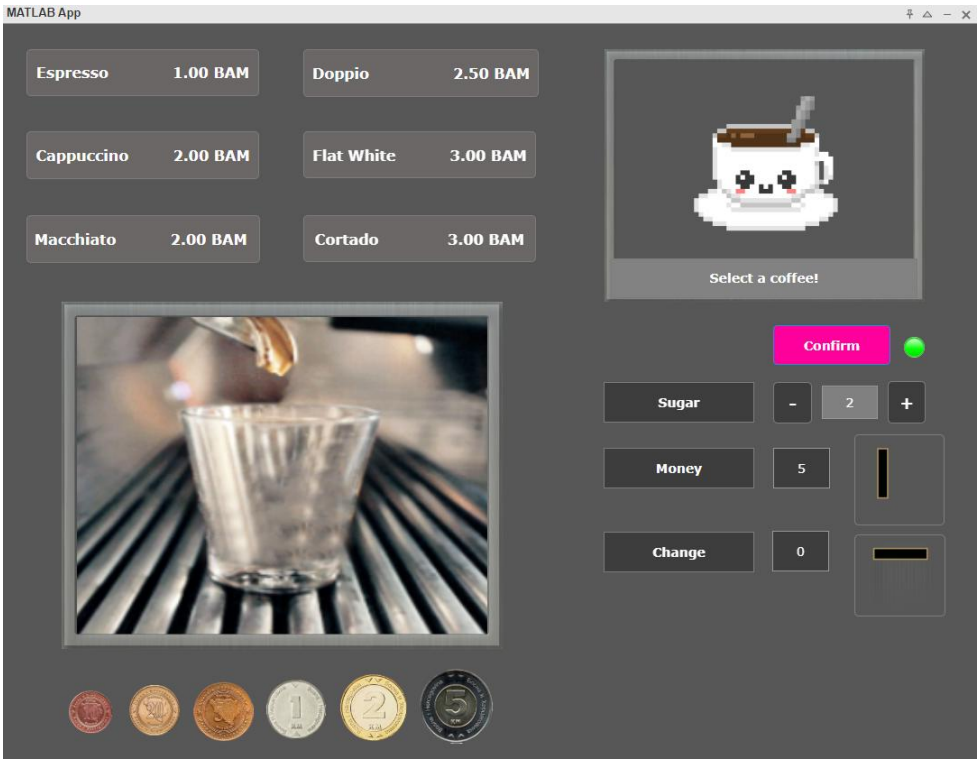

Figure 5, No coffee selected

5.2.2 No money entered

If confirm button is pressed after selecting a coffee, but no money was entered, “Please enter the money!” message will be displayed. Figure 6 shows what the coffee machine looks like in this state.

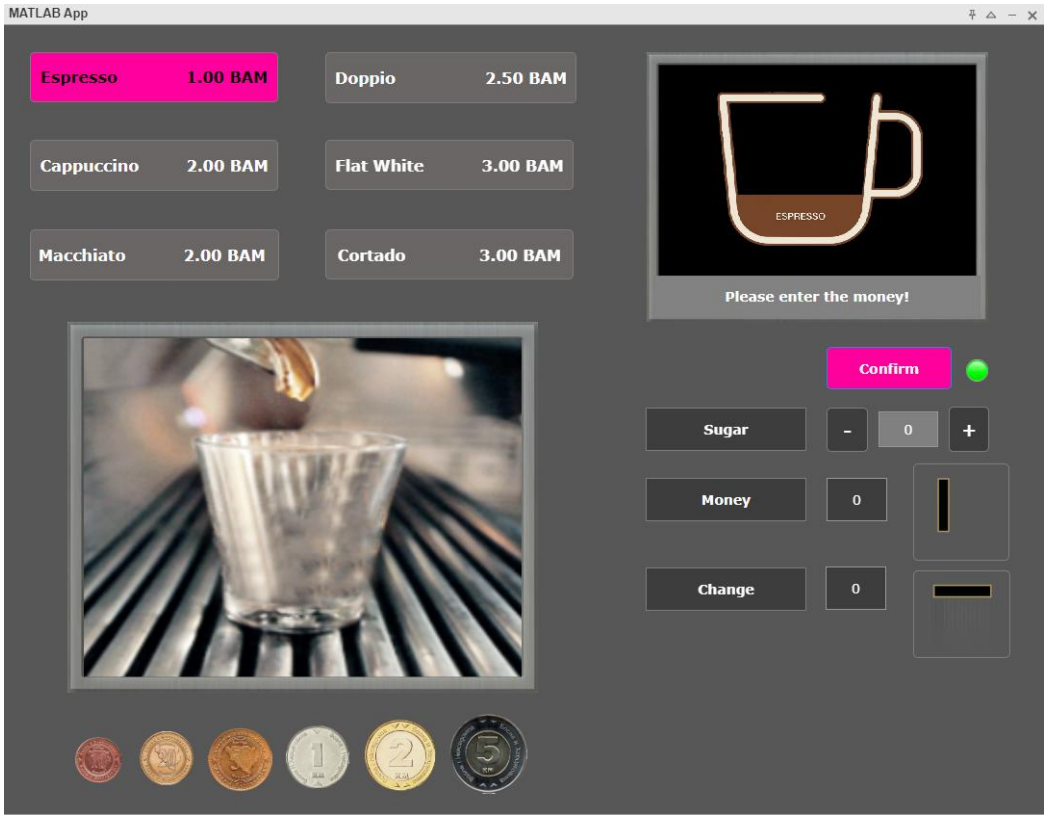

Figure 6, No money was entered

5.2.3 Not enough money entered

If confirm button is pressed after selecting a coffee, but not enough money was entered, “Not enough money!” message will be displayed. Figure 7 shows what the coffee machine looks like in this state.

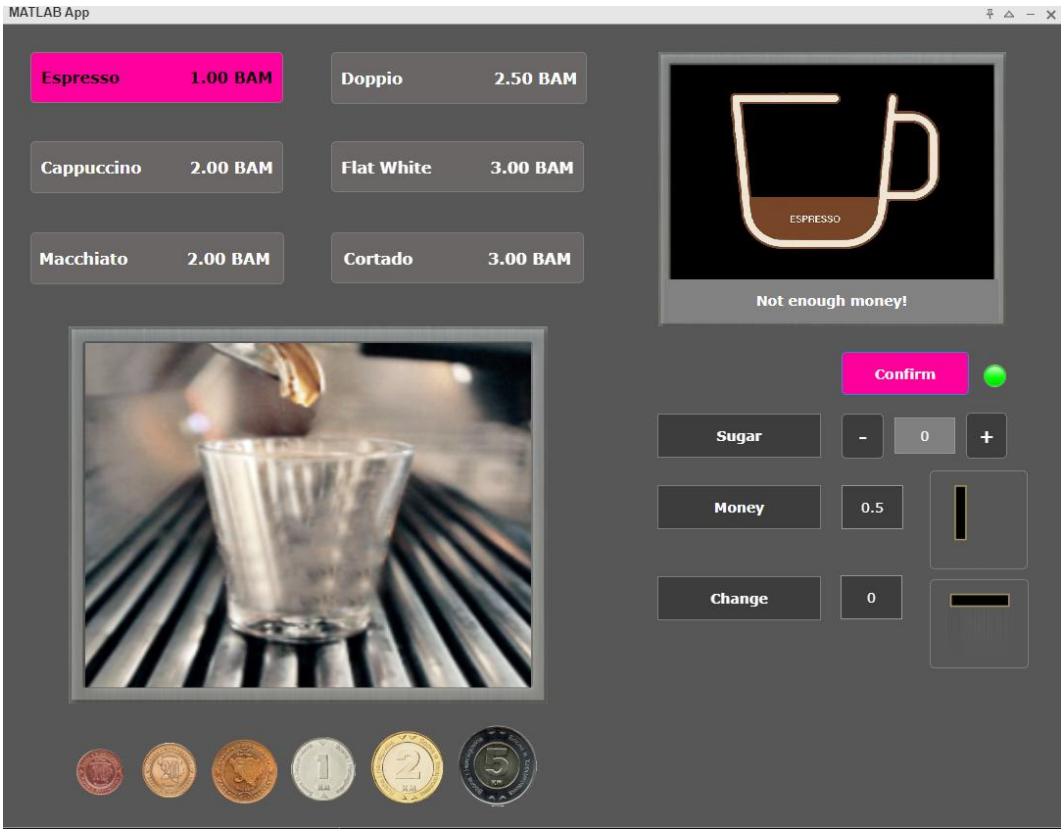

Figure 7, Not enough money entered

5.2.4 Machine has ran out of stock

If machine was used correctly, but machine has ran out of the resources, “OUT OF STOCK”, message will be displayed, and “MAINTENANCE” button will be shown. After pressing “MAINTENANCE” button, machine will be restarted. Figure 8 shows what the coffee machine looks like in this state.

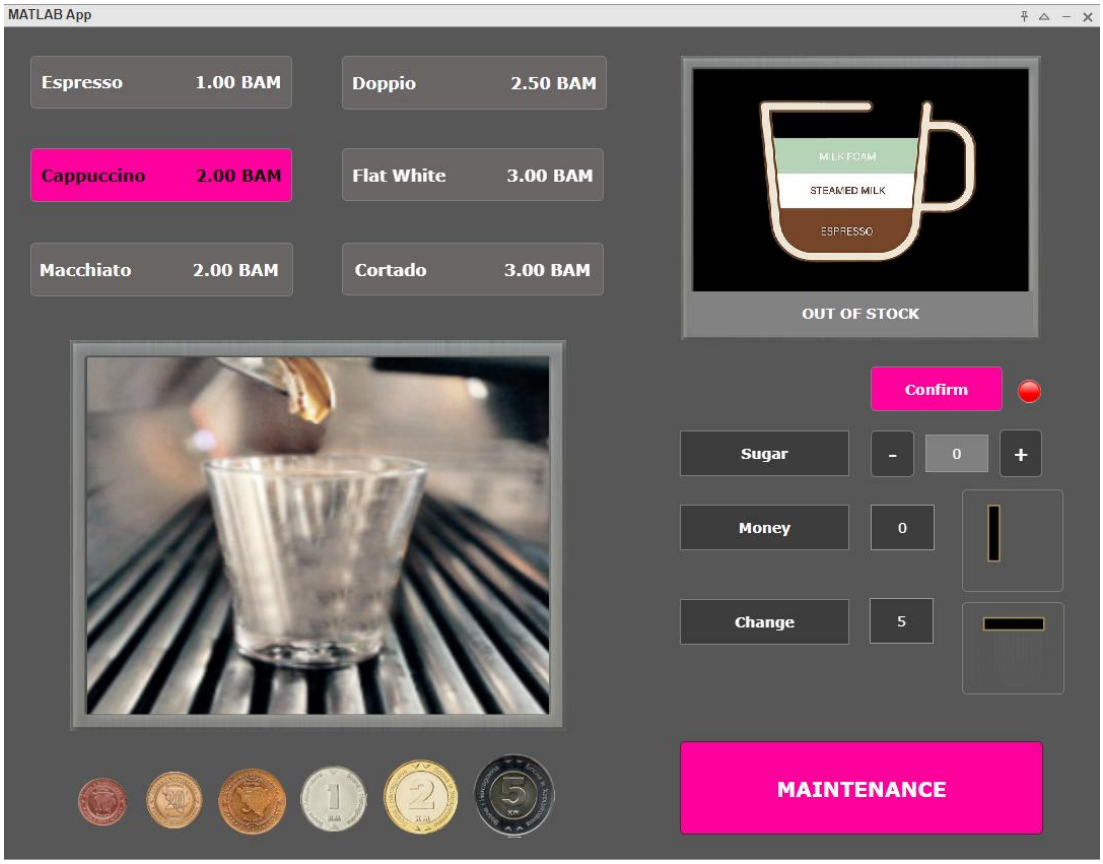

Figure 8, Machine has ran out of stock

5.2.5 Cannot process the payment

If machine does not have enough money to return change, “Cannot process the payment.” message will be displayed, and entered money will be returned. Figure 9 shows what the coffee machine looks like in this state.

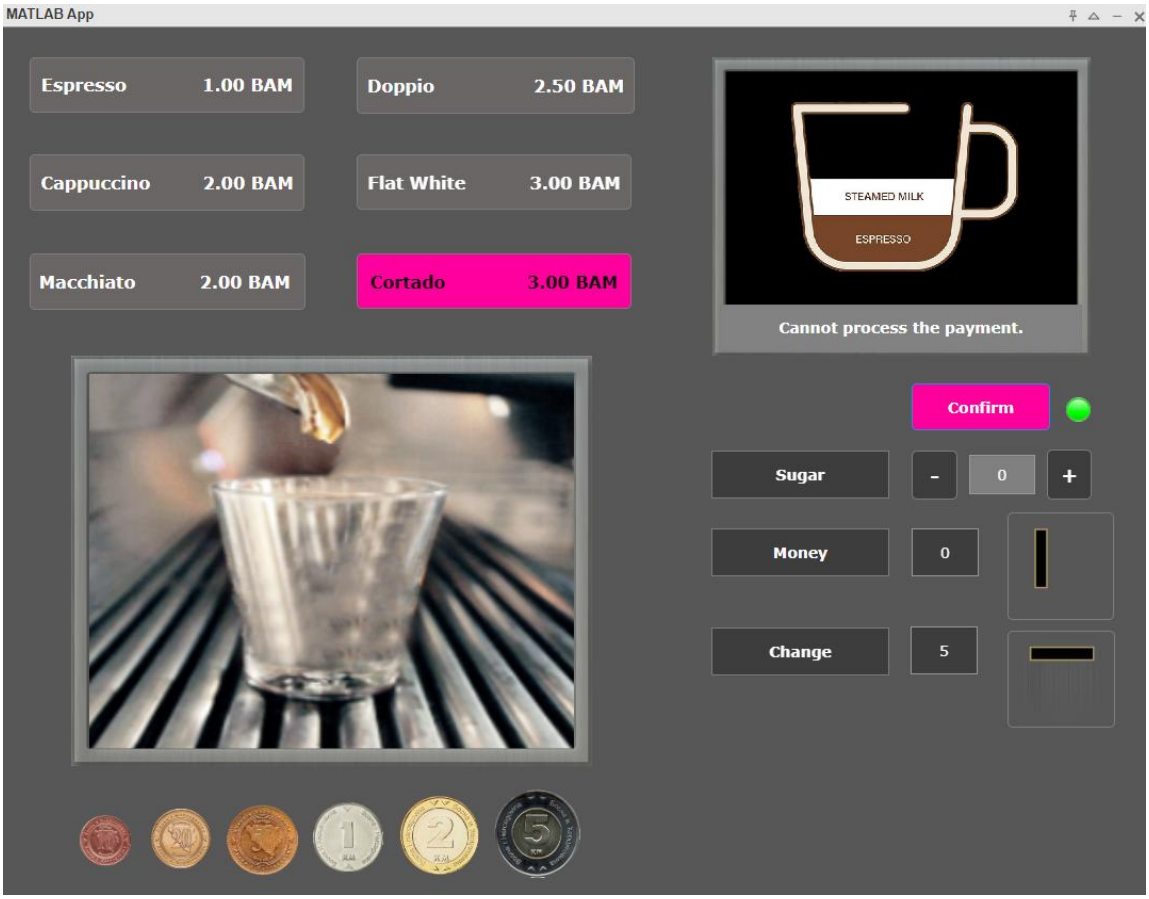

Figure 9, Cannot process the payment

6. REFERENCES

6.1 Images

- https://www.cbbh.ba/content/read/1072?lang=hr
- https://www.mymokafe.com/post/different-types-of-coffee-explained
- https://www.reddit.com/r/gifs/comments/pkt0b/coffee_machine_pouring_coffee/

6.2 Sounds

- https://pixabay.com/sound-effects/search/buttons/
- https://mixkit.co/free-sound-effects/coin/
- https://pixabay.com/sound-effects/coffee-machine-30756/

6.3 MATLAB links

- https://www.mathworks.com/matlabcentral/answers/368430-push-button-audio-for-gui
- https://www.mathworks.com/matlabcentral/answers/39121-change-button-color-back-to-original-value
- https://www.mathworks.com/matlabcentral/answers/303648-how-do-i-see-how-many-times-a-button-has-been-pressed-in-app-designer
- https://www.mathworks.com/matlabcentral/answers/769512-open-an-image-when-push-button-is-clicked-in-app-designer
- https://www.mathworks.com/matlabcentral/answers/485773-app-designer-_-pushbutton-callback
- https://www.mathworks.com/matlabcentral/answers/436723-how-can-i-get-user-input-and-then-put-it-in-a-variable-in-appdesigner
- https://www.mathworks.com/matlabcentral/answers/57300-how-to-read-display-multiple-images-from-a-folder

6.4 Websites

- Superior Sensor Technology. (2023). *Making the Perfect Coffee with Pressure Sensors.* AZoSensors. Retrieved on December 25, 2023 from https://www.azosensors.com/article.aspx?ArticleID=2746.

- WMF Marketing Team. (2018). *The history of the coffee machine*. Uk.com. https://blog.wmf-coffeemachines.uk.com/the-history-of-the-coffee-machine
- Koehler, J. (2016). *When Mr. Coffee Was The Must-Have Christmas Gift For Java Snobs*. NPR. https://www.npr.org/sections/thesalt/2016/12/16/505464932/when-mr-coffee-was-the-must-have-christmas-gift-for-java-snobs
- Morris, J. (2020). *The Faema E61 Espresso Machine | Perspectives on History | AHA*. Historians.org. https://www.historians.org/research-and-publications/perspectives-on-history/january-2021/the-faema-e61-espresso-machine
- Mione, A. (2023). *Exploring the History of the Home Coffee Machine*. Coffee Complex. https://coffeecomplex.com.au/home-coffee-machine-history/
- BYJU'S FutureSchool. (2022). How Does Code Help You Make Coffee? BYJU'S Future School Blog; BYJU'S Future School Blog. https://www.byjusfutureschool.com/blog/how-does-code-help-you-make-coffee/#The_Working_Algorithm_of_a_Coffee_Machine
- Raper, A. (2017). *How Do Espresso Machines Work?* Clive Coffee; Clive Coffee. https://clivecoffee.com/blogs/learn/how-do-espresso-machines-work