UNIVERSITY OF BATANGAS – LIPA CAMPUS

COLLEGE OF ENGINEERING AND

ARCHITECTURE COMPUTER ENGINEERING

4x4 Keypad Matrix

Experiment #3

MICROPROCESSOR SYSTEMS

NAME: TIAMSIM EDSEL, CASTILLO JOAQUIN, LASAT ALAN

STUDENT NUMBER:

DATE OF SUBMISSION:

April 13 2023

ENGR. CHARLES RAY JUANILLAS

PROFESSOR

Activity Questions:

1. Based on this activity, why did you use byte variable in declaring the number of rows and columns?

The data type of a variable specifies the kind of values it may hold and the actions it can execute. Some are used to store numbers, while others are intended to store text or much more sophisticated data. You may define and initialize a Byte variable by giving it a decimal, hexadecimal, or octal literal, or by using Rows and Columns, as we did in this activity.

2. What does the variable keymap represents?

It utilizes the Keypad library's Keymap function to specify keys arrays such as rowPins, which holds the pin numbers of row pins, colPins, which holds the pin numbers of column pins, ROWS, which holds the number of rows in the keypad, and COLS, which has the number of columns in the keypad. The list of characters to assign to each key, for example, is char userKeymap(). The "makeKeymap()" section is a Keypad.h macro that converts an address to a char pointer.

Results And Discussion:

1. Open Arduino IDE and copy the sample code below. Make sure the Keypad Library is download to your library.

2. Run the code and open the Serial Monitor. Explain how the code works.

```
#include <Keypad.h>

const byte numRows= 4;
const byte numCols= 4;

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

byte rowPins[numRows] = {9,8,7,6};
byte colPins[numCols] = {5,4,3,2};

Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins, numRows,
     numCols);

void setup() {
  // put your setup code here, to run once:
    Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
    char keypressed = myKeypad.getKey();
    if (keypressed != NO_KEY) {
          Serial.println(keypressed);
    }
     }
```

Four columns and four rows make up a 16-button keypad. By pressing a button, one of the row outputs is switched to one of the column outputs. The Arduino can deduce which button was pressed based on this data. When key 1 is hit, for example, column 1 and row 1 are shortened.

3. Create another Arduino program. Create a program that will do the following:

a. Disable the 4 rows with the characters ‘A’, ‘B’, ‘C’, and ‘D’.

```
#include <Keypad.h>

const byte numRows= 4;
const byte numCols= 3;

char keymap[numRows][numCols]=
{
  {'1','2','3'},
  {'4','5','6'},
  {'7','8','9'},
  {'*','0','#'}
};

byte rowPins[numRows] = {9,8,7,6};
byte colPins[numCols] = {5,4,3};

Keypad myKeypad= Keypad(makeKeymap(keymap), rowPins, colPins,
     numRows, numCols);

void setup() {
  // put your setup code here, to run once:
    Serial.begin(9600);
}

void loop() {
  // put your main code here, to run repeatedly:
    char keypressed = myKeypad.getKey();
    if (keypressed != NO_KEY) {
         Serial.println(keypressed);
    }
     }
```

b. Change the function of the keypad matrix to text mode. The numeric keys have an equivalent alphabetic letter by pressing a key once or several times in rapid succession. With each key press, the next letter in sequence must become available.

```
#include <Keypad.h>
 //
 define the keypad pins and layout const byte ROWS = 4; const byte COLS = 4; char
keys[ROWS][COLS] = {
{'1','2','3','A'},
{'4','5','6','B'},
{'7','8','9','C'},
{'*','0','#','D'}
};
byte rowPins[ROWS] = {10, 11, 12, 13}; byte colPins[COLS] = {6, 7, 8, 9}; Keypad
keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
// variables to keep track of the key presses char lastKey = '\0'; unsigned long
lastKeyPressTime = 0; int lastKeyCount = 0;

void setup() {
// initialize the serial port
Serial.begin(9600);
}
void loop() {
 // read the keypad and get the current key
char key = keypad.getKey();

// if a key is pressed, handle it
if (key != NO_KEY) {

// if the current key is the same as the last key and it has been less than 500
milliseconds since the last key press, increment the key count
 if (key == lastKey && millis() - lastKeyPressTime < 500) {
lastKeyCount++;
 }
// otherwise, reset the key count to 1
else {
lastKeyCount = 1;
}

// save the current key and key press time for the next iteration
lastKey = key;
lastKeyPressTime = millis();
 }

 // if no key has been pressed for at least 500 milliseconds and there is a last key,
output the appropriate letter(s) to the serial monitor and reset the key count and last key
```

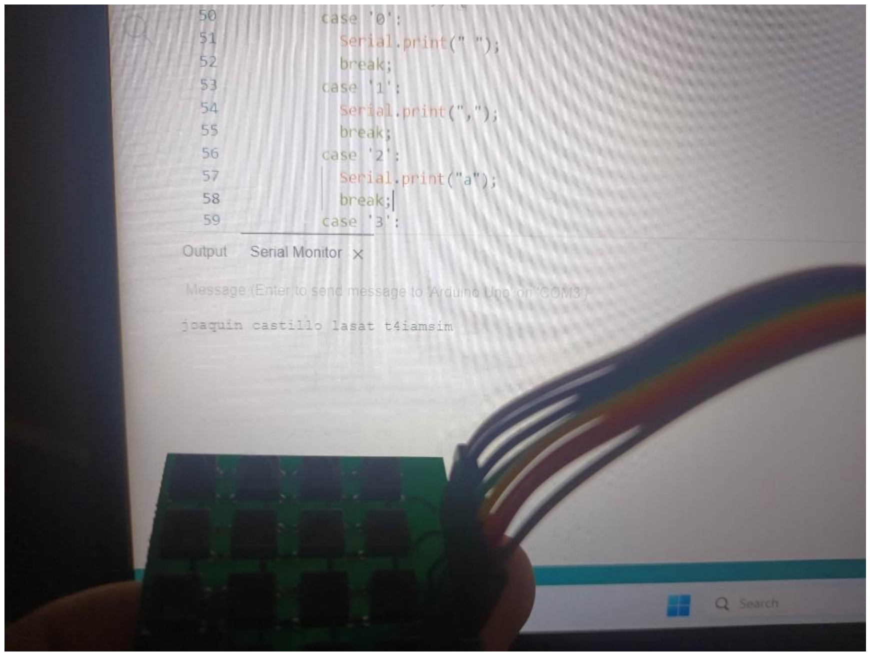
```
if (lastKey != '\0' && millis() –
 lastKeyPressTime >= 500)
 {
if (lastKeyCount ==
1) {
 switch (lastKey) {
case '0':
Serial.print(" ");
break;
case '1':
Serial.print(",");
break;
 case '2':
Serial.print("a");
break; case '3':
Serial.print("d");
break; case '4':
Serial.print("g");
break;
case '5':
 Serial.print("j");
break;
case '6':

Serial.print("m");
break;
case '7':
 Serial.print("p");
 break;
case '8':
 Serial.print("t");
 break;
 case '9':
 Serial.print("w");
 break;
 default:
 break;
 }
 }
else if
(lastKeyCount == 2) {
switch (lastKey)
 { case '0':
Serial.print("0");
break;
case '1':
```

```
Serial.print(".");
 break;
case '2':
Serial.print("b");
break;
 case '3':
 Serial.print("e");
break;
case '4':
 Serial.print("h");
 break;
case '5':
 Serial.print("k");
 break;
case '6':
 Serial.print("n");
 break;
case '7': Serial.print("q");
break;
case '8':
Serial.print("u");
break;
 case '9':
Serial.print("x");
break;
 default:
 break;
 }
 }
 else if (lastKeyCount == 3) {
 switch (lastKey) {
case '1':
Serial.print("?");
 break;
 case '2':
 Serial.print("c");
break;
case '3':
 Serial.print("f");
break;
case '4':
 Serial.print("i");
break;
case '5':
 Serial.print("l");
 break;
```

```
case '6':
Serial.print("o");
break;
case '7':
Serial.print("r");
break;
case '8':
Serial.print("v");
break;
case '9':
 Serial.print("y");
break;
default:
break;
}
 }
 else if (lastKeyCount == 4) {
 switch (lastKey) {
case '1':
Serial.print("!");
break;
 case '2':
Serial.print("2");
break;
case '3':
 Serial.print("3");
break;
case '4':
Serial.print("4");
 break;
case '5':
Serial.print("5");
break;
 case '6':
Serial.print("6");
break;
case '7':
 Serial.print("s");
break;
case '8':
 Serial.print("8");
break;
case '9':
 Serial.print("z");
break;
 default:
```

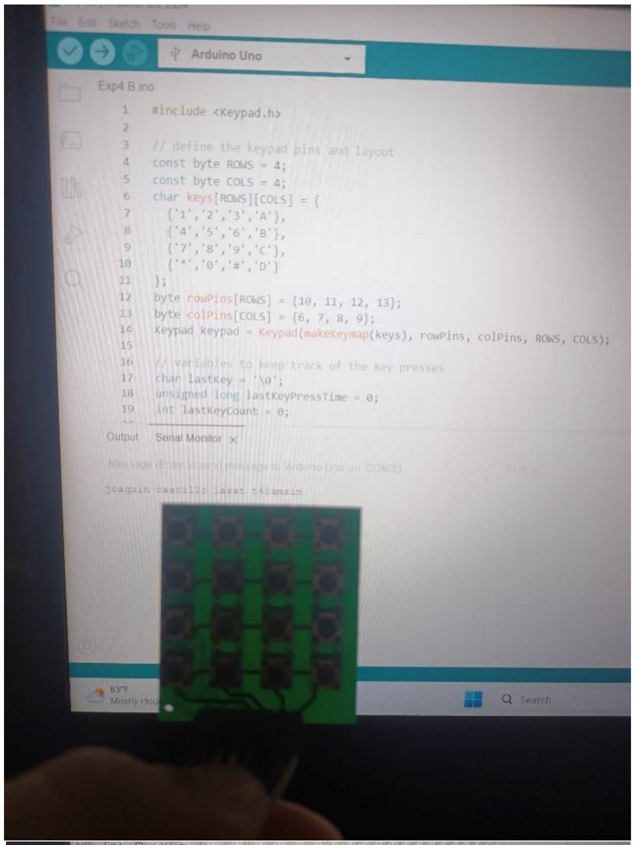
```
break;
}
 }
 else if (lastKeyCount == 5) {
 switch (lastKey) {
 case '7':
 Serial.print("7");
break;
case '9':
 Serial.print("9");
break;
default:
break;
 }
 }
 lastKeyCount = 0;
 lastKey = '\0';
}
}
```

Documentation:



```
        case '0':
          Serial.print(" ");
          break;
        case '1':
          Serial.print(",");
          break;
        case '2':
          Serial.print("a");
          break;
        case '3':
```

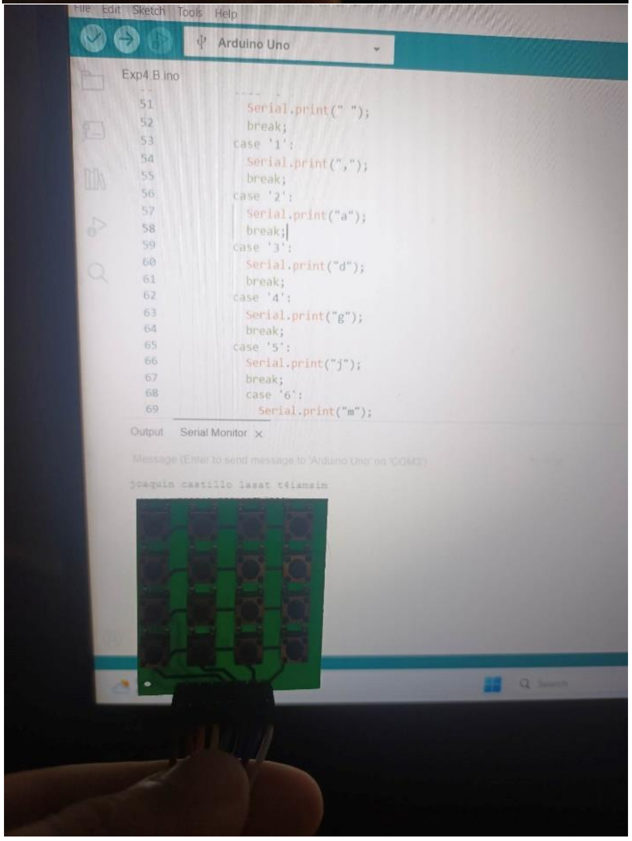
```
joaquin castillo lasat t4iamsim
```



```
#include <Keypad.h>

// define the keypad pins and layout
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {10, 11, 12, 13};
byte colPins[COLS] = {6, 7, 8, 9};
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// variables to keep track of the key presses
char lastKey = '\0';
unsigned long lastKeyPressTime = 0;
int lastKeyCount = 0;
```

Message (Enter to send message to 'Arduino Uno' on 'COM3')

```
joaquin castillo lasat t4iansim
```



```
        Serial.print(" ");
        break;
      case '1':
        Serial.print(",");
        break;
      case '2':
        Serial.print("a");
        break;
      case '3':
        Serial.print("d");
        break;
      case '4':
        Serial.print("g");
        break;
      case '5':
        Serial.print("j");
        break;
        case '6':
          Serial.print("m");
```

Message (Enter to send message to 'Arduino Uno' on 'COM3')

```
joaquin castillo lasat t4iansim
```

Conclusion:

We learn how to use the keypad and link it to the Arduino. Keypad is an Arduino library that allows you to use matrix style keypads. The 4x4 keypad has four rows and four columns. The switch between a column and a row trace is closed when a button is pressed, enabling current to pass between a column pin and a row pin. The row and column pins linked to the button are detected by the Arduino to determine which button is pushed.

https://drive.google.com/drive/u/0/folders/12_oecHhvZQO9vvgzIPMmY-m6S9v57ygO?fbclid=IwAR2tzyFWV6vcbimNN1lVujlPmBnm-AW21NKf0K559rUe3hzhpdWoiKsATVQ