**Task Assigned Date: May 06,2024,**

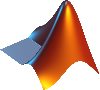
**Task Finished Date: May 07,2024**

**Step by Step Document on Creating Crypto Card GUI in MATLAB.**

In the document. There is documented process on how to create GUI using MATLAB. The aim of the GUI is to transmit data inputted from the GUI Field. The Data inputted will be in Numerical Form but the transmission should be in Binary Form. The Data should be transmitted and received using UART mode. Below is the Step by Step Guide to Achieve This Goal.

# Step 1: Opening MATLAB

1. Open **MATLAB** by Clicking on the Logo or **MATLAB.exe**.

Or by running command on run if you have installed it on default location.

**C:\Program Files\Polyspace\R<version>\bin\matlab.exe**

# Step 2: Navigating to App Designer

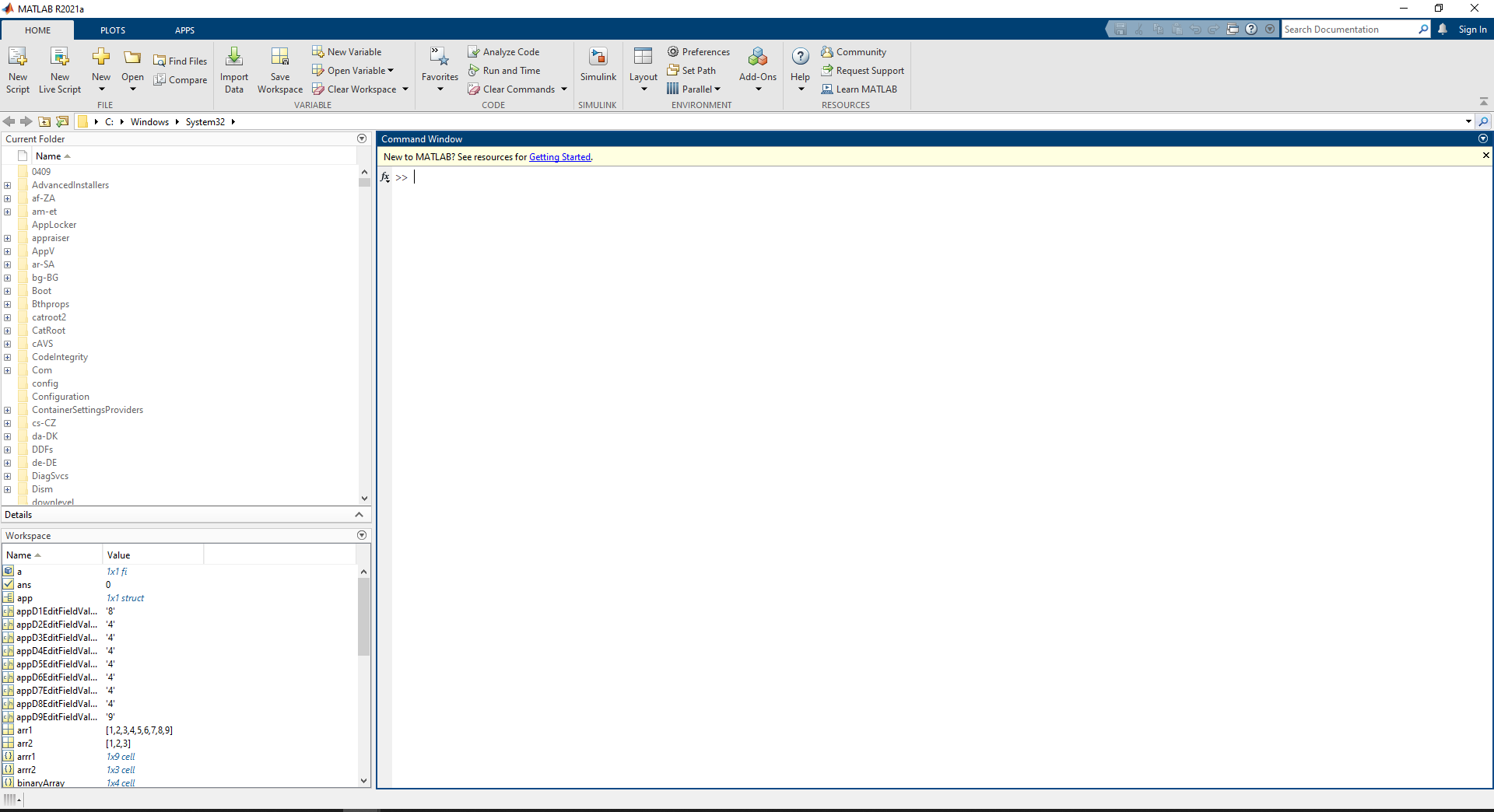
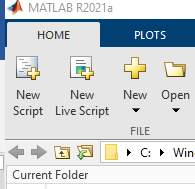
1. When MATLAB is open. The Default Screen should look like Figure 1. 

Figure : MATLAB Default Screen

1. Go to the Top Left Corner of the MATLAB Screen and click on the drop-down button underneath the New Button as shown in Figure 2.



Drop Down Menu

Figure : New Button in MATLAB

1. Now, the drop-down selection will look like the following. Choose/ Click on the App from the List which is shown as highlighted in the Figure 3.

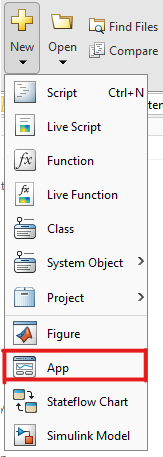


Figure : Drop Menu on the New Button

# Step 3: Starting and Navigating around App Designer

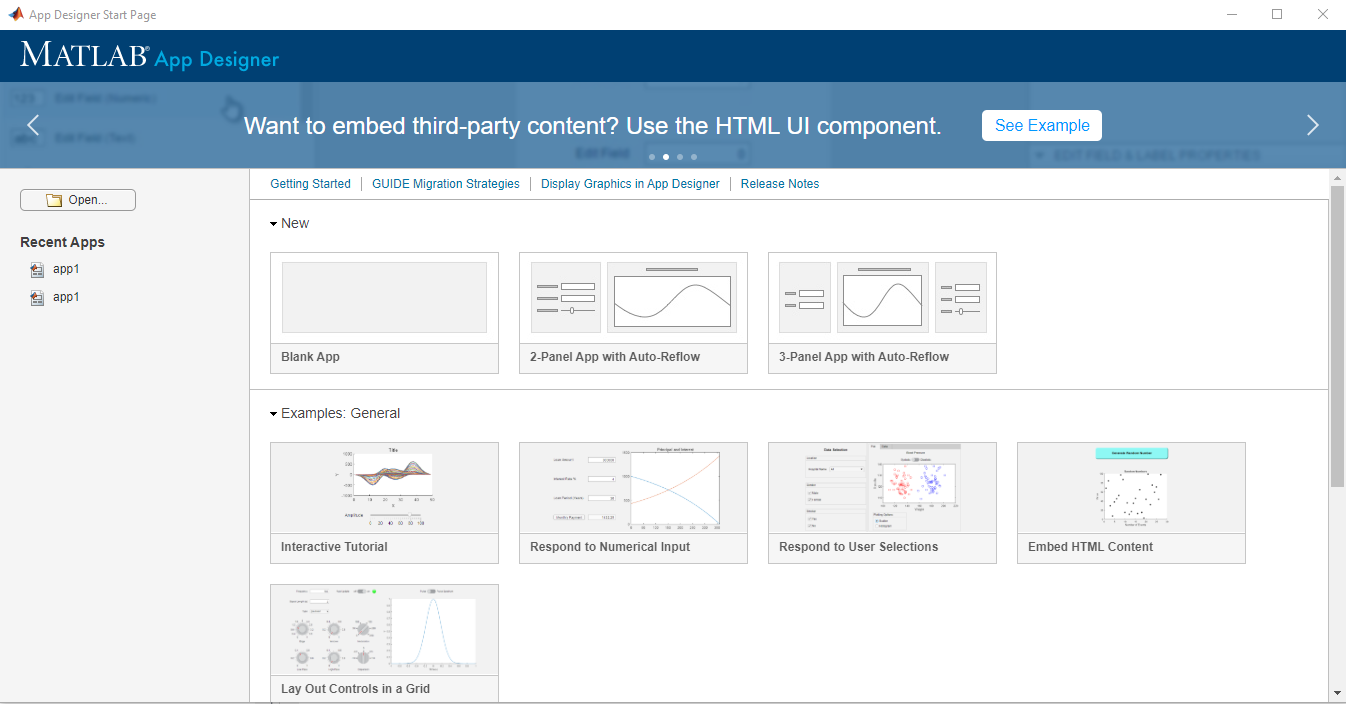
1. App Designer Start Page will popup and the following screen will be displayed as shown in Figure 4.

Figure : App Designer Start Page Default Screen

1. Select the **Blank App** from the following Selection as highlighted in Figure 5

Figure : Blank App in the Selection Screen

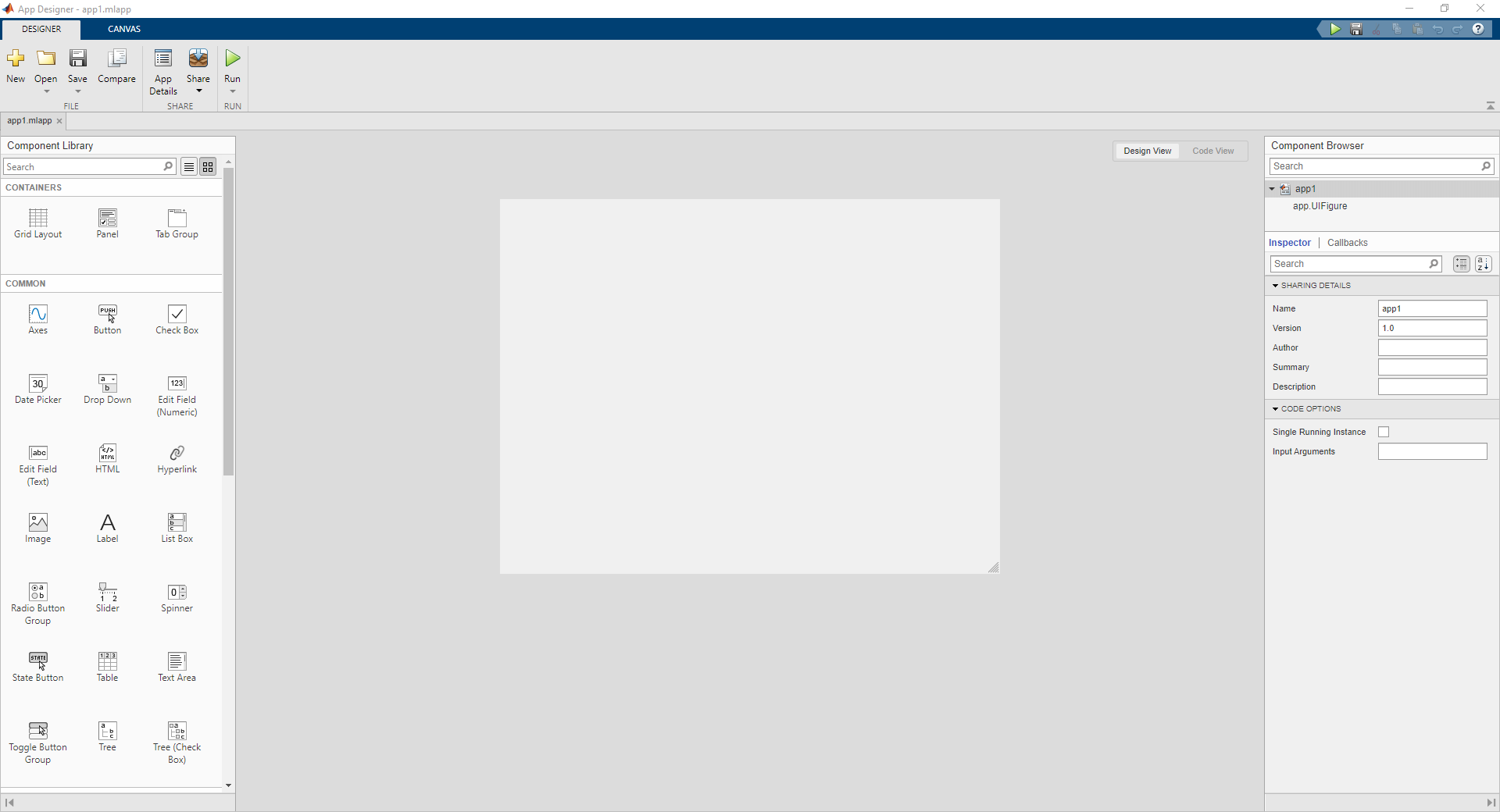
1. The Default Window Should Look Like the picture Shown in Figure 6. 

Figure : Default Screen of the App Designer

# Step 4: Exploring and Using App Designer a little more.

1. Now, there should be a **Component Library** on the Right Side of the **App Designer**. It should be something Like the Picture Shown in Figure 7. There are different **types of components** used for various types of things.

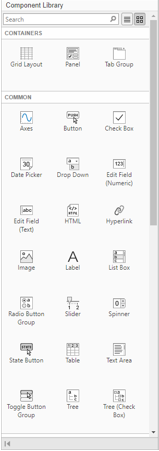


Figure : Component Library in App Designer

1. For our purposes, we want to build an App to **Send Data** with Spreading and Staggering Index. So, we will only be using “**Panel**” for looks and “**Edit Field (Text)**” for Data Entry and “**Spinner**” and lastly “**Image**” for inserting Logo of the “**AKSA-SDS**” for **Spreading and Staggering Index**. I have highlighted them as following

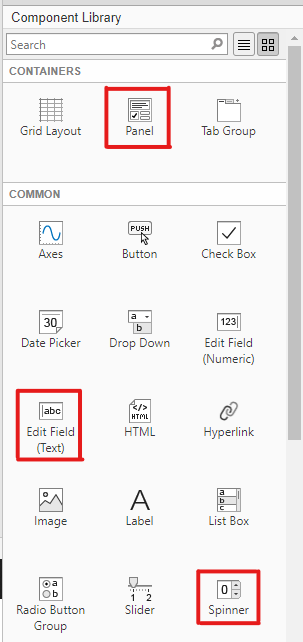


Figure : Component Selection Required for the GUI

1. You can **drag and drop** components from the library to the Main Window as shown in the figure below.

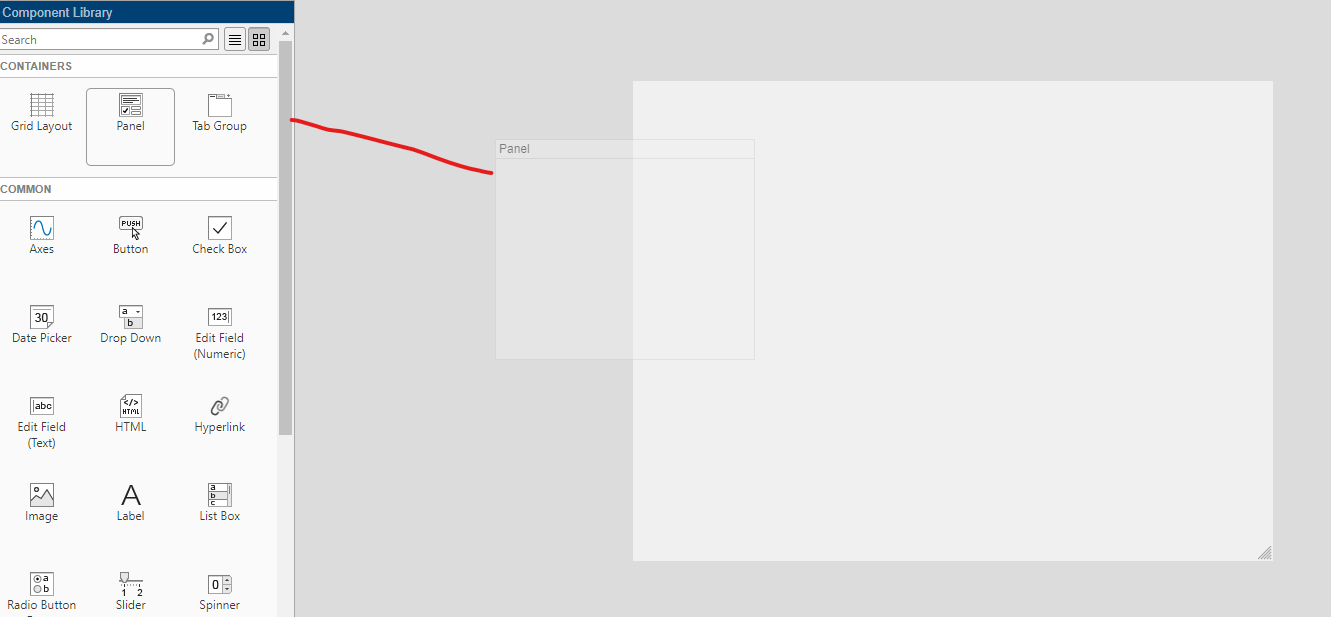


Figure : Drag and Drop Component from the Component Library

1. When an **Object** is dropped on the Window. On the Right-Side Component Browser will appear.

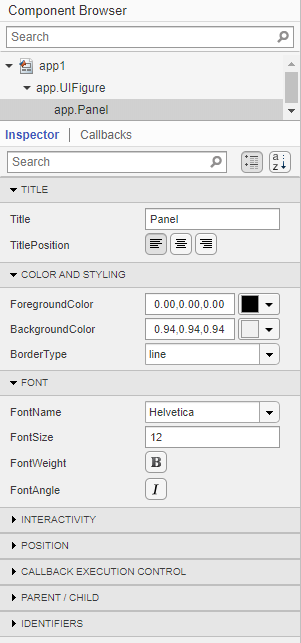


Figure : Component Browser on the App Designer

1. The name of the variable associated will be displayed on the top of component browser window as shown in Figure 11. We made the following window and its respective objects are as follows.

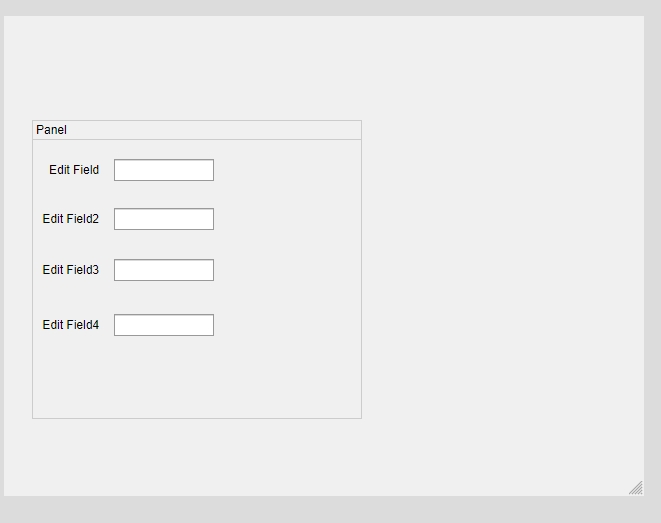
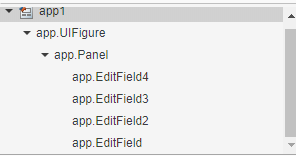


Figure : Object Created from the Edit Field

Figure : Edit Field in the App

1. We can use these object names to access the properties and modify things as we need.
2. Double Clicking on the Edit Field Name Enables it to change the name.

# Step 5: Creating and Styling the GUI Design

1. Make Design as you see fit. For our following project. We Designed the GUI as shown in Figure 13.

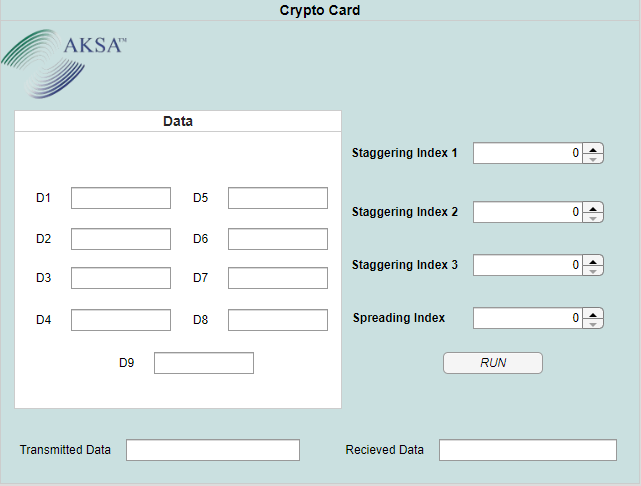


Figure : Completed GUI of the Project

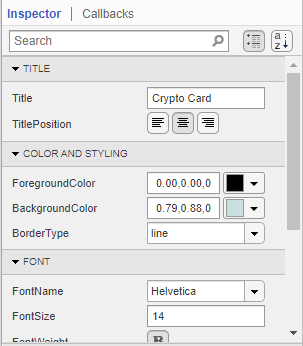
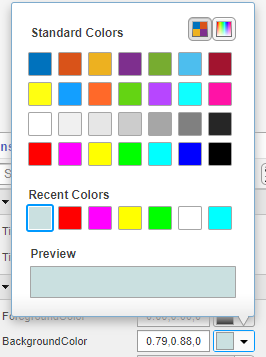
The Background color can be changed by component Browser window as shown in figure 14,15.  

Figure : Styling and Coloring Figure : Color Option

1. In Figure 15, the highlight box shows three different ways to style the Panel. “**ForegroundColor**” Changes the Color of the Top Line Text. While, “**BackgroundColor**” changes the color of the Background of the Panel. The Figure Shows the color scheme available to change the color.
2. The Logo Can be insert using the image box in the **Component Library**.
3. Once Design is Completed. Next Step is to Code the Functionality when the “**RUN**” button is pressed.

# Step 6: Adding Functionality to the GUI

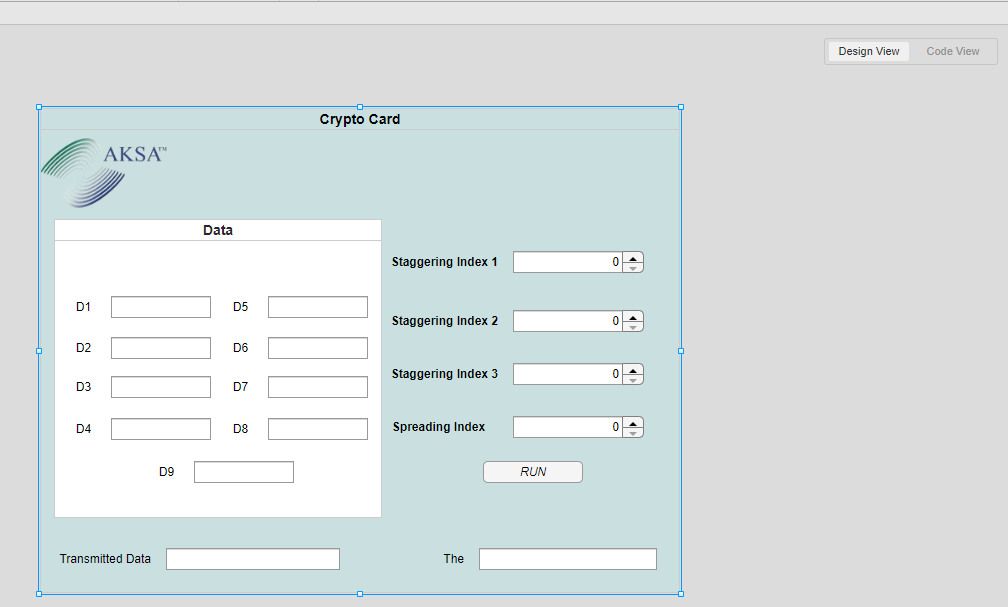
1. One the Designer Click on the Code View as highlighted in the Figure 16.

Figure : Design and Code View Option

1. Before That, to add function to a button. We need to add **Function Callbacks**. So, Select Run Button. Right Click on the **RUN** button. Go to Callbacks -> Go to **RUNButtonPushed** callback.

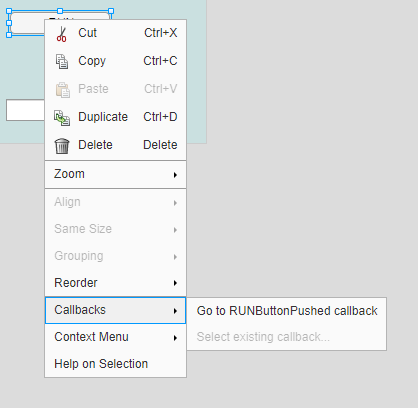


Figure : Callback Addition

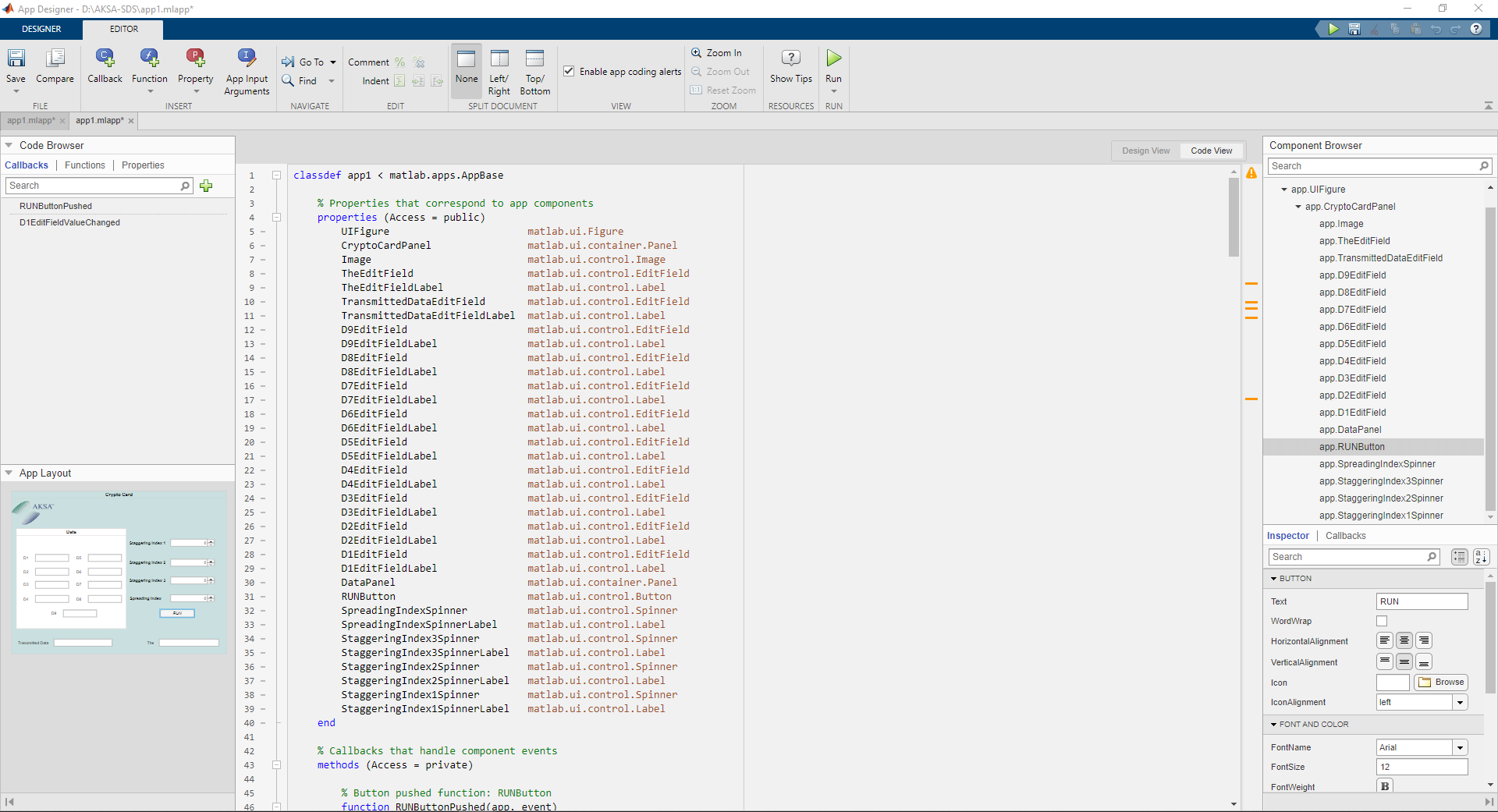
1. With This a New Function to define RUN Button Functionality will be added. Now, go in the Code View Button as shown in the Figure 18. The Following Window with slight variation will appear as your requirement differ. 

Figure : Default Coding Screen if the GUI

1. The **Gray Area** of the code indicates unchangeable code. Which is **locked/restricted** by MATLAB. The White Area of the code is changeable and can be modified to suit our needs.
2. Let’s go to the **Callback function** of the RUN Button.



Figure : Callback Function of the RUN Button from GUI

1. There, our **RUN Button** Function is found.

# Step 7: Understanding and Developing Algorithm for the Requirement

Now, lets understand our requirement carefully before writing code. Our Requirements are as follows:

1. We have inputs from **9 different data blocks**. Named **D1 to D9**. Each block should be only be able to input numerical numbers from **0-15**.
2. Data blocks should be contained in **8-bits** where **MSB Left Side 4-bits should be ‘0000’**. Whereas, the last 4-bit should contain the data inputted from the GUI.
3. If Data outside 0-15 is inserted. An **Error Dialog Box** should appear. The data should be converted into its **binary counterpart**.
4. Next, we have three other block named Staggering Index **1 to 3**. Each block should only be able to input numerical numbers from 0-15.
5. This time **no restrictions** are implemented and each should display a 4-bit binary number
6. Lastly, we have one block named Spreading Index with numerical numbers from 0-255. It also has no restriction although the restriction is imposed to insert data **from 0-255 meaning 8-bit data**.
7. We have total of 13 Blocks input. **With each of the blocks converted to binary**. Data Blocks converted into 8-bit binary numbers.
8. All of the Data should be concatenated together and form a 100-bit string which needs to be send through UART.
9. Lastly, **UART code** need to be inserted. It should connect to Port “**COM7**” with a baud rate of **9600**.
10. Finally, it should receive the data send at serial port “COM7” at baud rate of 9600.
11. The **Transmitted Data** should be displayed on Transmitted Data Field while **received** data should be displayed on the Received Field
12. The Device which will transmit and received data is as follows: This is shown in the Device Manager as shown in Figure 20.

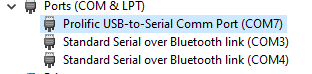


Figure : Prolific USB COM7 Port in Device Manager

1. That is all of our requirement from the button function.

# Step 8: Writing and Understanding Code according to the requirement

1. First, let’s get some things started with initiating important numbers.

% Set loop parameters

loop = 9; % Number of D fields

staggeringindexloop = 3; % Number of staggering index fields

% Initialize arrays

value = [];

staggeringval = zeros(1, staggeringindexloop);

binaryval = {};

staggeringbinaryval = cell(1, staggeringindexloop);

1. The Highlighted Code shows starts the initialization of the variables required to store and process data. We stored number of Data Input Rows and Staggering Index rows. Then, we need arrays to store data inputted in the field and also the binary values we need to store by converting the inputted data from the field.
2. **Value** and **binaryval** store data from D1 to D9 with numerical and Binary Values Respectively.
3. Now, we need to store value from field into the array. To do this, we need For Loop. The Code is as follows:

% Get values from edit fields and spinners

for i = 1:loop

varName = ['app.D', num2str(i), 'EditField.Value'];

value{i} = eval(varName);

end

for i = 1:staggeringindexloop

varName = ['app.StaggeringIndex', num2str(i), 'Spinner.Value'];

staggeringval(i) = eval(varName);

end

Spreadingval = app.SpreadingIndexSpinner.Value;

It takes data from all of the fields and store it in different arrays.

1. Now, we need to convert them into binary values and while also limiting the input able to be passed from the field into the processing part.

% Convert values to binary strings

for i = 1:loop

if isnan(str2num(value{i})) || ~isreal(str2num(value{i}))

errordlg(['Please Enter Value between 0 - 15 on D', num2str(i)], 'Incorrect Value')

return;

elseif str2num(value{i}) < 0 || str2num(value{i}) > 15

errordlg(['Please Enter Value between 0 - 15 on D', num2str(i)], 'Incorrect Value')

return;

else

stored = dec2bin(value{i}, 8);

sorted = stored(5:end);

binaryval{i} = ['0000' sorted];

end

end

**Note: The code inside the else statement is used to add 4-bit 0’s on the LEFT SIDE MSB**

for i = 1:staggeringindexloop

if isnan(staggeringval(i)) || ~isreal(staggeringval(i))

errordlg(['Please Enter Value between 0 - 15 on Staggering Index', num2str(i)], 'Incorrect Value')

return;

elseif staggeringval(i) < 0 || staggeringval(i) > 15

errordlg(['Please Enter Value between 0 - 15 on Staggering Index', num2str(i)], 'Incorrect Value')

return;

else

staggeringbinaryval{i} = dec2bin(staggeringval(i), 4);

end

end

if isnan(Spreadingval) || ~isreal(Spreadingval)

errordlg('Please Enter Value on Spreading Index between 0 - 255', 'Incorrect Value')

return;

elseif Spreadingval < 0 || Spreadingval > 255

errordlg('Please Enter Value on Spreading Index between 0 - 255', 'Incorrect Value')

return;

else

spreadingbinaryval = dec2bin(Spreadingval, 8);

end

The Highlighted codes If, elseif and else statements are used to limit values from going over the limit while dec2bin function convert decimal values into Binary Values.

Dec2bin function takes two arguments.

**Dec2bin(<value>, <min number of bits>)**

1. In the next part, we join/concatenate all of the binary output and combine them into one.

% Concatenate binary strings

binaryout = [binaryval{:}, staggeringbinaryval{:}, spreadingbinaryval];

% Set the output

app.TransmittedDataEditField.Value = binaryout;

This part indicates the Concatenation of the Binary Values and outputting them into the Transmitted Data Field.

1. Now, last part of the piece of the puzzle is to send it to UART and receive the value. The Code for that is as follows.

%%Transmitting the Data to UART

message = serialport("COM7", 9600);

writeline(message, binaryout);

%%Recieving Data from UART

recieveddata = readline(message);

%Sending Value to Edit Field

app.TheEditField.Value = recieveddata;

The Code Sends and receive the 100-bit binary value through COM7 port with baud rate of 9600.

1. To Run the GUI, Need to Press Green Start button as shown in Figure 21.

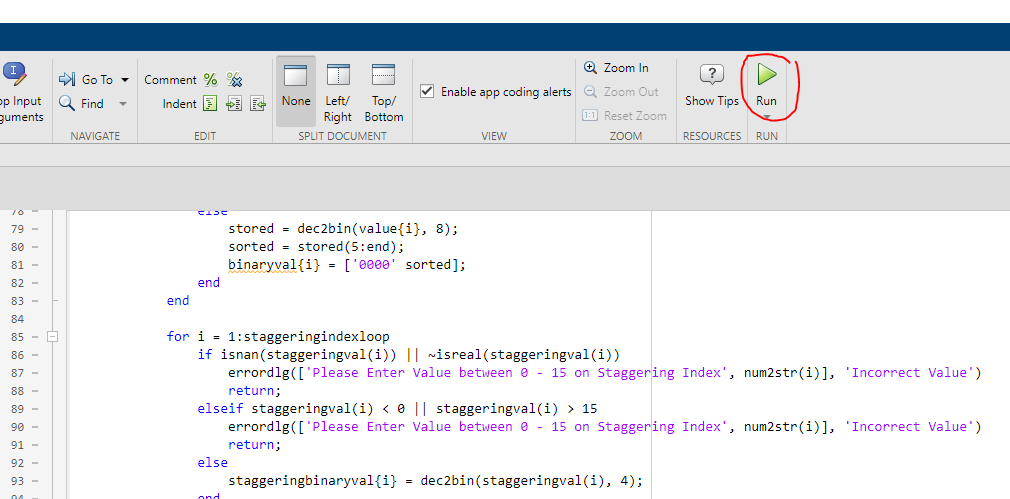


Figure : Run Button To Start GUI

Here is the working condition of the GUI

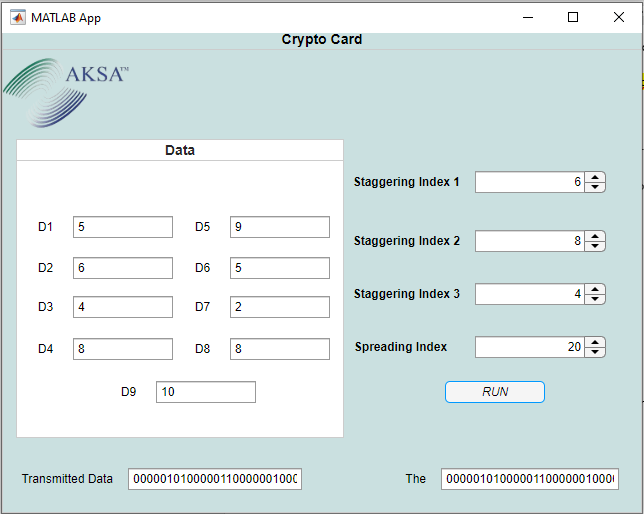


Figure : Working of the GUI with Coding Addition

It is receiving the same data it is transmitted which verifies that the system is working as intended.

Code:

% Button pushed function: RUNButton

function RUNButtonPushed(app, event)

% Set loop parameters

loop = 9; % Number of D fields

staggeringindexloop = 3; % Number of staggering index fields

% Initialize arrays

value = [];

staggeringval = zeros(1, staggeringindexloop);

binaryval = {};

staggeringbinaryval = cell(1, staggeringindexloop);

% Get values from edit fields and spinners

for i = 1:loop

varName = ['app.D', num2str(i), 'EditField.Value'];

value{i} = eval(varName);

end

for i = 1:staggeringindexloop

varName = ['app.StaggeringIndex', num2str(i), 'Spinner.Value'];

staggeringval(i) = eval(varName);

end

Spreadingval = app.SpreadingIndexSpinner.Value;

% Convert values to binary strings

for i = 1:loop

if isnan(str2num(value{i})) || ~isreal(str2num(value{i}))

errordlg(['Please Enter Value between 0 - 15 on D', num2str(i)], 'Incorrect Value')

return;

elseif str2num(value{i}) < 0 || str2num(value{i}) > 15

errordlg(['Please Enter Value between 0 - 15 on D', num2str(i)], 'Incorrect Value')

return;

else

stored = dec2bin(value{i}, 8);

sorted = stored(5:end);

binaryval{i} = ['0000' sorted];

end

end

for i = 1:staggeringindexloop

if isnan(staggeringval(i)) || ~isreal(staggeringval(i))

errordlg(['Please Enter Value between 0 - 15 on Staggering Index', num2str(i)], 'Incorrect Value')

return;

elseif staggeringval(i) < 0 || staggeringval(i) > 15

errordlg(['Please Enter Value between 0 - 15 on Staggering Index', num2str(i)], 'Incorrect Value')

return;

else

staggeringbinaryval{i} = dec2bin(staggeringval(i), 4);

end

end

if isnan(Spreadingval) || ~isreal(Spreadingval)

errordlg('Please Enter Value on Spreading Index between 0 - 255', 'Incorrect Value')

return;

elseif Spreadingval < 0 || Spreadingval > 255

errordlg('Please Enter Value on Spreading Index between 0 - 255', 'Incorrect Value')

return;

else

spreadingbinaryval = dec2bin(Spreadingval, 8);

end

% Concatenate binary strings

binaryout = [binaryval{:}, staggeringbinaryval{:}, spreadingbinaryval];

% Set the output

app.TransmittedDataEditField.Value = binaryout;

%%Transmitting the Data to UART

message = serialport("COM7", 9600);

writeline(message, binaryout);

%%Recieving Data from UART

recieveddata = readline(message);

%Sending Value to Edit Field

app.TheEditField.Value = recieveddata;

end