

簡易電子ロック

概要

最近では、防犯に関する様々な製品が販売されています。 防犯カメラ、防犯センサー、防犯アラーム、防犯器具、防犯灯、防盜金庫、防犯フィルム。。。。

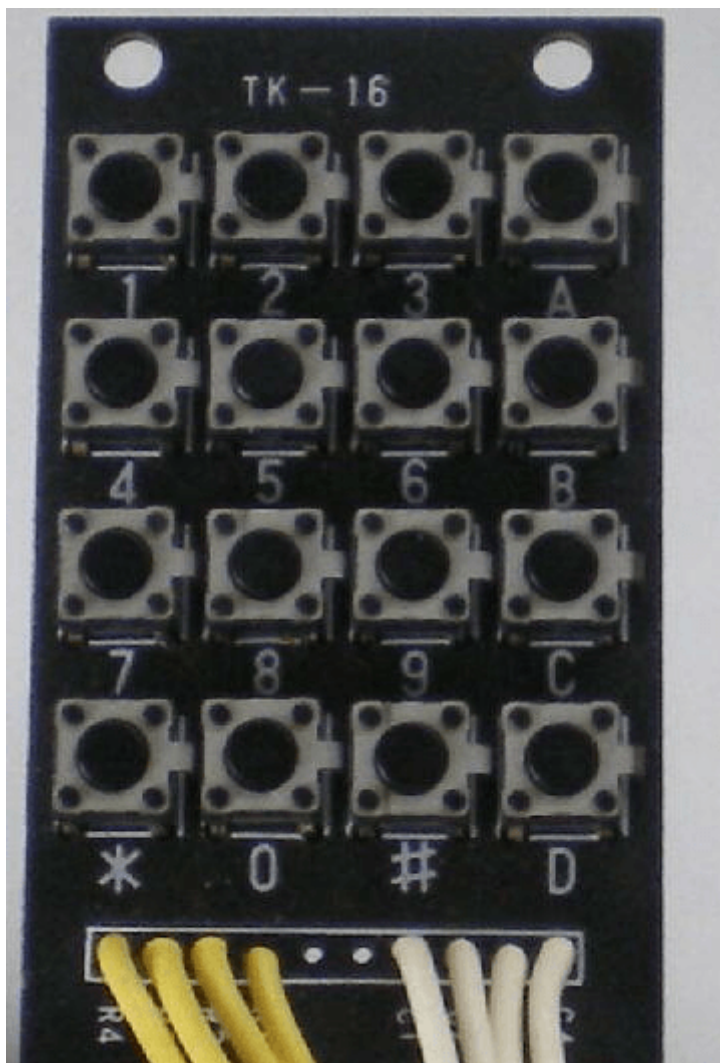
そこで手持ちのキーパッド(4×4)を活かして、簡易な電子ロックを製作してみました。

<仕様>

- 暗証番号は、4桁の数字とします。
暗証番号は、PIC内蔵のEEPROMに記録されます。
初期状態の時の暗証番号は、“ 0123 ” です。
- 暗証番号を入力する都度に、【アンロック状態】 【ロック状態】を繰り返します。
4桁の数字の後に、“ * ” を入力します。
- 暗証番号の変更を可能とします
4桁の数字の後に、“ # ” を入力します。
変更は、【アンロック状態】の時のみ可能です。
- 出力電圧
【アンロック状態】の時に5Vが出力されます。
【ロック状態】の時に0Vが出力されます。
この出力を利用して、出力装置をON/OFFします。
- キー押下や各処理に応じて、ブザー音を鳴らせます。

動作原理

キー入力には、4×4のキーパッド(keypad)を使用しました。プッシュスイッチを16個並べて自作するこ



とも出来ます。

<暗証番号の書き込み処理>

- 暗証番号は、PIC内蔵のEEPROMに書き込みます。
- 暗証番号の4桁を、最初の4バイトと、次の4バイトに、2度書します。

<暗証番号の読み出し処理>

- 初期状態ではEEPROMの内容は不定な値となっています。
- 暗証番号の4桁を、最初の4バイトと、次の4バイトから、2度読みします。
- 暗証番号の4桁が全て、数値であることをチェックします。
- 読み込んだ、最初の4バイトと、次の4バイトが一致するかをチェックします。
- これらのチェックに失敗すると、初期状態とみなし、暗証番号を“0123”とします。

<キーボードからのキー取得処理>

- mikroC提供のライブラリ“Keypad4x4”を使用します。
- チャタリング防止のために、押下されたキーが連続(100msec間に10回)することをチェックします。

<暗証番号の確認処理>

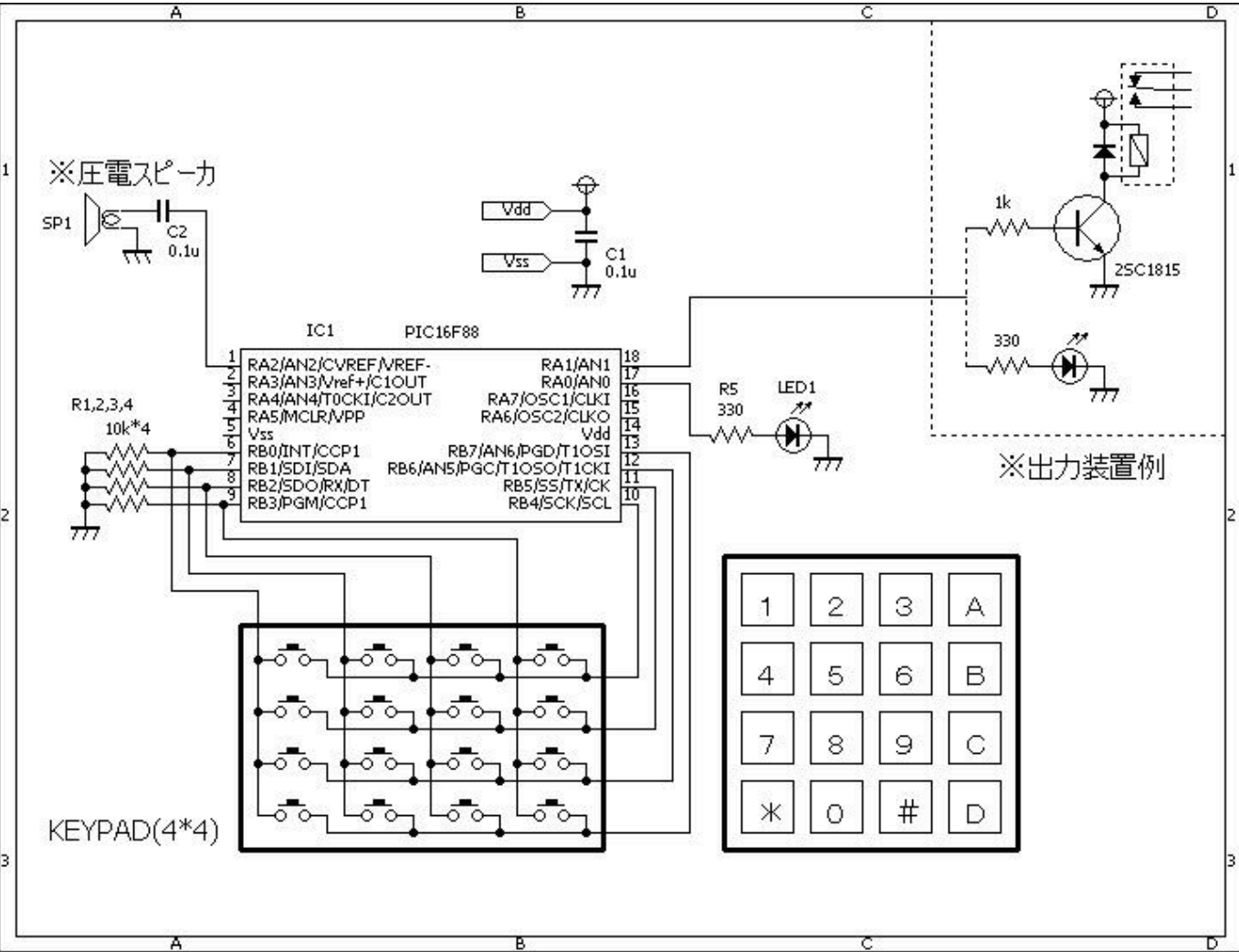
- “*”キーが押下されると、暗証番号をチェックします。
- 一致すると、【アンロック状態】であれば【ロック状態】にします。【ロック状態】であれば【アンロック状態】にします。

- 【アンロック状態】の時に5Vが出力されます。【ロック状態】の時に0Vが出力されます。

<暗証番号の変更処理>

- “ # “ キーが押下されると、【アンロック状態】であれば暗証番号を変更します。
- 【ロック状態】の時には、暗証番号は変更されません。

回路図



ソースコード

[electronic_lock.c](#)

```
//*****
*
/*
< 簡易電子ロック >
*/
//*****
*
```

```
// ■ ■ ■ 関数宣言 ■ ■ ■
extern void main();
extern void init_keypad();
extern char get_keypad();
extern void get_password(char *password);
extern void set_password(char *password);
extern void buzzer(unsigned freq, unsigned duration, unsigned
cnt);
//*****
*

// ■ ■ ■ マクロ定義 ■ ■ ■
//
sbit LED at RA0_bit;
sbit KEY at RA1_bit;
//other
#define OFF 0
#define ON 1
#define UNLOCK 1
#define LOCK 0
//*****
*

// ■ ■ ■ メイン関数 ■ ■ ■
void main()
{
    char password[10], temp[10], c, cnt;
    short sts;
    //
    OSCCON = 0b01110000; //クロックを8MHzに設定します。
    ANSEL = 0b00000000; //A/D変換モジュールは使用しません。
    TRISA = 0b00100000;
    //
    LED = OFF;
    KEY = OFF;
    init_keypad();
    Sound_Init(&PORTA, 2);
    //EEPROMに保存している暗証番号を取得します。
    get_password(password);
    buzzer(1000, 100, 5);
    //
    cnt = 0;
    strcpy(temp, "????");
    sts = LOCK;
    //
    while (1) {
        c = get_keypad();
        switch (c) {
            case '#': //EEPROMに新しい暗証番号を保存します。
                if (sts == UNLOCK) {
                    strcpy(password, temp);
                    set_password(password);
                    buzzer(2000, 100, 3);
                }
            }
        }
    }
```

```

        } else {
            buzzer(500, 100, 3);
        }
        strcpy(temp, "????");
        break;
    case '*': //暗証番号が一致しているかを確認します。
        if (strncmp(password, temp, 4) == 0) {
            if (sts == LOCK) {
                sts = UNLOCK;
                KEY = UNLOCK;
                buzzer(2000, 1000, 1);
            } else {
                sts = LOCK;
                KEY = LOCK;
                buzzer(2000, 1000, 1);
            }
        } else {
            buzzer(500, 1000, 1);
        }
        strcpy(temp, "????");
        break;
    case 'A':
    case 'B':
    case 'C':
    case 'D':
        buzzer(500, 100, 1);
        break;
    default: // 0,1,2,3,4,5,6,7,8,9
        temp[0] = temp[1];
        temp[1] = temp[2];
        temp[2] = temp[3];
        temp[3] = c;
        buzzer(1000, 100, 1);
        break;
    }
}

//*****
*
// ■ ■ ■ キーパッド初期化関数 ■ ■ ■
char keypadPort at PORTB;
void init_keypad()
{
    Keypad_Init();
}
//*****
*
// ■ ■ ■ キー取得関数 ■ ■ ■
char kp_tbl[16] =
{'1', '2', '3', 'A', '4', '5', '6', 'B', '7', '8', '9', 'C', '*', '0', '#', 'D'};
char get_keypad()

```

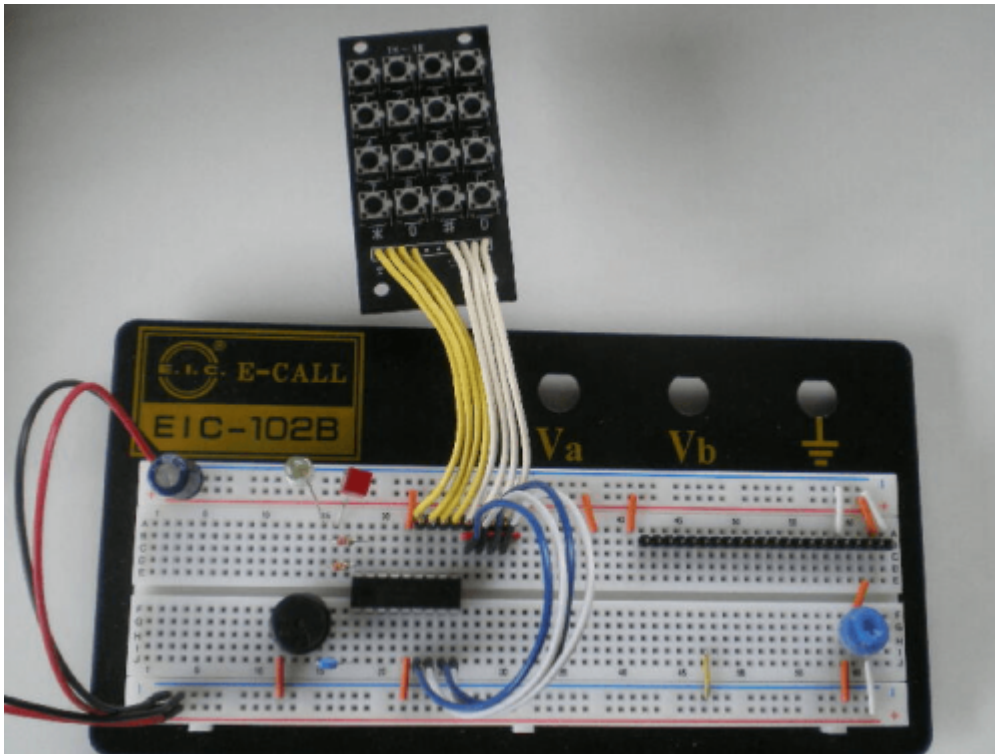
```
{
    char    kp, kp_old, cnt;
    //
    kp_old = 0;
    while (1) {
        for (cnt = 0; cnt < 10; cnt++) {
            kp = Keypad_Key_Press();
            if ((kp != kp_old) || (kp == 0)) {
                cnt = 0;
            }
            kp_old = kp;
            Delay_ms(10);
        }
        for (cnt = 0; cnt < 10; cnt++) {
            if (Keypad_Key_Press() != 0) {
                cnt = 0;
            }
            Delay_ms(10);
        }
        return (kp_tbl[kp - 1]);
    }
}

//*****
*
//■■■暗証番号読み込み関数■■■
void    get_password(char *password)
{
    char    p0, p1, p2, p3, p4, p5, p6, p7;
    //
    p0 = EEPROM_Read(0);
    p1 = EEPROM_Read(1);
    p2 = EEPROM_Read(2);
    p3 = EEPROM_Read(3);
    p4 = EEPROM_Read(4);
    p5 = EEPROM_Read(5);
    p6 = EEPROM_Read(6);
    p7 = EEPROM_Read(7);
    password[0] = '0';
    password[1] = '1';
    password[2] = '2';
    password[3] = '3';
    password[4] = 0x00;
    if ((isdigit(p0) != 1) || (isdigit(p1) != 1) || (isdigit(p2) !=
1) || (isdigit(p3) != 1)) {
        return;
    }
    if ((p0 != p4) || (p1 != p5) || (p2 != p6) || (p3 != p7)) {
        return;
    }
    password[0] = p0;
    password[1] = p1;
```

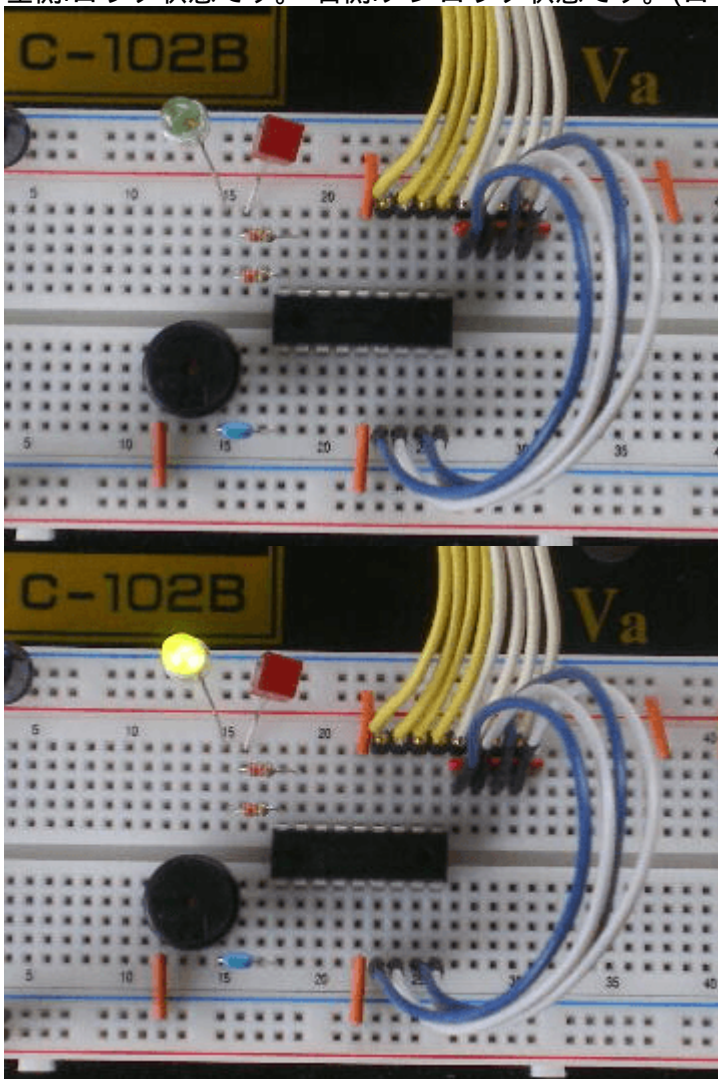
```
        password[2] = p2;
        password[3] = p3;
    }
//*****
*
//■■■暗証番号書き込み関数■■■
void    set_password(char *password)
{
    EEPROM_Write(0, password[0]);
    EEPROM_Write(1, password[1]);
    EEPROM_Write(2, password[2]);
    EEPROM_Write(3, password[3]);
    EEPROM_Write(4, password[0]);
    EEPROM_Write(5, password[1]);
    EEPROM_Write(6, password[2]);
    EEPROM_Write(7, password[3]);
}
//*****
*
//■■■ブザー関数■■■
void    buzzer(unsigned freq, unsigned duration, unsigned cnt)
{
    while (cnt > 0) {
        LED = ON;
        Sound_Play(freq, duration);
        LED = OFF;
        Delay_ms(100);
        cnt--;
    }
}
//*****
*
```

動作確認

出力装置にはLEDを使用しています。初期状態の暗証番号は、“0123”になっています。アンロック状態にしてから、暗証番号を変更してください。



左側:ロック状態です。 右側:アンロック状態です。(ロック解除)



な用途に使えるそうですね。{😊}!

如何ですか? 出力装置を工夫すれば、いろいろ

暗証番号を忘れると、ロックを解除することが出来ませんので、ご注意ください。

From:

<http://deepsky.jp/wiki/> - うごくといいな

Permanent link:

<http://deepsky.jp/wiki/doku.php?id=elechobby:picdic:pic16f88:138&rev=1588223119>

Last update: **2025/10/17 14:27**

