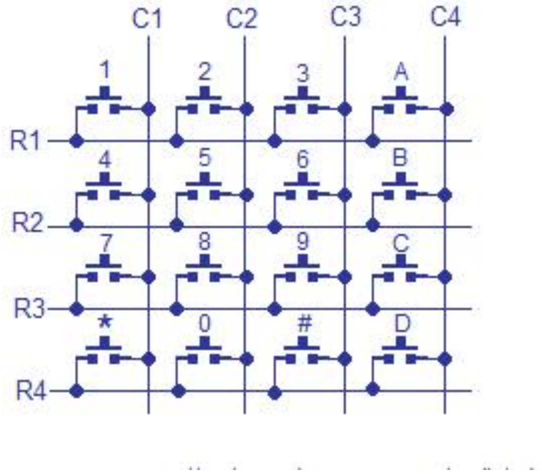
Interfacing of matrix keyboard with arduino uno

Matrix keypad is a small compact input device that accepts user inputs and processed by Microcontrollers. You might have seen this in most commonly used devices like Calculators, Digital locks, petrol pumps etx. It comes in different types, one of them is membrane keypads, it is thinner in size..



Initially, all rows are set to 0(LOW) and all Columns are set to 1(HIGH). When a key press occurs the **column pin** will get contacted to the **row pin** and makes the entire column state to low. To identify the exact pin at the column, we need to **scan each row by sending 1 (HIGH)** and **read the state at Column pins**. The column which changes the state from 0(LOW) to 1(HIGH) then that is the location of the pressed key (Passes the HIGH signal from Row to Column pin).

In the idle state, the row and column will be like this,

Row	R1 R2 R3 R4 – 0000
Column	C1 C2 C3 C4– 1111

The keys at each row are connected to a pin called ROW pin and we have 4 such ROWS named as R1, R2, R3, and R4. Similarly, the COLUMN Pins connected to a pin called a COLUMN Pin and we have 4 such Columns named as C1, C2, C3, and C4.

Detecting the keypress:

Stage 1: No Key pressed

In Idle state C1: C4 – 1111 and R1: R4 – 0000

Stage 2: Key Pressed

Now if we press the key 5. The entire 2nd Column will change to low since the LOW (0) signal from ROW pin flows through Column pin. Now the column values will become C1: C4 – 1011.

Stage 3: Scanning the key press

The Arduino knows the key is in which column. But we have 4 key in that column. We need to detect the exact key by scanning the Rows. The scanning happens by sending 1 (HIGH) to the Row pins one by one and read at column pins. This will return a HIGH (1) Signal only at the pressed key location.

Scanning ROW1:

Arduino set R1 as 1 (HIGH) and also monitors C1: C4 for state change. The key is not in ROW1 and won't make any change in C1: C4. So no key is pressed in R1.

Scanning ROW2:

Now Arduino sets R2 as 1 (HIGH) and monitor C1: C4. Here C2 changes from LOW (0) to HIGH (1). So the key location is R2C2 which is 'Key 5' in our membrane keypad. So the Arduino maps the character with sketch and prints it accordingly.

Scanning ROW3:

Now Arduino set R3 as 1 (HIGH) and also monitors C1: C4 for state change. The key is not in ROW3 and won't make any change in C1: C4. So no key is pressed in R3.

Scanning ROW4:

Now Arduino set R4 as 1 (HIGH) and also monitors C1: C4 for state change. The key is not in ROW4 and won't make any change in C1: C4. So no key is pressed in R4.

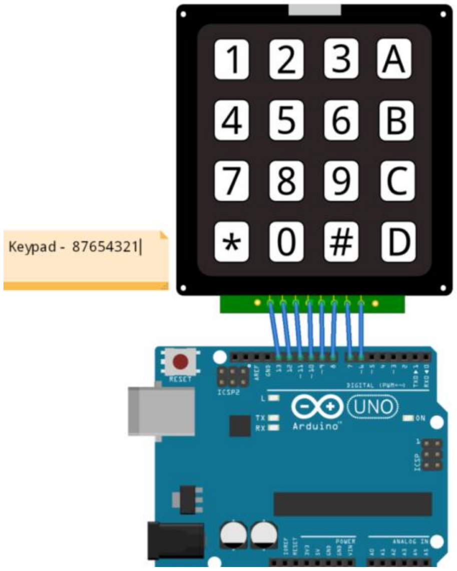
Note: In Arduino entire scan process happens in MicroSeconds (That's why it's called a Microcontroller). But Human can able to press the key by minimum 1s to 600ms. It is more than enough to find the pressed key by Arduino.

Stage 4: Map the pressed key by Arduino Library.

In Arduino sketch, the keypad keys are stored in an Array. A temporary variable stores this key value and can be used with our sketches. They keypad.h written by Mark Stanley and Alexander Brevig has many functions to use the keypad. The updated library also supports multiple key presses.

How to connect with Arduino:

The 4×4 matrix keypad reads the pin in reverse order. So numbering will start from Pin 8. The pin sequence will be 87654321. In which the first four pins are Row Pins – 8765 and remaining pins are column pins – 4321. Connect the keypad row pins 8,7,6,5 with Arduino pins 13,12,11,10. And Keypad Column pins 4,3,2,1 with Arduino pins 9,8,7,6. The connection diagram is given below.



Key pad pin	Arduino uno pin
1	6
2	7
3	8
4	9
5	10
6	11
7	12
8	13

Interfacing matrix keypad with Arduino:

The Arduino library called keypad.h is written by Mark Stanley and Alexander Brevig. Download this library and extract it to Arduino libraries folder.

We used 4×4 Matrix keypad, so the key arrangements and Arduino pins will be instantiated like this

```
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
char keys[ROWS][COLS] = {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};

byte rowPins[ROWS] = {13, 12, 11, 10}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {9, 8, 7, 6}; //connect to the column pinouts of the keypad

Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
```

Now we configured our keypad with Arduino. The key value is stored in a temporary variable called ‘key’. You can print this key variable in the sketch wherever needed.

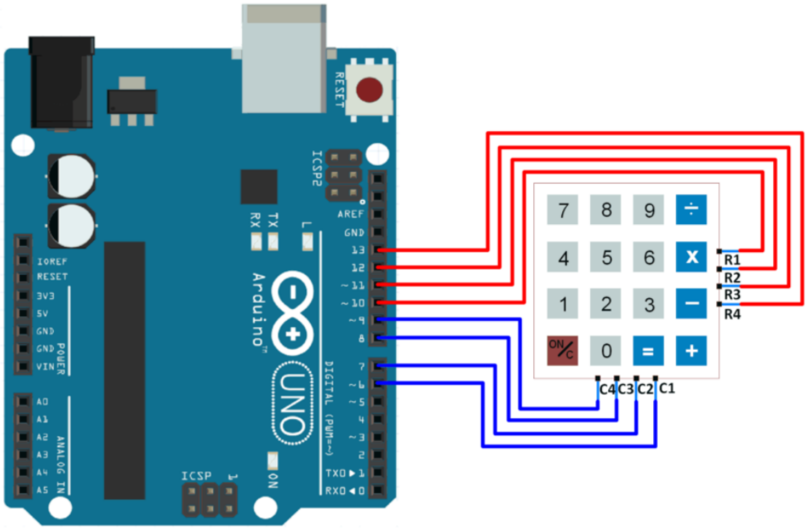
```
char key = keypad.getKey();
```

Complete Code:

Now upload the code to Arduino and open the serial monitor.

```
#include <Keypad.h>
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
char keys[ROWS][COLS] = {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};

byte rowPins[ROWS] = {13, 12, 11, 10}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {9, 8, 7, 6}; //connect to the column pinouts of the keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );

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

void loop(){
char key = keypad.getKey();  //Variable to store pressed key

if (key!= NO_KEY){
Serial.println(key); //Prints the key in serial monitor
}
}
```

Another code and diagram



Interfacing 4x4 Keypad with Arduino UNO

```
#include <Keypad.h>
const byte ROWS = 4; /* four rows */
const byte COLS = 4; /* four columns */
/* define the symbols on the buttons of the keypads */
char hexaKeys[ROWS][COLS] = {
  {'0','1','2','3'},
  {'4','5','6','7'},
  {'8','9','A','B'},
  {'C','D','E','F'}
};
byte rowPins[ROWS] = {10, 11, 12, 13}; /* connect to the row pinouts of the keypad */
byte colPins[COLS] = {6, 7, 8, 9}; /* connect to the column pinouts of the keypad */

/* initialize an instance of class NewKeypad */
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

void setup(){
Serial.begin(9600);
}
void loop(){
  char customKey = customKeypad.getKey();
  if (customKey){
Serial.println(customKey);
  }
}
```