4x4 Matrix Keypad Module Explained

A **4x4 matrix keypad module** is a simple input device commonly used in microcontroller projects to allow users to input numeric or alphanumeric data. It consists of **16 buttons** arranged in 4 rows and 4 columns. Here's how it works and how the buttons are assigned values:

Physical Structure

1. **16 Buttons**:
    - The keypad has 16 buttons in a 4x4 grid.
    - These buttons do not have intrinsic values like "1", "A", or "*". Physically, they are just momentary push buttons.
2. **Row and Column Pins**:
    - Instead of wiring all 16 buttons individually, the keypad uses **row pins** (4) and **column pins** (4).
    - This makes it efficient, requiring only 8 connections (4 rows + 4 columns) for 16 buttons.

Working Mechanism

- **Button Matrix**:
    - Each button is located at the intersection of a specific **row** and **column**.
    - For example, pressing button "5" closes the connection between **row 2** and **column 2**.
- **Scanning Technique**:
    - The microcontroller sends signals to the rows and reads signals from the columns.
    - It activates one row at a time and checks which column detects a signal.
    - This method allows the microcontroller to identify the specific button pressed.

Button Assignment

- **Default Behavior**:
    - The buttons themselves are just physical switches with labels.
    - They are assigned specific characters or values (e.g., "1", "A", "*") through **software logic** in the code.

- **Custom Mapping**:
    - A **keymap array** is defined in the code. It matches each button's row-column combination with a character.
    - For example:

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

    - Here:
        - The button at row 1, column 1 is assigned `'1'`.
        - The button at row 3, column 4 is assigned `'C'`.

You can change these assignments to represent different characters or numbers.

Example of a Button Press

1. Suppose you press the button labeled "6".
2. The keypad hardware closes the connection between **row 2** and **column 3**.
3. The microcontroller detects this by scanning the matrix.
4. Using the keymap array, it maps row 2, column 3 to the character `'6'`.

Why They Are Just Normal Buttons

- Each button is simply a **momentary push button** that makes or breaks an electrical connection.
- The labels like `'1'`, `'A'`, or `'#'` are **logical representations** defined in the code.
- The actual hardware doesn't "know" about these labels; it only senses whether a circuit is closed (button pressed) or open.

Advantages of Matrix Design

1. **Efficient Wiring**:
    - Reduces the number of connections needed from 16 (one for each button) to 8 (4 rows + 4 columns).
2. **Scalable**:
    - The same principle can be applied to keypads of other sizes, like 3x3 or 5x5.

Practical Use

- By customizing the **keymap array**, you can assign any value or function to the buttons.
- For instance:
    - Numeric input for a calculator.
    - Menu navigation in an embedded system.
    - Security code entry in a lock system.

The software-defined mapping ensures flexibility, making the same hardware usable for various applications.

4x4 Matrix Membrane Keypad Using Arduino Nano

Using Keypads with Arduino - Build an Electronic Lock