Usando o Teclado Matricial com Arduino

Introdução

O Teclado Matricial 4x4 foi desenvolvido para facilitar a entrada de dados em projetos microcontrolados. Este teclado possui 16 teclas, onde 10 delas são números, 4 são letras e 2 são caracteres especiais. Com ele podemos criar uma infinidade de projetos, tais como criar controles de acesso, teclados musicais, entre outros.

Nesse tutorial iremos montar um simples projeto, assim aprendendo o funcionamento básico do teclado, de imprimir as teclas pressionadas no monitor serial. Posteriormente iremos acionar cargas, mediante a uma senha programada.

Lista de Materiais

LISTA COMPLETA DE PRODUTOS

COMPRAR

BLACKBOARD UNO R3

R$ 94,90 no PIX

CABO USB AB 1.50M

R$ 10,35 no PIX

TECLADO MATRICIAL DE MEMBRANA 16 TECLAS

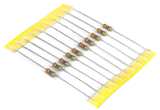
R$ 6,17 no PIX

LED 5MM VERMELHO (10 UNIDADES)

R$ 2,37 no PIX

LED 5MM VERDE (10 UNIDADES)

R$ 2,37 no PIX



RESISTOR 300Ω - PACOTE COM 10 UNIDADES

R$ 0,71 no PIX

PROTOBOARD 400 PONTOS

R$ 6,55 no PIX

WORKPLATE 400 - PRETA

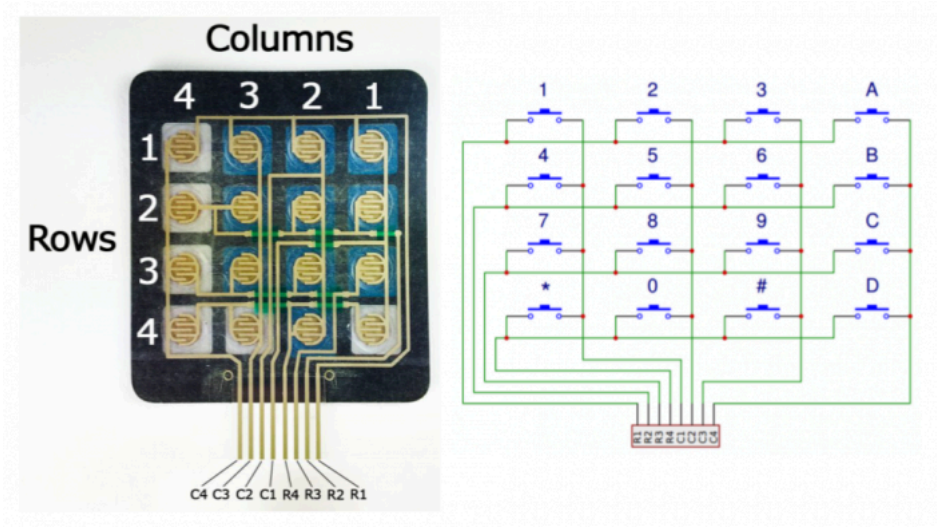
R$ 14,15 no PIX

JUMPERS MACHO-MACHO X40 UNIDADES

R$ 7,50 no PIX

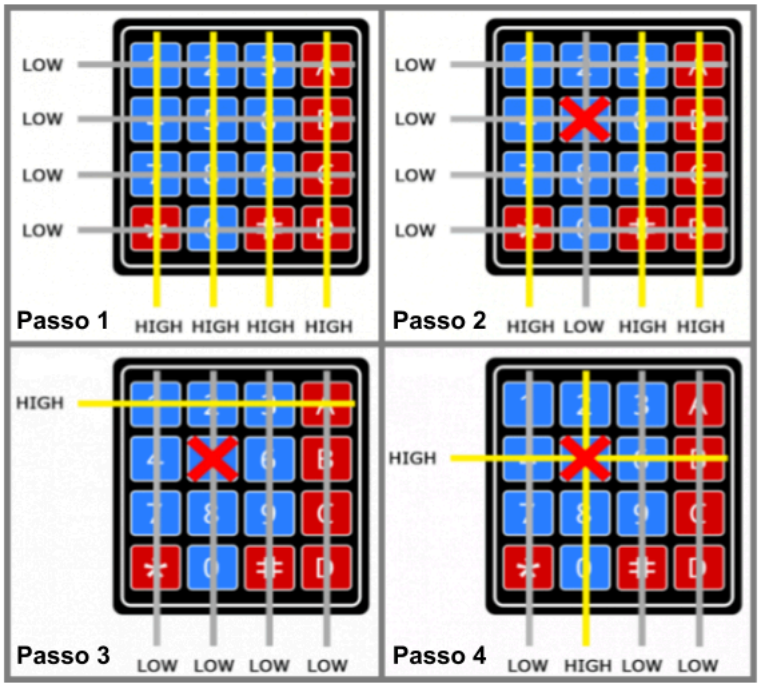
Conceitos Teóricos

O teclado possui 16 teclas, que estão dispostas em 4 linhas por 4 colunas, e ele possui 8 pinos para ligação. Embaixo de cada tecla há um interruptor de membrana. Cada interruptor em uma linha é conectado aos outros interruptores da mesma linha por um traço condutor sob o bloco, e da mesma forma são conectadas às colunas, onde todos os botões da coluna também estão conectados. Ou seja, todos os botões do teclado estão conectados a uma linha e a uma coluna, por isso que é chamado de teclado matricial. A imagem abaixo ilustra o circuito do teclado.



Esquema de Ligação dos Botões e Terminais de Conexão
Fonte: Circuit Basics

Para identificar qual botão foi pressionado, a placa Arduino executa quatro passos. O primeiro é configurar todas as colunas da matriz como entradas em nível lógico alto (resistor de *pull-up* interno ativado), e todas as linhas como saídas em nível lógico baixo. Deste modo, caso o botão "5" seja pressionado, a coluna "2" passará para o nível lógico baixo, completando o segundo passo. Já no terceiro passo, o Arduino identifica a linha que foi pressionada invertendo o nível lógico anterior, ou seja, configurando as colunas para nível lógico baixo, e mantendo as linhas em nível lógico alto. Desta forma, a linha "2", em nosso exemplo, passará para o nível lógico baixo, e o microcontrolador identificará que o botão pressionado está na linha "2" e coluna "2", completando, portanto, o quarto e último passo, como na imagem abaixo.



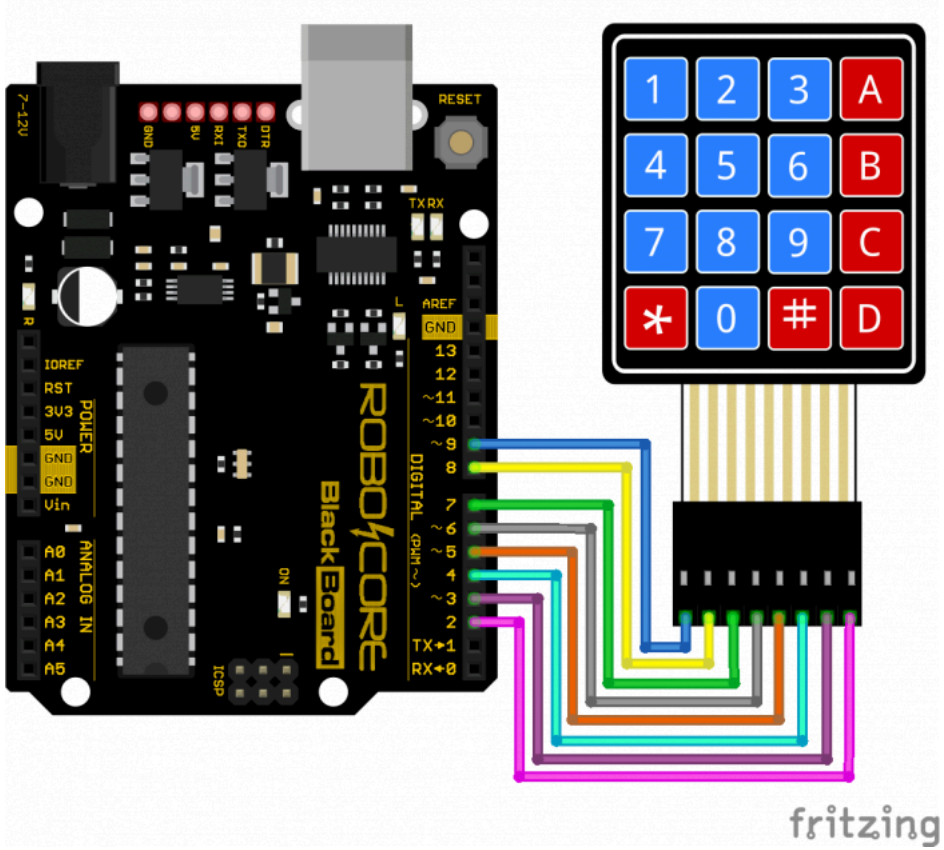
Passos para a Leitura das Teclas Pressionadas
Editado de Circuit Basics

Projeto Teclando com Arduino

Para começar a trabalhar com o teclado matricial, vamos criar um projeto simples de exibição das teclas pressionadas na tela do computador.

Circuito

Para exibir no monitor serial as teclas presionadas, monte o circuito a seguir.



Circuito Elétrico
(clique na imagem para ampliar)

Software

Biblioteca

Para que possamos começar nossos estudos sobre o teclado matricial, baixe e instale a biblioteca através do botão a seguir.

Download da Biblioteca "Keypad"

Caso você não saiba como instalar bibliotecas na Arduino IDE, siga os passos de nosso tutorial Adicionando Bibliotecas na IDE Arduino.

Código

Com a biblioteca adicionada à Arduino IDE, copie o código abaixo e carregue-o para sua BlackBoard.

```
1  /*****************************************************************************
2  * Teclado Matricial 16 Teclas : Primeiros Passos (v1.0)
3  *
```

```
 * Codigo base para exibir as teclas pressionadas no monitor serial da IDE.
 *
 * Copyright 2020 RoboCore.
 * Escrito por Matheus Cassioli (30/07/2019).
 * Atualizado por Giovanni de Castro (22/01/2020).
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version (<https://www.gnu.org/licenses/>).
 ******************************************************************************/

#include <Keypad.h> // Biblioteca do codigo

const byte LINHAS = 4; // Linhas do teclado
const byte COLUNAS = 4; // Colunas do teclado

const char TECLAS_MATRIZ[LINHAS][COLUNAS] = { // Matriz de caracteres (mapeamento do teclado)
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

const byte PINOS_LINHAS[LINHAS] = {9, 8, 7, 6}; // Pinos de conexao com as linhas do teclado
const byte PINOS_COLUNAS[COLUNAS] = {5, 4, 3, 2}; // Pinos de conexao com as colunas do teclado

Keypad teclado_personalizado = Keypad(makeKeymap(TECLAS_MATRIZ), PINOS_LINHAS, PINOS_COLUNAS, LINHAS, COLUNAS); // Inicia teclado

void setup() {
  Serial.begin(9600); // Inicia porta serial
}

void loop() {

   char leitura_teclas = teclado_personalizado.getKey(); // Atribui a variavel a leitura do teclado

  if (leitura_teclas) { // Se alguma tecla foi pressionada
    Serial.println(leitura_teclas); // Imprime a tecla pressionada na porta serial
```

```
  }

}
```

Entendendo o Código

Logo no início, após a inclusão da biblioteca "Keypad", declaramos em variáveis a quantidade de linhas e de colunas do teclado utilizado, no nosso caso, 4 linhas e 4 colunas.

```
#include <Keypad.h> // Biblioteca do codigo

const byte LINHAS = 4; // Linhas do teclado
const byte COLUNAS = 4; // Colunas do teclado
```

Também criamos a matriz `TECLAS_MATRIZ`, responsável por armazenar as informações do nosso teclado, e onde indicamos para nosso microcontrolador, através de uma matriz, qual caractere é impresso quando um determinado botão do teclado é pressionado.

```
const char TECLAS_MATRIZ[LINHAS][COLUNAS] = { // Matriz de caracteres (mapeamento do teclado)
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};
```

Abaixo definimos em vetores quais pinos são responsáveis pelas linhas e pelas colunas (respectivamente `PINOS_LINHAS` e `PINOS_COLUNAS`). Posteriormente inserimos o comando `Keypad teclado_personalizado`, para iniciar o teclado e associar nossa matriz de acordo com o pinos estabelecidos para colunas e linhas.

```
const byte PINOS_LINHAS[LINHAS] = {9, 8, 7, 6}; // Pinos de conexao com as linhas do teclado
const byte PINOS_COLUNAS[COLUNAS] = {5, 4, 3, 2}; // Pinos de conexao com as colunas do teclado

Keypad teclado_personalizado = Keypad(makeKeymap(TECLAS_MATRIZ), PINOS_LINHAS, PINOS_COLUNAS,
LINHAS, COLUNAS); // Inicia teclado
```

No setup do programa, inserimos o comando `Serial.begin(9600)`, assim iniciando a porta serial em 9600 bits por segundo.

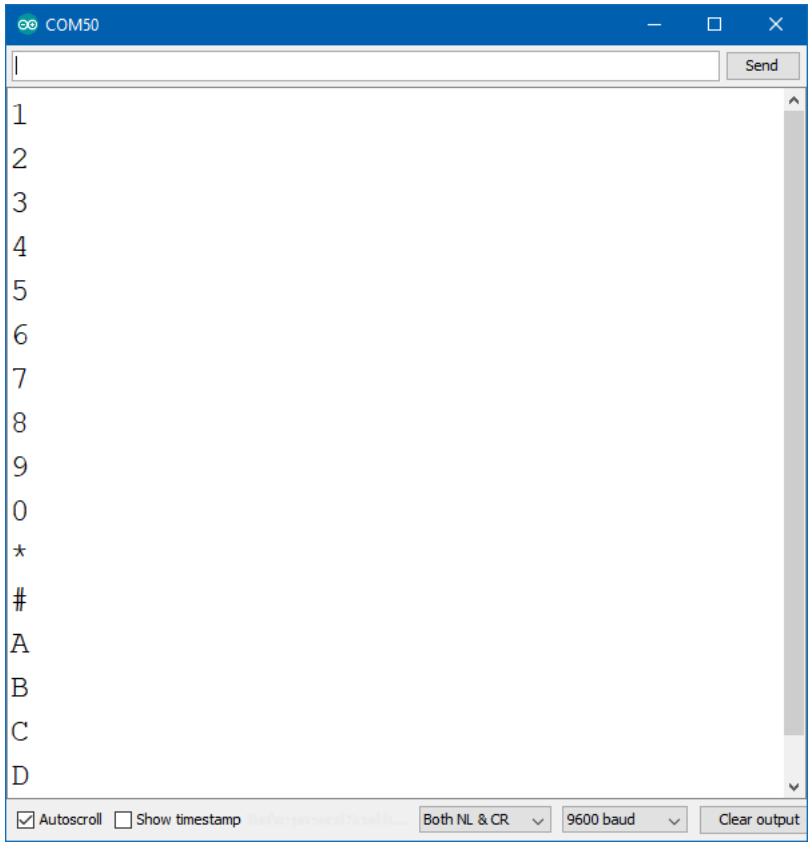
Por fim, no looping do programa, e com o comando `leitura_teclas = teclado_personalizado.getKey()`, atribuímos à variável `leitura_teclas` o caractere do botão que foi pressionado. Com essa variável, verificamos se alguma tecla foi realmente pressionada através da condição `if (leitura_teclas)`. Caso alguma tecla tenha sido realmente pressionada, o

valor da variável será diferente de "0" e a condição será verdadeira, resultando na impressão da tecla pressionada no monitor serial.

```
char leitura_teclas = teclado_personalizado.getKey(); // Atribui a variavel a leitura do teclado

if (leitura_teclas) { // Se alguma tecla foi pressionada
  Serial.println(leitura_teclas); // Imprime a tecla pressionada na porta serial
}
```

O Que Deve Acontecer

Ao abrir o Monitor Serial em 9600 bps, pressione o botão desejado e o mesmo será impresso na tela, como na figura abaixo.



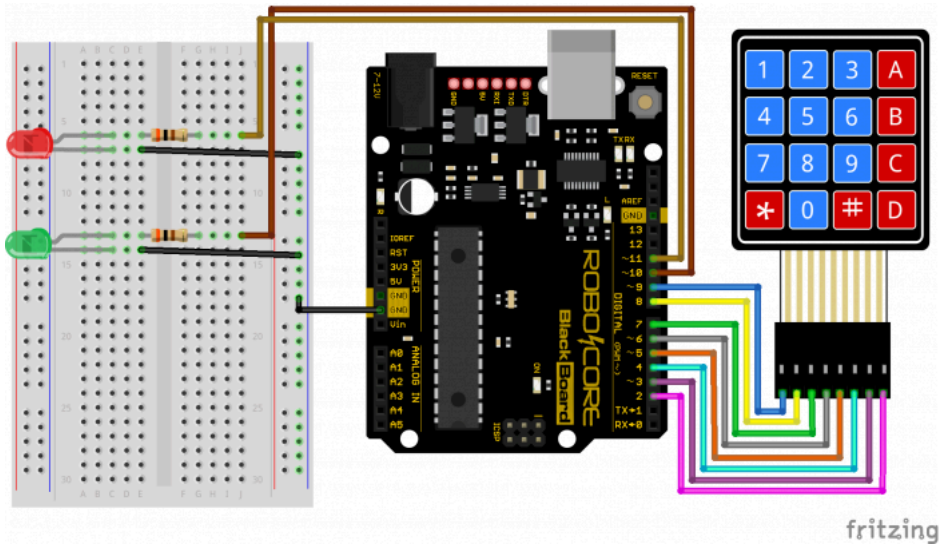
Resultado Final

Projeto Controle de Acesso

Já que aprendemos o funcionamento básico do Teclado Matricial, que tal criarmos uma senha de acesso a um cofre, ou até mesmo uma liberação de uma fechadura?

Circuito

No projeto a seguir, iremos utilizar dois LEDs, sendo eles um vermelho e um verde. Os mesmos representam se a fechadura foi ou não liberada.



Circuito Elétrico
(clique na imagem para ampliar)

Software

Biblioteca

Para que possamos criar um controle de acesso através um senha pré-definida, baixe e instale a biblioteca a seguir.

Download da Biblioteca "Password"

Caso você não saiba como instalar bibliotecas na Arduino IDE, siga os passos de nosso tutorial Adicionando Bibliotecas na IDE Arduino.

Código

Após a inclusão da biblioteca, copie o código abaixo e passe-o para sua placa. Esse código foi baseado no projeto acima, porém incluindo uma nova biblioteca e adicionando alguns comandos.

```
/*****************************************************************************
* Teclado Matricial 16 Teclas : Controle de Acesso (v1.0)
*
```

```
 * O codigo ira verificar se a senha digitada esta correta. Caso correta, o
 * acesso e liberado, caso contrario o acesso se mantem travado.
 *
 * Copyright 2020 RoboCore.
 * Escrito por Matheus Cassioli (30/07/2019).
 * Atualizado por Giovanni de Castro (22/01/2020).
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version (<https://www.gnu.org/licenses/>).
 ************************************************************************/

#include <Password.h> // Biblioteca utilizada para controle de senha
#include <Keypad.h> // Biblioteca para controle do teclado de matrizes

const byte LINHAS = 4; // Linhas do teclado
const byte COLUNAS = 4; // Colunas do teclado

Password senha = Password( "8765" ); // Senha utilizada para liberacao

const int PINO_LED_VERMELHO = 11; // LED vermelho conectado ao pino 11
const int PINO_LED_VERDE = 10; // LED verde conectado ao pino 10

const char TECLAS_MATRIZ[LINHAS][COLUNAS] = { // Matriz de caracteres (mapeamento do teclado)
  {'1', '2', '3', 'A'},
  {'4', '5', '6', 'B'},
  {'7', '8', '9', 'C'},
  {'*', '0', '#', 'D'}
};

const byte PINOS_LINHAS[LINHAS] = {9, 8, 7, 6}; // Pinos de conexao com as linhas do teclado
const byte PINOS_COLUNAS[COLUNAS] = {5, 4, 3, 2}; // Pinos de conexao com as colunas do teclado

Keypad teclado_personalizado = Keypad(makeKeymap(TECLAS_MATRIZ), PINOS_LINHAS, PINOS_COLUNAS, LINHAS, COLUNAS); // Inicia teclado

void setup() {

  Serial.begin(9600); // Inicializa serial monitor

```

```
  pinMode(PINO_LED_VERMELHO, OUTPUT); // Define pino 10 como saida
  pinMode(PINO_LED_VERDE, OUTPUT); // Define pino 11 como saida

  digitalWrite(PINO_LED_VERDE, LOW); // LED Verde apagado
  digitalWrite(PINO_LED_VERMELHO, LOW); // LED Vermelho apagado

}

void loop() {

   char leitura_teclas = teclado_personalizado.getKey(); // Atribui a variavel a leitura do teclado

  if(leitura_teclas){ // Se alguma tecla foi pressionada

    if(leitura_teclas == 'C'){ // Se a tecla 'C' foi pressionada

      if(senha.evaluate()){ // Verifica se a senha digitada esta correta

        Serial.println("Senha confirmada!"); // Exibe a mensagem que a senha esta correta
        for(int i = 0; i < 5; i++){ // Pisca o LED 5 vezes rapidamente
          digitalWrite(PINO_LED_VERDE, HIGH);
          delay(50);
          digitalWrite(PINO_LED_VERDE, LOW);
          delay(50);

        }
      } else { // Caso a senha esteja incorreta

        Serial.println("Senha incorreta!"); // Exibe a mensagem que a senha esta errada
        for(int i = 0; i < 5; i++){ // Pisca o LED 5 vezes rapidamente
          digitalWrite(PINO_LED_VERMELHO, HIGH);
          delay(50);
          digitalWrite(PINO_LED_VERMELHO, LOW);
          delay(50);

        }
      }

      senha.reset(); // Limpa a variavel senha

    } else { // Caso outra tecla tenha sido pressionada
```

```
85
86        Serial.println(leitura_teclas); // Exibe a tecla pressionada
87        senha.append(leitura_teclas); // Salva o valor da tecla pressionada na variavel senha
88
89      }
90    }
91
92  }
```

Entendendo o Código

Primeiramente, com a ajuda da biblioteca "Password", criamos a variável `senha` e indicamos os caracteres que irão fazer parte dela, sendo números, letras ou caracteres especiais. Além disso, declaramos as variáveis que armazenam os pinos onde os LEDs do circuito estão conectados.

```
1  #include <Password.h> // Biblioteca utilizada para controle de senha
2
3  Password senha = Password( "8765" ); // Senha utilizada para liberacao
4
5  const int PINO_LED_VERMELHO = 11; // LED vermelho conectado ao pino 11
6  const int PINO_LED_VERDE = 10; // LED verde conectado ao pino 10
```

No setup do programa, configuramos os pinos dos LEDs (`PINO_LED_VERDE` e `PINO_LED_VERMELHO`) como saídas, em nível lógico baixo.

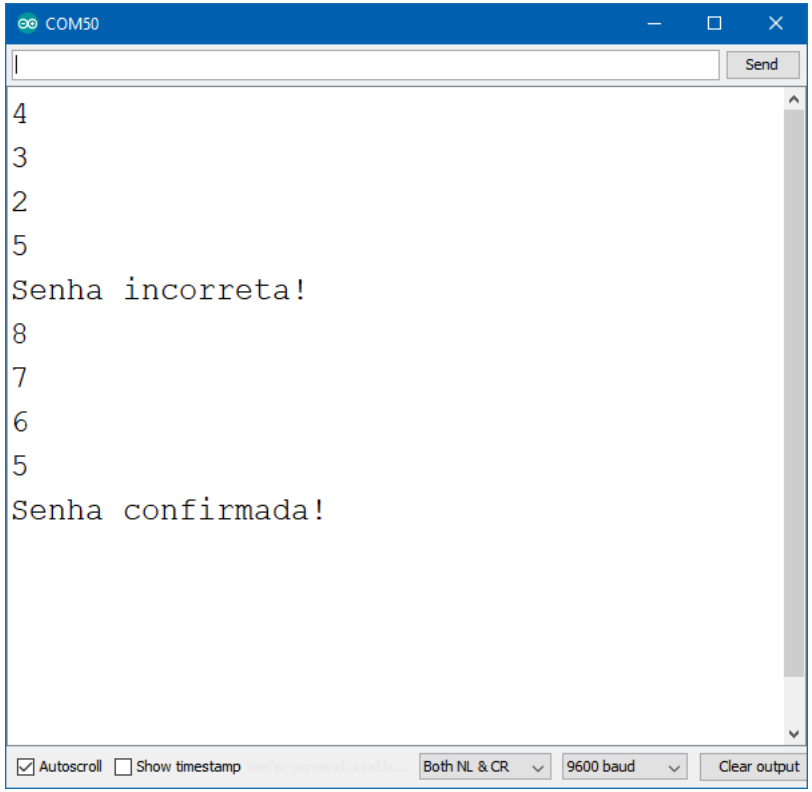
```
1  pinMode(PINO_LED_VERMELHO, OUTPUT); // Define pino 10 como saida
2  pinMode(PINO_LED_VERDE, OUTPUT); // Define pino 11 como saida
3
4  digitalWrite(PINO_LED_VERDE, LOW); // LED Verde apagado
5  digitalWrite(PINO_LED_VERMELHO, LOW); // LED Vermelho apagado
```

No looping, iniciamos a repetição atribuindo à variável `leitura_teclas` a leitura dos botões, como no código anterior. Entretanto, neste código, após verificarmos se alguma tecla foi realmente pressionada, verificamos, através da condição `if(leitura_teclas == 'C')`, se a tecla "C", responsável pela confirmação da senha, foi pressionada. Caso ela tenha sido pressionada, damos continuidade para a verificação da senha digitada através da condição `if(senha.evaluate())`. Caso a senha digitada esteja correta, piscamos o LED verde e exibimos a mensagem que a senha foi confirmada. Caso contrário, por meio do comando `else`, piscamos o LED vermelho e exibimos a mensagem que a senha está incorreta. Logo após a verificação da senha, graças ao comando `senha.reset()`, limpamos a variável `senha` para que ela possa ser digitada novamente. Entretanto, caso a tecla pressionada não seja a tecla "C", a tecla que foi pressionada é impressa no monitor, e acrescentada à variável `senha` através do comando `senha.append(leitura_teclas)`.

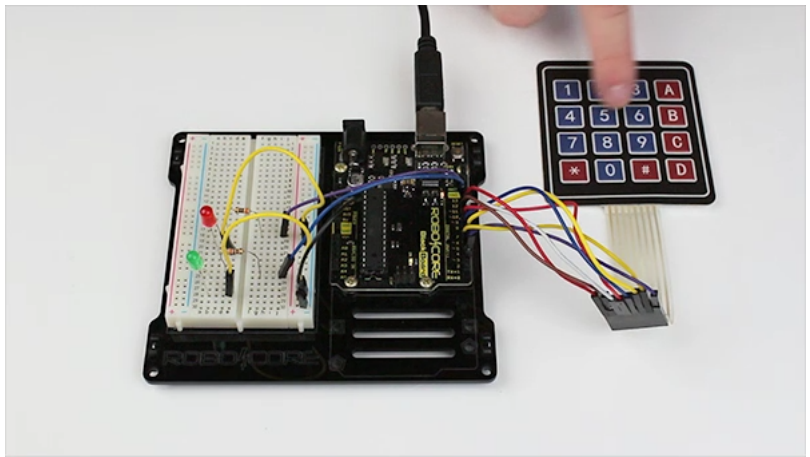
```
char leitura_teclas = teclado_personalizado.getKey(); // Atribui a variavel a leitura do teclado

if(leitura_teclas){ // Se alguma tecla foi pressionada

  if(leitura_teclas == 'C'){ // Se a tecla 'C' foi pressionada

    if(senha.evaluate()){ // Verifica se a senha digitada esta correta

      Serial.println("Senha confirmada!"); // Exibe a mensagem que a senha esta correta
      for(int i = 0; i < 5; i++){ // Pisca o LED 5 vezes rapidamente
        digitalWrite(PINO_LED_VERDE, HIGH);
        delay(50);
        digitalWrite(PINO_LED_VERDE, LOW);
        delay(50);
      }

    } else { // Caso a senha esteja incorreta

      Serial.println("Senha incorreta!"); // Exibe a mensagem que a senha esta errada
      for(int i = 0; i < 5; i++){ // Pisca o LED 5 vezes rapidamente
        digitalWrite(PINO_LED_VERMELHO, HIGH);
        delay(50);
        digitalWrite(PINO_LED_VERMELHO, LOW);
        delay(50);
      }

    }

    senha.reset(); // Limpa a variavel senha

  } else { // Caso outra tecla tenha sido pressionada

    Serial.println(leitura_teclas); // Exibe a tecla pressionada
    senha.append(leitura_teclas); // Salva o valor da tecla pressionada na variavel senha

  }
}
```

O Que Deve Acontecer

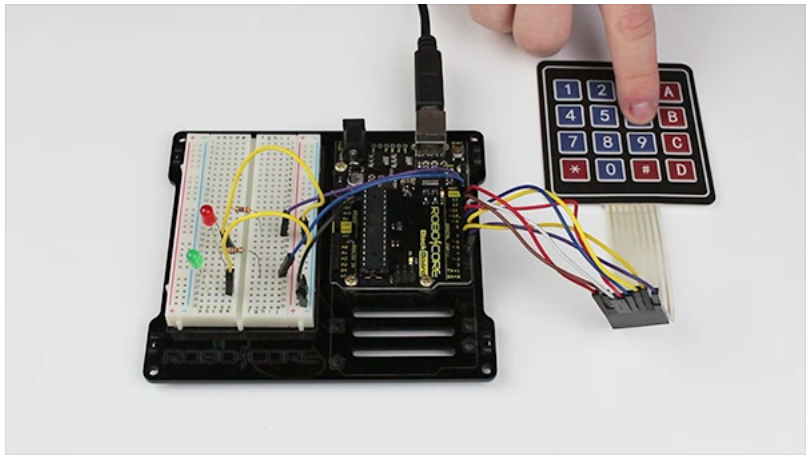
Ao abrir o Monitor Serial em 9600 bps, digite a senha programada anteriormente e confirme pressionando a tecla "**C**". Note que o LED verde irá piscar rapidamente 5 vezes. Você também pode digitar uma senha incorreta para verificar que o LED vermelho irá piscar rapidamente 5 vezes, como nos GIFs abaixo.



Resultado Final



Resultado Final - Senha Incorreta



Resultado Final - Senha Correta

Indo Além

Embora utilizamos a biblioteca `Keypad.h` neste tutorial, é possível realizar a leitura da tecla pressionada sem a utilização dos comandos da biblioteca. Para isso, basta você seguir os passos de leitura mencionados na seção Conceitos Teóricos, e verificar a linha e a coluna que estão interligados no botão pressionado.

por Giovanni de Castro

"Graduando de Automação Industrial pelo IFSP, terminando um TCC nas horas vagas."

e Matheus Cassioli

"Ao infinito e além"