KEYPAD

What is our GOAL for this CLASS?

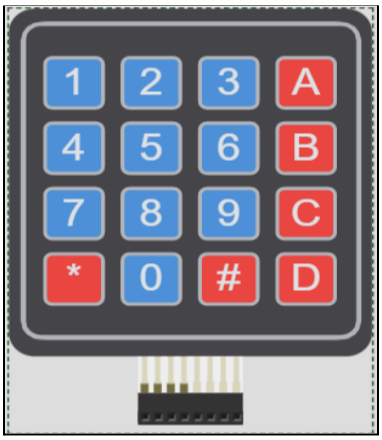
In this class, we learned how to use a **Keypad.**

What did we ACHIEVE in the class TODAY?

- We learned about Keypad Connections
- We learned about Keypad Working
- We learned about Secret Password

Which CONCEPTS/ CODING BLOCKS did we cover today?

- We learned about Membrane Keypads
    - Membrane keypads are made of thin, flexible membrane material. They do come in many sizes 4×3, 4×4, 4×1, etc. Regardless of their size, they all work in the same way

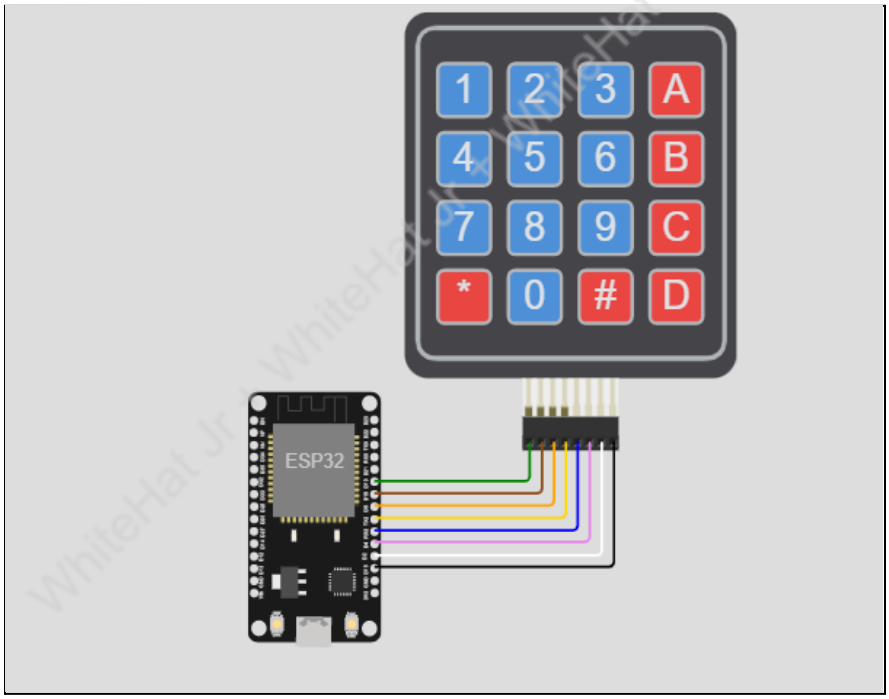


- Working of 4 x 4 keypad:
    - Here 4x4 means 4 rows (R1 - R4) and 4 columns (C1-C4). In total it has 16 keys.
    - Beneath each key, there is a special membrane switch.
    - All these membrane switches are connected to each other with conductive trace underneath the pad forming a matrix of a 4×4 grid.
    - Pressing a button shorts one of the row lines to one of the column lines, allowing current to flow between them.
- A microcontroller can scan these lines for a button-pressed state. To do this, it follows the below procedure.
    - The microcontroller sets all the column and row lines to input.
    - Then, it picks a row and sets it HIGH.
    - After that, it checks the column lines one at a time.
    - If the column connection stays LOW, the button on the row has not been pressed.
    - If it goes HIGH, the microcontroller knows which row was set HIGH, and which column was detected HIGH when checked.
    - Finally, it knows which button was pressed that corresponds to the detected row & column.

How did we DO the activities?

1. Gather the material from the IoT kit Collect the material
   - 1 x ESP32
   - 1 x Keypad: 4 Rows (R1-R4) and 4 Columns (C1-C4)

2. Do connections:
   - **Keypad R1** pin is connected to **ESP32 GPIO 19** PIN.
   - **Keypad R2** pin is connected to **ESP32 GPIO 18** PIN.
   - **Keypad R3** pin is connected to **ESP32 GPIO 5** PIN.
   - **Keypad R4** pin is connected to **ESP32 TX2** PIN.
   - **Keypad C1** pin is connected to **ESP32 RX2** PIN.
   - **Keypad C2** pin is connected to **ESP32 GPIO 4** PIN.
   - **Keypad C3** pin is connected to **ESP32 GPIO 4** PIN.
   - **Keypad C4** pin is connected to **ESP32 GPIO 15** PIN.



3. Write the program:
   - Include the keypad library to access the keypad application. This library takes care of setting up the pins and pulling the different columns and rows.
   - set the number of rows and columns on the keypad
   - define **row_num 4**
   - define **col_num 4**
   - As we are using **4 x 4** matrix

```
#include <Keypad.h>

#define row_num     4 // four rows
#define col_num  4 // four columns
```

- **char** is a keyword, **keys** is a variable, write all the keys map array for both Rows and Columns.

```
char keys[row_num][col_num] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
```

- Define all the pin numbers for row and column
- Byte is the keyword that is used to store **row_pins, row_pins** is the variable that will store all the rows pins.
- **col_pins(col_num)** which will store all the column pins.

```
byte row_pins[row_num]     = {19, 18, 5, 17}; // GIOP19, GIOP18, GIOP5, GIOP17(TX0) connect to the row pins
byte col_pins[col_num] = {16, 4, 2, 15};   // GIOP16,(RX0) GIOP4, GIOP0, GIOP2 connect to the column pins
```

- Create the keypad object for the keypad class to access all the keys.It will access all **rows_pins,col_pins** along with **row_num** and **col_num**

```
Keypad keypad = Keypad( makeKeymap(keys), row_pins, col_pins, row_num, col_num);
```

- Initialize using **void setup()** function
  - **void setup()** is used to initialize
  - **Serial.begin() Serial.begin(9600)** is used for data exchange data speed. This tells the Arduino to get ready to exchange messages with the Serial Monitor at a data rate of 9600 bits per second. That's 9600 binary ones or zeros per second and is commonly called a baud rate.

```
void setup() {
  Serial.begin(9600);
}
```

4. **Execution of the main process:**

- **void loop()** function is used to execute the main process.
- **Keypad.getKey()** returns the keycode of the pressed key,. If the key is pressed return the keycode. The keycode is retrieved from the keymap array.

```
void loop() {

  //  getkey method returns a character
  char key = keypad.getKey();

  if (key)
  {
    //  if '#' pressed , check password
    if (key  ==  '#'){

      Serial.println();

      if (input.compareTo(password)  ==  0){
        Serial.println("Access granted, welcome !");
        while(true);
      }

      else{
        Serial.println("Access denied!");
        Serial.print("Try again : ");
      }

      //  clear the input string
      input = "";
    }
```

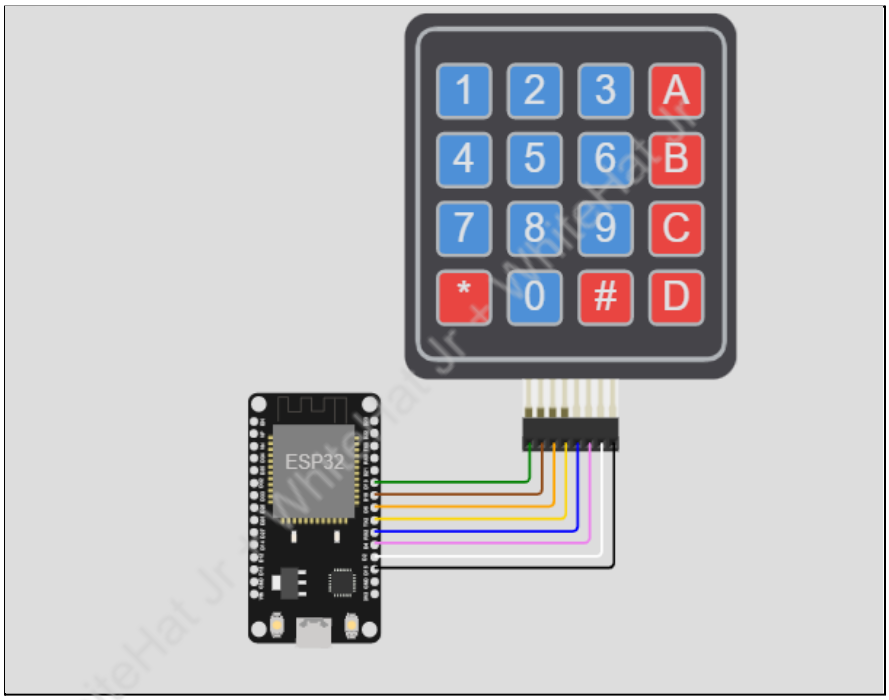
5. **Output:**
   - Click on the Save button and then click on the simulation button
   - Press the key and see the output on the Serial Monitor of the simulator.
   - Just press the keys and you will get the output.

```
3
4
5
6
7
8
9
```

6. Create a security password lock application.

7. **Gather the material from the IoT kit:**
   - 1 x ESP32
   - 1 x Keypad: 4 Rows (R1-R4) and 4 Columns (C1-C4)

8. **Do connections:**
   - Keypad **R1 pin** is connected to **ESP32 GPIO 19** PIN.
   - Keypad **R2 pin** is connected to **ESP32 GPIO 18** PIN.
   - Keypad **R3 pin** is connected to **ESP32 GPIO 5** PIN.
   - Keypad **R4 pin** is connected to **ESP32 TX2** PIN.
   - Keypad **C1 pin** is connected to **ESP32 RX2** PIN.
   - Keypad **C2 pin** is connected to **ESP32 GPIO 4** PIN.
   - Keypad **C3 pin** is connected to **ESP32 GPIO 4** PIN.
   - Keypad **C4 pin** is connected to **ESP32 GPIO 15** PIN.



9. Write the program:
   - Include the keypad library to access the keypad application. This library takes care of setting up the pins and polling the different columns and rows.
   - set the number of rows and columns on the keypad
   - to define **row_num 4**
   - define **col_num 4**
   - As we are using **4 x 4** matrix

```
#include <Keypad.h>

#define row_num     4 // four rows
#define col_num  4 // four columns
```

10. **Set the** password for the same

- **Store password** in variable password i.e “11111”, it can be anything . Just save in the string password.
- **String input** to save user input from the user and it will match the input with password.

```
String password = "111111";
String input = "";
```

- **char** is a keyword, **keys** is a variable, write all the keys map array for both Rows and Columns.

```
char keys[row_num][col_num] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
```

- Define all the pin numbers for row and column
- Byte is the keyword that is used to store **row_pins, row_pins** is the variable which will store all the rows pins.
- **col_pins(col_num)** which will store all the column pins.

```
byte row_pins[row_num]     = {19, 18, 5, 17}; // GIOP19, GIOP18, GIOP5, GIOP17(TX0) connect to the row pins
byte col_pins[col_num] = {16, 4, 2, 15};   // GIOP16,(RX0) GIOP4, GIOP0, GIOP2 connect to the column pins
```

- Create the keypad object for the keypad class to access all the keys. It will access all **rows_pins,col_pins** along with **row_num** and **col_num**

```
Keypad keypad = Keypad( makeKeymap(keys), row_pins, col_pins, row_num, col_num);
```

- Initialize using **void setup()** function
  - **void setup()** is used to initialize
  - **Serial.begin() Serial.begin(9600)** is used for data exchange data speed. This tells the Arduino to get ready to exchange messages with the Serial Monitor at a data rate of 9600 bits per second.That's 9600 binary ones or zeros per second and is commonly called a **baud rate.**
  - **Serial.print** is used to print or display the user_input password on Serial Monitor

```
void setup() {
  Serial.begin(115200);
  Serial.print("Enter password : ");
}
```

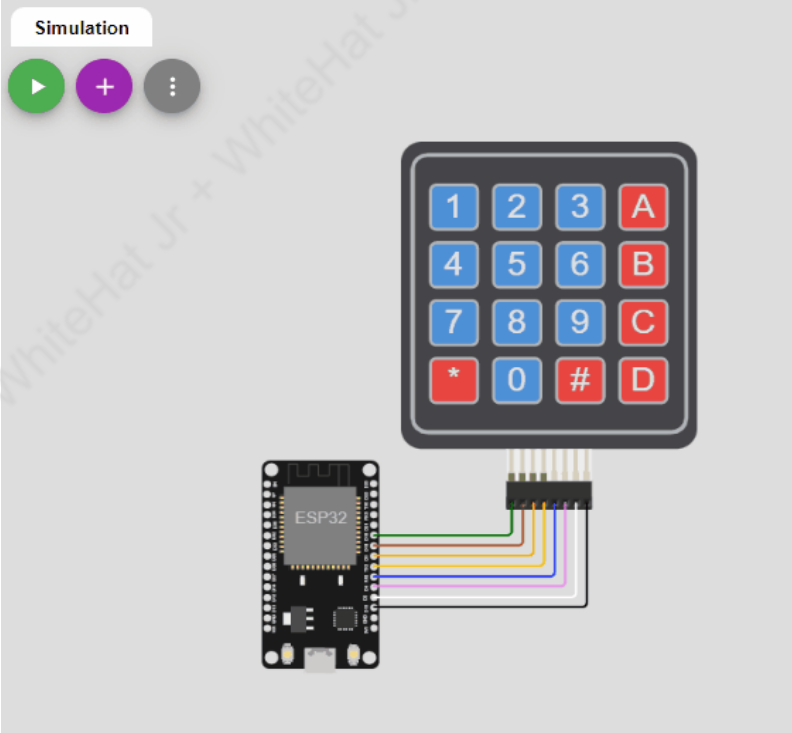
- **void loop()** function is used to execute the main process.
- **Keypad.getKey()** returns the keycode of the pressed key,. If the key is pressed return the keycode. The keycode is retrieved from the keymap array.
- **#** is used to check the password. After entering the six digit password press **#** key to check the correct password. compare method will check the user input with store password.
- If it doesnt match with password then it will display the Access Denied.
- if input string match with the password string print **Access granted, welcome!** else print **Access denied.Try Again!**

```
void loop() {

  //  getkey method returns a character
  char key = keypad.getKey();

  if (key)
  {
    //  if '#' pressed , check password
    if (key  ==  '#'){

      Serial.println();

      if (input.compareTo(password)  ==  0){
        Serial.println("Access granted, welcome !");
        while(true);
      }

      else{
        Serial.println("Access denied!");
        Serial.print("Try again : ");
      }

      //  clear the input string
      input = "";
    }
```

- Now suppose user entered wrong password and user wants to try new password.
- To try password again press the ***** and write the password again.
- ***** will clear the **input** string and get ready to take another input from the user.
- **Serial.println()** is used to print the statement.After clearing it will display **(" Password cleared, enter again")**
- Concat function will used to add the string.

```
    else if (key  ==  '*'){

      //  clear the input string
      input = "";
      Serial.println();
      Serial.print("Password cleared, enter again : ");

    }

    else{

      //  adding character to input string
      input.concat(key);
      Serial.print(key);
    }
  }
}
```

11. Output
    - Click on the Save button and then click on the simulation button
    - Press the key and see the output on the Serial Monitor of the simulator.
    - Just press the keys and you will get the output



What's NEXT?

In the **next class**, we will learn about **WEB SERVERS.**

Expand Your Knowledge

To know more about **Keypad** click here.