Using Arduinos in Vocational Training

UsingARDinVET

IPSIA "G.Giorgi" - Potenza (Italy)

Keypad Module

Sometimes it is necessary for a human to interact directly with an Arduino project and this means using buttons or keypads. The term keypad usually refers to a series of buttons next to each other. Usually the keys are labelled with letters and numbers, like the examples shown in Figure 1. Keypads do not have to be rectangular, but can be found in a wide range of styles and layouts.

Figure 1: Keypads.

Generally, the keypad buttons are arranged in rows and columns. A 3x4 keypad has 4 rows and 3 columns. Each button in a row is connected to the other buttons in the row by a conductive trace underneath the pad. Each button in a column is connected the same way: one side of the button is connected to all of the other buttons in that column by a conductive trace. Each row and column is brought out to a single pin, for a total of 7 pins. Pressing a button closes the switch between a column and a row trace, allowing current to flow between a column pin and a row pin 2.
The Arduino detects which button is pressed by detecting the row and column pin that's connected to the button. This happens in four steps:

1. First, when no buttons are pressed, all of the column pins are held HIGH, and all of the row pins are held LOW.

2. When a button is pressed, the column pin is pulled LOW since the current from the HIGH column flows to the LOW row pin.

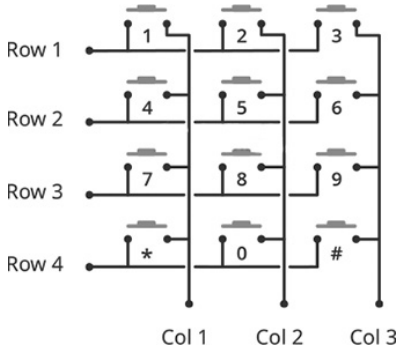

Figure 2: Keypad internal diagram.

3. The Arduino now knows which column the button is in, so now it just needs to find the row the button is in. It does this by switching each one of the row pins HIGH, and at the same time reading all of the column pins to detect which column pin returns to HIGH.

4. When the column pin goes HIGH again, the Arduino has found the row pin that is connected to the button.

Circuit 1. Reading a Keypad

You have a matrix keypad and want to read the key presses in your sketch. For example, you have a telephone-style keypad. Wire the rows and columns from the keypad connector to the Arduino, as shown in figure 3.

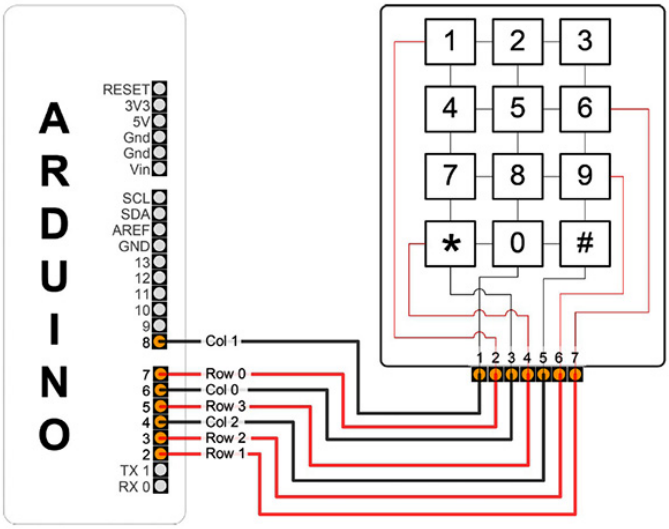

Figure 3: Connecting the keyboard matrix.

The following sketch will print key presses to the Serial Monitor:

```
#include <Keypad.h>
const byte ROWS = 4;
```

```
const byte COLS = 3;
// define the symbols on the buttons of the keypad
char keys[ROWS][COLS] = {
    {'1', '2', '3'},
    {'4', '5', '6'},
    {'7', '8', '9'},
    {'*', '0', '#'}
};
byte rowPins[ROWS] = {7, 2, 3, 5};  // connect to the row pinouts
byte colPins[COLS] = {6, 8, 4};  // connect to the column pinouts
// initialize an instance of class Keypad
Keypad myKeypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  Serial.begin(9600);  // Initialize the serial port
}

void loop() {
  // Get the character input
  char keyPressed = myKeypad.getKey();
  // If there is a character input, sent it to the serial port
  if (keyPressed) {
    Serial.println(keyPressed);
  }
}
```

We'll use the Keypad library by Mark Stanley and Alexander Brevig. This library takes care of setting up the pins and polling the different columns and rows. To install the Keypad library, go to Sketch → Include Library → Manage Libraries and search for keypad. Click on the library, then click install.

Circuit 2. Controlling RGB LED using Keypad

In this circuit, we'll set up an RGB LED that can change colors based on which button you press on a 4x4 keypad (Figure 4). To initiate the assembly, one should connect the keypad to the digital pins on the Arduino. For example, digital pins 4 and 7 to 13 may be utilized for this purpose. The subsequent step involves connecting the RGB LED. This component features four pins: one for the common cathode and one for each of the color channels (red, green, and blue). The common cathode of the LED should be connected to the ground (GND) of the Arduino. The red pin of the LED can be connected to a designated digital pin on the Arduino (pin 6) via a 220Ω resistor. In a similar manner, the green pin should be connected to another digital pin (pin 3), while the blue pin is connected to a third digital pin (pin 5).

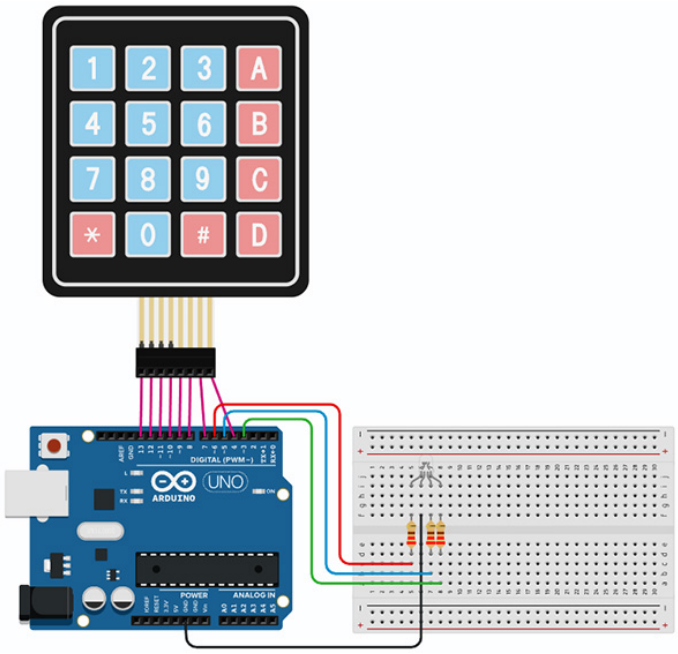

Figure 4: Keypad and RGB LED connection.

This sketch defines the configuration of the keypad, specifying the rows and columns of keys. It establishes the RGB LED pins as output. Within the loop function, the code continuously checks for key presses and utilizes a `switch` statement to determine the appropriate color for the LED based on the pressed key. For example, pressing `1` will illuminate the LED red, `2` will illuminate it green, `3` will illuminate it blue, and `4` will produce a yellow light. If a key that is not recognized is pressed, the LED will be turned off.

```
#include <Keypad.h>

// Define the dimensions of the keypad
const byte ROWS = 4;
const byte COLS = 4;

// Keypad layout
char keys[ROWS][COLS] = {
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

// Define the pins for the keypad
byte rowPins[ROWS] = {12, 11, 10, 9};
byte colPins[COLS] = {8, 7, 6, 5};
```

```
// Create a Keypad object
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

// Define pins for the RGB LED
const int redPin = 4;
const int bluePin = 3;
const int greenPin = 2;

void setup() {
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
  Serial.begin(9600); // Initialize serial communication
}

void loop() {
  char key = keypad.getKey(); // Check for key press

  if (key) {
    Serial.println(key); // Print the pressed key to the Serial Monitor
    switch (key) {
      case '1':
        setColor(255, 0, 0); // Red
        break;
      case '2':
        setColor(0, 255, 0); // Green
        break;
      case '3':
        setColor(0, 0, 255); // Blue
        break;
      case '4':
        setColor(255, 255, 0); // Yellow
        break;
      case '5':
        setColor(0, 255, 255); // Cyan
        break;
      case '6':
        setColor(255, 0, 255); // Magenta
        break;
      case '7':
        setColor(255, 255, 255); // White
```

```
        break;
      default:
        setColor(0, 0, 0); // Off
    }
  }
}

void setColor(int red, int green, int blue) {
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

Circuit 3. Keypad Calculator

This circuit (Figure 5) involves the development of a simple calculator using a 4x4 keypad for user input and the serial monitor for output display. The calculator allows to perform basic arithmetic operations, including addition, subtraction, multiplication, and division.

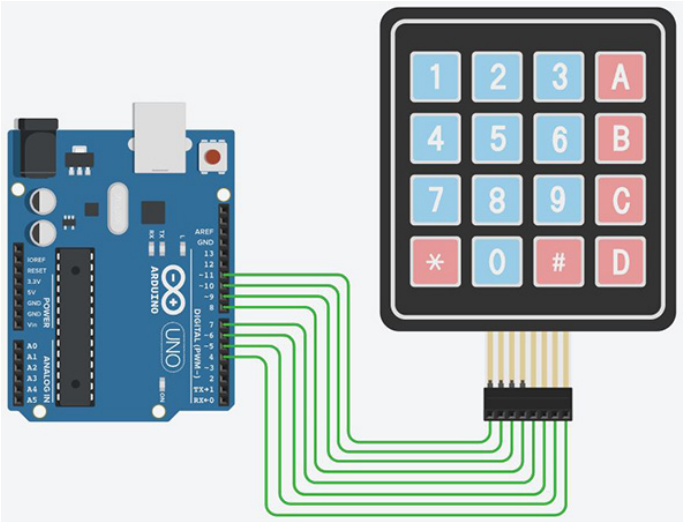

Figure 5: Keypad connection.

The `@` key is specifically designated for initiating a new calculation. Once pressed, the user is prompted to input the first operand by pressing the corresponding numeric key on the keypad. Subsequently, the user inputs the second operand. After entering the second number, the user selects an arithmetic operation by pressing the appropriate operator key. The program then computes the result based on the selected operation and displays it in the Serial Monitor. After the calculator has processed the arithmetic operation and presented the result, the program instructs the user to press the `#` key to initiate a new calculation.

```
#include <Keypad.h>

// Define the dimensions of the keypad
const byte ROWS = 4; // four rows
const byte COLS = 4; // four columns

// Define the keymap
char keys[ROWS][COLS] = {
  {'1', '2', '3', '+'},
  {'4', '5', '6', '-'},
  {'7', '8', '9', '*'},
  {'@', '0', '#', '/'}
};

// Define the row and column pins
byte rowPins[ROWS] = {11, 10, 9, 8}; // connect to the row pinouts of the keypad
byte colPins[COLS] = {7, 6, 5, 4}; // Connect to the column pins of the keypad

// Create the keypad object
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

void setup() {
  // Start the Serial communication
  Serial.begin(9600);
}

void loop() {
  int num1 = -1;
  int num2 = -1;
  char operation = 0;
  int result = 0;
  bool validOperation = true;

  // Start the operation cycle
  Serial.println("Press '@' to start a new operation.");

  // Wait for the user to press '*'
  int flag = 1;
  while (true) {
    char startKey = keypad.getKey();
    if (startKey == '@') {
      Serial.println("Starting new operation...");
```

```
      break;
    }
  }

  // Get the first number
  Serial.print("ENTER THE FIRST NUMBER: ");
  while (num1 == -1) {
    char key = keypad.getKey();
    if (key >= '0' && key <= '9') {
      num1 = key - '0'; // Convert character to integer
      Serial.println(num1);
    }
  }

  // Get the second number
  Serial.print("ENTER THE SECOND NUMBER: ");
  while (num2 == -1) {
    char key = keypad.getKey();
    if (key >= '0' && key <= '9') {
      num2 = key - '0'; // Convert character to integer
      Serial.println(num2);
    }
  }

  // Get the operation
  Serial.println("Operations: A = addition, B = subtraction,
                  C = multiplication, D = division");
  Serial.print("ENTER THE OPERATION: ");
  while (operation == 0) {
    operation = keypad.getKey();
    if (operation == '+' || operation == '-' ||
        operation == '*' || operation == '/') {
      Serial.println(operation);
    }
  }

  // Perform the operation
  if (operation == '+') {
    result = num1 + num2;
  } else if (operation == '-') {
    result = num1 - num2;
  } else if (operation == '*') {
```

```
    result = num1 * num2;
  } else if (operation == '/') {
    if (num2 != 0) {
      result = num1 / num2;
    } else {
      Serial.println("Error: Division by zero!");
      validOperation = false;
    }
  } else {
    validOperation = false;
    Serial.println("Invalid operation!");
  }

  // Output the result if the operation was valid
  if (validOperation) {
    Serial.print("Result: ");
    Serial.print(num1);
    Serial.print(" ");
    Serial.print(operation);
    Serial.print(" ");
    Serial.print(num2);
    Serial.print(" = ");
    Serial.println(result);
  }

  // Wait for the user to press '#' before starting a new operation
  Serial.println("Operation complete. Press '#' to start again.");
  while (true) {
    char endKey = keypad.getKey();
    if (endKey == '#') {
      Serial.println("Restarting...");
      break;
    }
  }
}
```

Circuit 4. Password System

This circuit demonstrates a simple password protection system using a 4x4 matrix keypad. Figure 6 shows how to connect the keypad and the LEDs to the Arduino board. The cathode of the LED is connected to GND with a 220Ω resistor to limit the current.

The system allows you to enter a password through the keypad and confirms it. Each time a

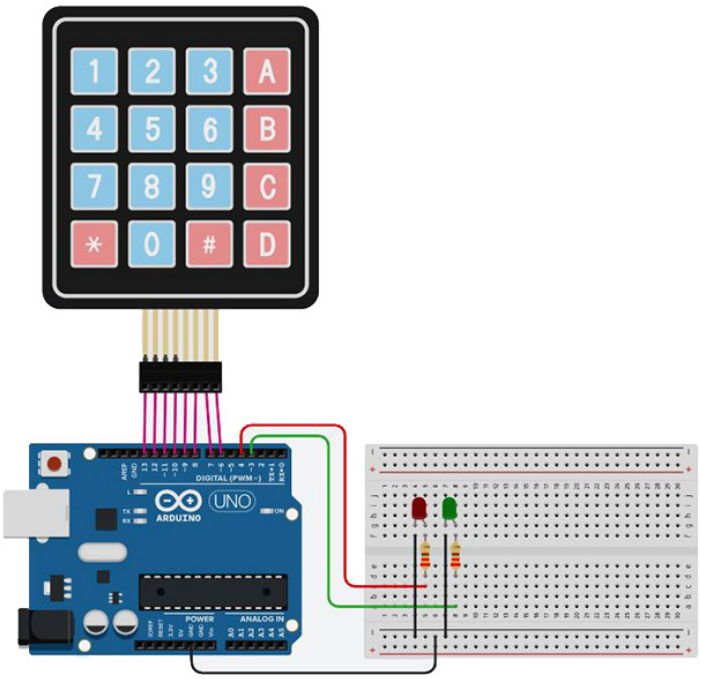

Figure 6: Keypad and LEDs connection.

key is pressed, it is appended to the string representing the entered password. When the # key is pressed, the program compares the entered password to a predefined correct password (in this case, 1234). If the correct password is entered, a green LED lights up as a visual confirmation; otherwise, the system gives feedback with a red LED to indicate an incorrect password. The system also allows resetting the entered password by pressing the * key.

```
#include <Keypad.h>

// Define the dimensions of the keypad
const byte ROWS = 4;
const byte COLS = 4;

// Keypad layout
char keys[ROWS][COLS] = {{'1', '2', '3', 'A'},
                         {'4', '5', '6', 'B'},
                         {'7', '8', '9', 'C'},
                         {'*', '0', '#', 'D'}};

// Define the pins for the keypad
byte rowPins[ROWS] = {13, 12, 11, 10};
byte colPins[COLS] = {9, 8, 7, 6};

// Create a Keypad object
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);
```

```
// Define pins
int red = 4;
int green = 3;

// Change your password here
const String password = "1234";
String input_password;

void setup() {
  Serial.begin(9600);
  pinMode(green, OUTPUT);
  pinMode(red, OUTPUT);
  input_password.reserve(32);  // maximum input characters is 33
  Serial.println("Press * to clear or # to confirm ");
  Serial.println("Insert password: ");
}

void loop() {
  char key = keypad.getKey();
  if (key) {
    Serial.println(key);
    if (key == '*') {
      input_password = "";  // clear input password
      digitalWrite(green, LOW);
      digitalWrite(red, LOW);
    } else if (key == '#') {
      if (password == input_password) {
        Serial.println("password is correct");
        digitalWrite(green, HIGH);
        digitalWrite(red, LOW);
      } else {
        Serial.println("password is incorrect, try again");
        digitalWrite(red, HIGH);
        digitalWrite(green, LOW);
      }
      input_password = "";  // clear input password
    } else {
      input_password += key;  // append new character to input password string
    }
  }
}
```