Topic	Guess the Key	
Class Description	**Students will learn how to create a Guessing game using the 4X4 Keypad.**	
Class	**PRO C260**	
Class time	**50 mins**	
Goal	● Connecting a 4X4 keypad with ESP32. ● Writing program to create the game.	
Resources Required	● Teacher Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ○ Smartphone ● Student Resources: ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen	
Class structure	**Warm-Up** **Teacher-Led Activity** **Student-Led Activity** **Wrap-Up**	**10 mins** **15 mins** **15 mins** **10 mins**
Credit & Permissions:	**Code samples used for Firebase-Google Authentication are licensed under the [Apache 2.0 License].** **Expo documentation used from - [https://expo.io]** **Note: Keep this row section only if applicable**	

WARM-UP SESSION - 10 mins	
Teacher Action	**Student Action**

Hey <student's name>. How are you? It's great to see you! Are you excited to learn something new today? **Following are the WARM-UP session deliverables:** • Greet the student. • Revision of previous class activities. • Quizzes.	**ESR**: Hi, thanks! Yes, I am excited about it! Click on the slide show tab and present the slides
WARM-UP QUIZ Click on In-Class Quiz	
Activity Details **Following are the session deliverables:** • Appreciate the student. • Narrate the story by using hand gestures and voice modulation methods to bring in more interest in students.	
TEACHER-LED ACTIVITY 15mins	
Teacher Initiates Screen Share	
• **Keypad connections.** • **Algorithm to generate key codes randomly.**	
Teacher Action	**Student Action**
So, in the last class, we learned how to use a keypad with ESP32 and created a secret password application. Do you have any questions from the previous class? *If the student has any doubts, clarify the doubts* Have you ever played, guess the number game?	 **ESR**: Varied **ESR** : Varied

Let's play the game so that we can understand everything about the game. Are you excited? Great, consider the matrix below,	**ESR** : Yes

	Column 1	Column 2	Column 3	Column 4
Row 1	1	2	3	4
Row 2	5	6	7	8
Row 3	A	B	C	D
Row 4	E	F	G	H

Can you guess the element at the intersection of the **2nd Row** and **3rd column**?	**ESR** : 7
Correct!. Not very interesting, right?	**ESR** : Yes
Ok, let's do some improvements. Let's add a time factor.	
Can you guess the element at the intersection of the **3rd Row** and **2nd column** in **1 second**? Your time starts now! *Note : Have a stopwatch in your hand.*	**ESR** : B
It's a bit challenging, right?	**ESR** : Yes
Let's add a bit more fun. Let's switch the order of hints?	

<table>
<tr>
<td>Can you guess the element at the intersection of the,

a) 3rd Row and 2nd Column.
b) 1st Column and 4th Row.
c) 4th Row and 3rd Column.
d) 4th Column and 2nd Row.

You have 1 second for each question. Go!

Let’s try to create this game on our 4x4 keypad.</td>
<td>ESR : Varied</td>
</tr>
<tr>
<td>Go to the wokwi simulator and create a new ESP32 project,

Let’s try to create the circuit diagram first.

Step -1:Select the components

● 1 x ESP32
● 1 x Keypad: 4 Rows (R1-R4) and 4 Columns (C1-C4)

Step -2: Let’s do connections:

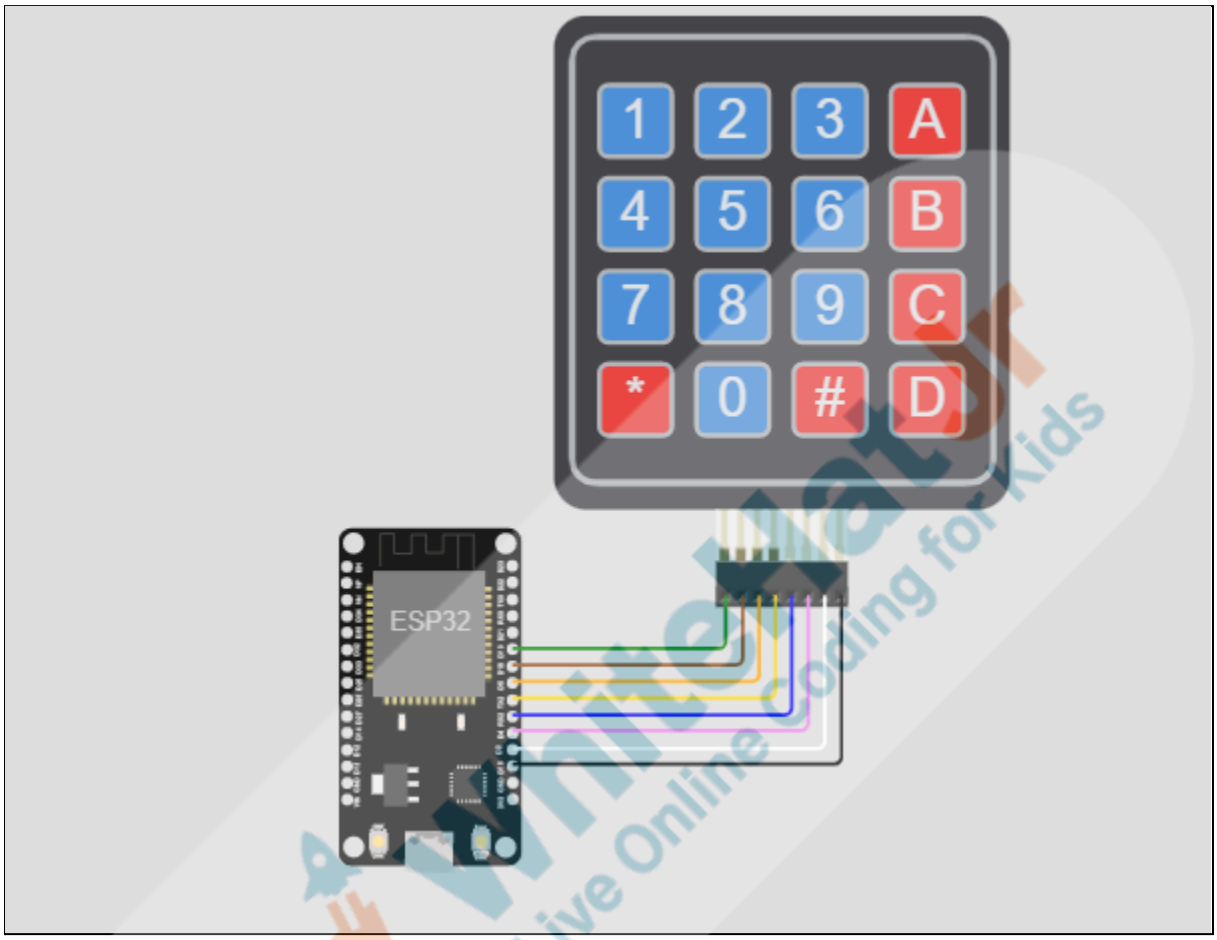
The circuit of this project consists of an ESP32 Controller, and a Keypad.

● Select Keypad from the simulator list. Connect as per the below instructions:

<table>
<tr><th>KEYPAD</th><th>ESP32 PIN</th></tr>
<tr><td>R1</td><td>GPIO 19</td></tr>
<tr><td>R2</td><td>GPIO 18</td></tr>
<tr><td>R3</td><td>GPIO 5</td></tr>
</table>
</td>
<td></td>
</tr>
</table>

R4	**TX2**
C1	**RX2**
C2	**GPIO 4**
C3	**GPIO 2**
C4	**GPIO 15**

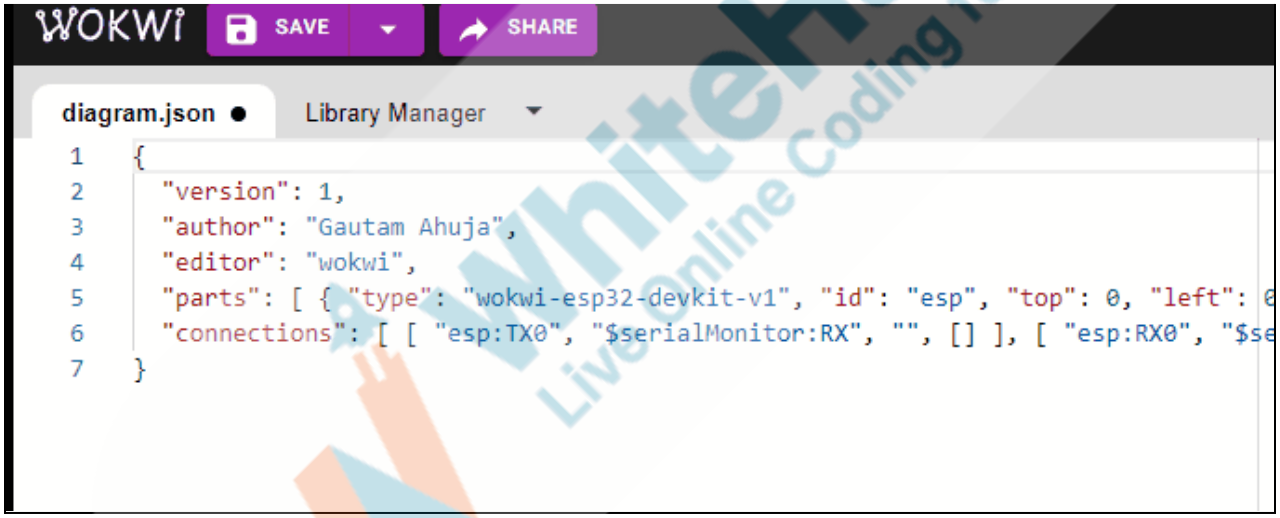
*Note: Wire color can be changed by **clicking over it** and selecting the color, or via **diagram.json** file. Go to the diagram.json wire and change the color of the wire. Any design changes or color changes can be done via the diagram.json file. Keep the track of the component and change the design settings.*



Now it's time to write the code. For that, first go to the **sketch.ino** file and **delete** it.	

Open the [Teacher Activity 1] link and download all the files.

Import **sketch.ino** files in your project.



To build this game, we will be needing 5 global variables as,

- **long int current_time = 0,** which will hold the number of milliseconds passed, since the **ESP32** board began running this current program.
- **long int prev_time = 0,** which will be used to create a time interval, along with the **current_time** variable.

- **int threshold = 4000,** which will be used to store a threshold time interval (in milliseconds), at which our loop will break and restart. Currently, it holds a value of **4000**, which means our loop will restart at an interval of **4000 ms or 4 seconds.**

- **char keypressed = '\0',** which will be used to check which key is pressed on a keypad. Currently, it holds a **null character**.

- **int score = 0,** which will be used to hold the current value score made by the user.

```
long int current_time = 0;
long int prev_time = 0;
int threshold = 4000;
char keypressed = '\0';
int score = 0;
```

Our teacher activity part of the program can be broken down into the following parts,

1) Picking and displaying **random numbers** from our 2D **keys** array.

2) Creating **multiple** hints for the user.

3) Picking up a **single hint** from the **multiple created hints** in the above step.

So let's start with the first part, where we are trying to pick up a **random character** from our **2D keys array.**

If we want to pick up an element from a 2D array, we should have its **row number** and **column number**. First, let's try to generate these **randomly.**

For that, first let's use the **random() method** as,

byte random_row = random(1,5) **byte random_column = random(1,5)** The above lines of code will generate a random number from **1** to **4 (excluding 5),** and store it in variables named as **random_row** and **random_column.** Now, let's use these randomly generated row and column numbers to pick up an element from our array as, **char random_element =** **keys[random_row - 1][random_column - 1]** Can you tell why we have written **random_row - 1,** instead of just **random_row**? We know that elements of an array are **indexed** from **0 onwards**, and we are generating random numbers from **1 onwards**, so a **'-1'** will compensate for that.	**ESR** : Varied

```
byte random_row = random(1,5);
byte random_column = random(1,5);
char random_element = keys[random_row-1][random_column-1];
```

Next, the player should know the **row number** and **column number** of the element, so that they can press the correct key on the keypad, within the specified **threshold** time. For that, we can simply display the row and column number for an element as, **Row : 2 Column : 1,** but this would make our game really simple, as the player would simply look at **numbers** only, assuming the first number as **row** and second as **column**. To make things a bit more complex, let's try to switch the **row** and **column** position randomly as, **Row : 2 Column : 1 or,** **Column : 2 Row : 1 or,**	

Row : 2 Column : 3 or,
Column : 1 Row : 4

For that, let's create 2 strings as,

String hint1 = "Row : " + String(random_row);
String hint2 = "Column : " + String(random_column);

Finally, let's create a variable named **random_hint**, which will help us to toggle the messages randomly as,

byte random_hint = random(1,3);

```
String hint1 = "Row : " + String(random_row);
String hint2 = "Column : " + String(random_column);
byte random_hint = random(1,3);
```

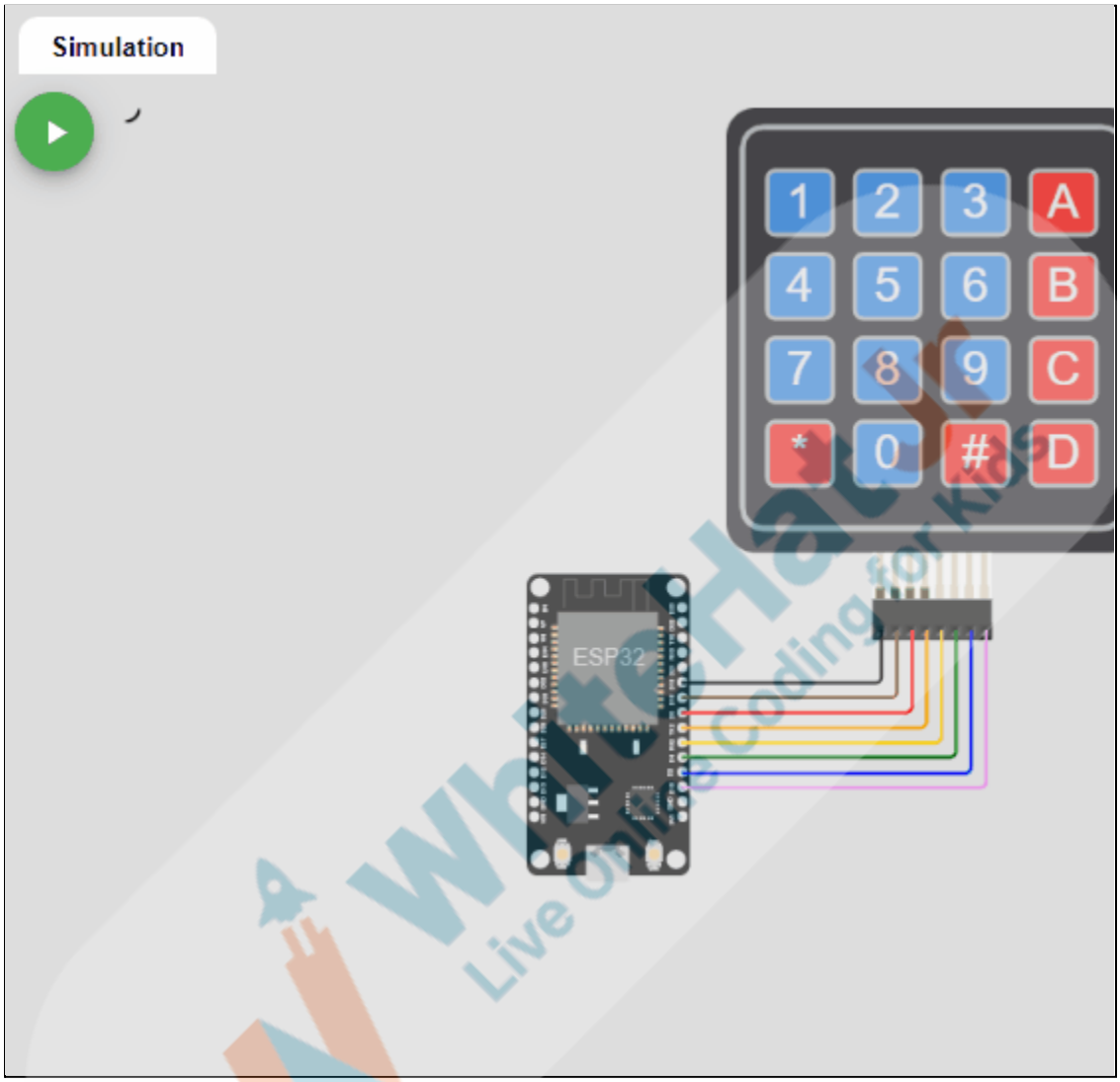
Once we have our algorithm created, let's print these messages randomly using an **if-else** conditional statement as,

if (random_hint == 1)Serial.println(hint1 + "\t" + hint2);
else Serial.println(hint2 + "\t" + hint1);

The above code will either print **hint1 (row number first)** followed by **hint2 (column number second)** or **hint2 (column number first)** followed by **hint1 (row number second)** randomly.

```
if (random_hint  ==  1)Serial.println(hint1 + "\t" + hint2);
else Serial.println(hint2 + "\t" + hint1);
```

The output for the following code will look like,



Student Stops Screen Share	
We have one more class challenge for you. Can you solve it? Let's try. I will guide you through it.	

STUDENT-LED ACTIVITY- 15 mins

- **Ask the student to press the ESC key to come back to the panel.**

- **Guide the student to start Screen Share.**
- **The teacher gets into Full Screen.**

Student Initiates Screen Share

ACTIVITY

- **Keypad Connections.**
- **Algorithm to write the game loop.**

Teacher Action	Student Action
Teacher helps the students	Student open wokwi simulator.

Let's try to create the circuit diagram first.

Step -1:Select the components

- 1 x **ESP32**
- 1 x **Keypad**: 4 Rows (R1-R4) and 4 Columns (C1-C4)

Step -2: Let's do connections:

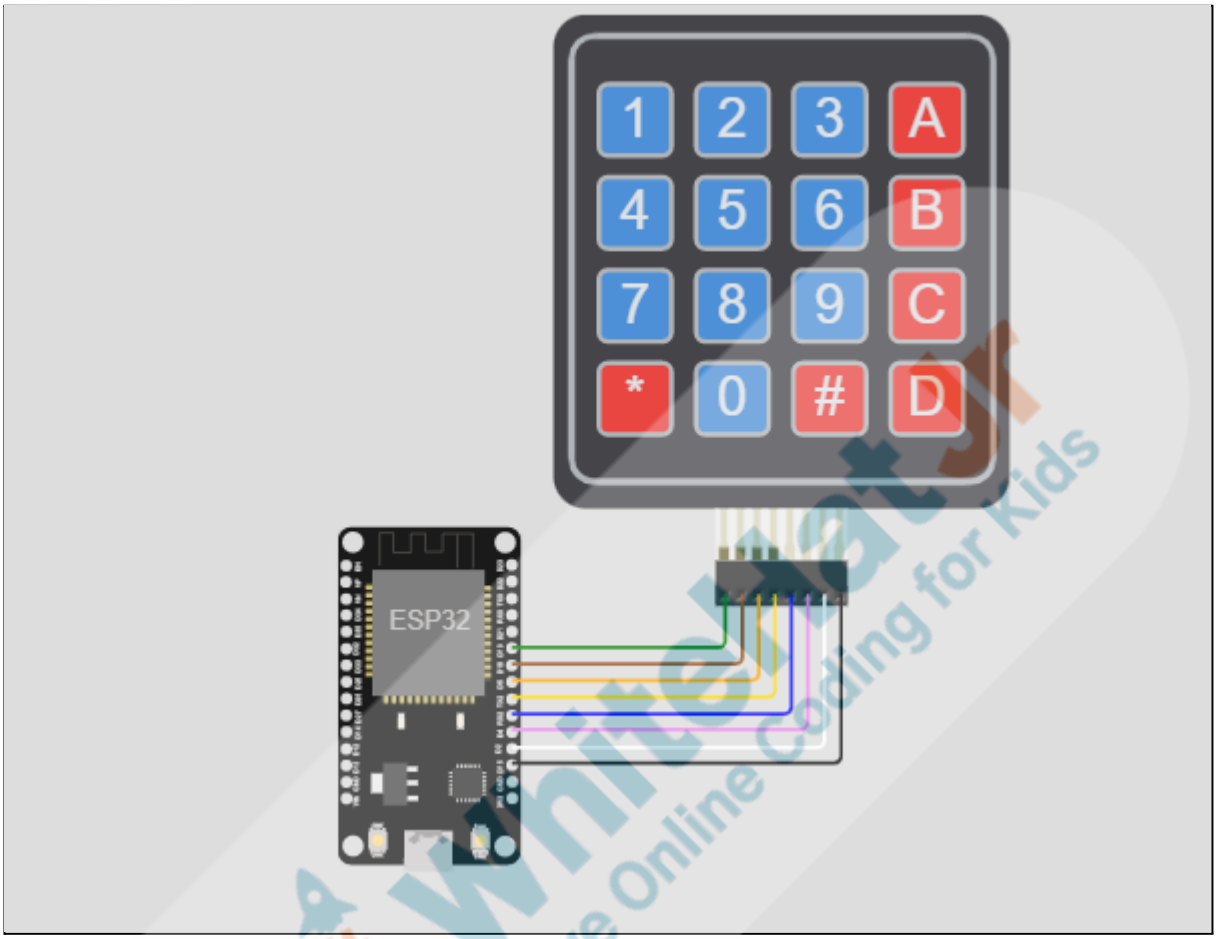
The circuit of this project consists of an **ESP32** Controller, and a **Keypad**.

- Select **Keypad** from the simulator list. Connect as per the below instructions:

KEYPAD	ESP32 PIN
R1	GPIO 19
R2	GPIO 18
R3	GPIO 5
R4	TX2

C1	**RX2**
C2	**GPIO 4**
C3	**GPIO 2**
C4	**GPIO 15**

*Note: Wire color can be changed by **clicking over it** and selecting the color, or via **diagram.json** file. Go to the diagram.json wire and change the color of the wire. Any design changes or color changes can be done via the diagram.json file. Keep the track of the component and change the design settings.*



Delete the **sketch.ino** file which is already there in the project and upload the new **sketch.ino** file from the Student activity link.

Next, we have to write an algorithm, or create a **'game loop'** where we can,

1) Listen to keypad inputs **continuously**.
2) Start the **timer**.
3) Check if the **threshold** time is crossed.
4) Write the **winning** conditions.

5) Write the **losing** conditions.

6) **Break** the loop, so that we can repeat the process again.

So let's start with the first part, where we have to listen to the **player inputs continuously**.

To do things **continuously**, let's use an **infinite while** loop, where we can listen to the player inputs using the **getKey() method.** If the user presses a key, let's store the input in the **keypressed** variable as,

```
while (true){

   char key = k.getKey();
   If (key){

      Keypressed = key;
   }
}
```

```
while (true){


  char key = k.getKey();


  if (key){
    keypressed = key;
  }
```

Moving on to the second part, let's start recording the time, using the **millis()** method as,

current_time = millis(),

The **millis() method** counts the number of **milliseconds** elapsed, since the **ESP32** board began running the current program.

Can you try and tell the difference between the **millis()** method and the **delay()** method? The major differences between them are, ● **millis()** method is a **non blocking method**, i.e., it doesn't **pause** the program while running. ● **millis()** method is more accurate than the **delay()** method, in terms of executing a task at a specific interval.	**ESR** : Varied
`current_time = millis();`	
For the third part of the program, let's use an **if** statement to check if the **threshold** time has been crossed or not as, **if (current_time - prev_time == threshold) {** **}**	
`if (current_time - prev_time == threshold){`	
Now for the **winning** condition, let's check if the **key pressed** by the player is same as the one we have randomly generated as, **if (keypressed == random_element){** **}** Can you tell what all actions we need to take whenever the player guesses the right key? *Note : Let the student try and answer.* If the player guesses the right key key, we have to, a) Increment the **score** variable by 1 and write the **game won** condition as,	**ESR** : Varied

```
score++;
if (score > 6){
   Serial.println("You WON!");
   while(1); }
```

Note : while(1) will make sure that our program gets stuck in an infinite loop if we have won the game.

b) Display the **"Correct guess"** message and the **current score** as,

```
Serial.println("Correct guess, score : " + String(score))
```

Note : + sign can be used to concatenate 2 strings.

c) To make the game a bit more exciting, let's **decrease** the **threshold** by **500 ms**, every time the player guesses the right key. This will give the user less time to guess the next answer, thus making the game more interesting. Let's write the code for it as,

```
threshold = threshold - 500;
if (threshold < 500)threshold = 500;
```

Note : Using the if condition, we made sure that whenever the threshold value goes under 500 ms, it stays as 500 ms.

```
if (keypressed  ==  random_element){
  score++;
  if (score > 6){
    Serial.println("You WON!");
    while(1);
  }
  Serial.println("Correct guess, score : " + String(score));
  threshold = threshold - 500;
  if (threshold < 500)threshold = 500;
}
```

Let's write the **losing** condition in an **else** statement as,

else{ **Serial.println("You LOSE!");** **while(1);** **}** The above code will print the **You LOSE!** Message whenever the player guesses the wrong key. After that, the program will get stuck in an **infinite** loop.	

```
else{
  Serial.println("You LOSE!");
  while(1);
}
```

Now, let's update our **prev_time** variable, so that we can run our code at proper **threshold** intervals as, **prev_time = current_time** Finally, let's **break** the game loop using a **break** statement.	

```
prev_time = current_time;
break;
```

The output for the code will look like,	

Great, we have now completed our '**Guess Game**' using a 4x4 Keypad.

Teacher Guides Student to Stop Screen Share

WRAP-UP SESSION - 05 mins

Activity details

Following are the WRAP-UP session deliverables:

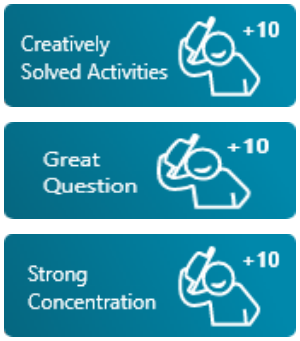
- Appreciate the student.
- Revise the current class activities.
- Discuss the quizzes.

WRAP-UP QUIZ
Click on In-Class Quiz

Activity Details

Following are the session deliverables:
- Explain the facts and trivia
- Next class challenge
- Project for the day
- Additional Activity (Optional)

FEEDBACK
- **Appreciate and compliment the student for trying to learn a difficult concept.**
- **Get to know how they are feeling after the session.**
- **Review and check their understanding.**

Teacher Action	Student Action
You get "hats-off" for your excellent work! In the next class, we will learn about servo motors.	*Make sure you have given at least 2 hats-off during the class for:* 

PROJECT OVERVIEW DISCUSSION
Refer the document below in Activity Links Sections

Teacher Clicks	✖ End Class
ADDITIONAL ACTIVITIES (Optional)	
Additional Activities	

ACTIVITY LINKS		
Activity Name	**Description**	**Links**
Teacher Activity 1	Simulator	https://wokwi.com/
Teacher Activity 2	Keypad	https://docs.wokwi.com/parts/wokwi-membrane-keypad
Teacher Reference 1	Teacher Activity Reference Code	https://github.com/procodingclass/PRO-C260-Teacher-Activity.git
Teacher Reference 2	Reference Code	https://github.com/procodingclass/PRO-C260-Reference-Code.git
Teacher Reference 3	Project	https://s3-whjr-curriculum-uploads.whjr.online/78f037b5-2812-4b98-9fc4-c10a2f9d911c.pdf
Teacher Reference 4	Project Solution	https://github.com/procodingclass/PRO-C260-Project-Solution

Teacher Reference 5	In-Class Quiz	https://s3-whjr-curriculum-uploads.whjr.online/9c172168-cd06-4275-b6a1-29b5f0f1a782.pdf
Student Activity 1	Simulator	https://wokwi.com/
Student Activity 2	Keypad	https://docs.wokwi.com/parts/wokwi-membrane-keypad
Student Activity 3	Student Activity Reference Code	https://github.com/procodingclass/PRO-C260-Student-Activity.git