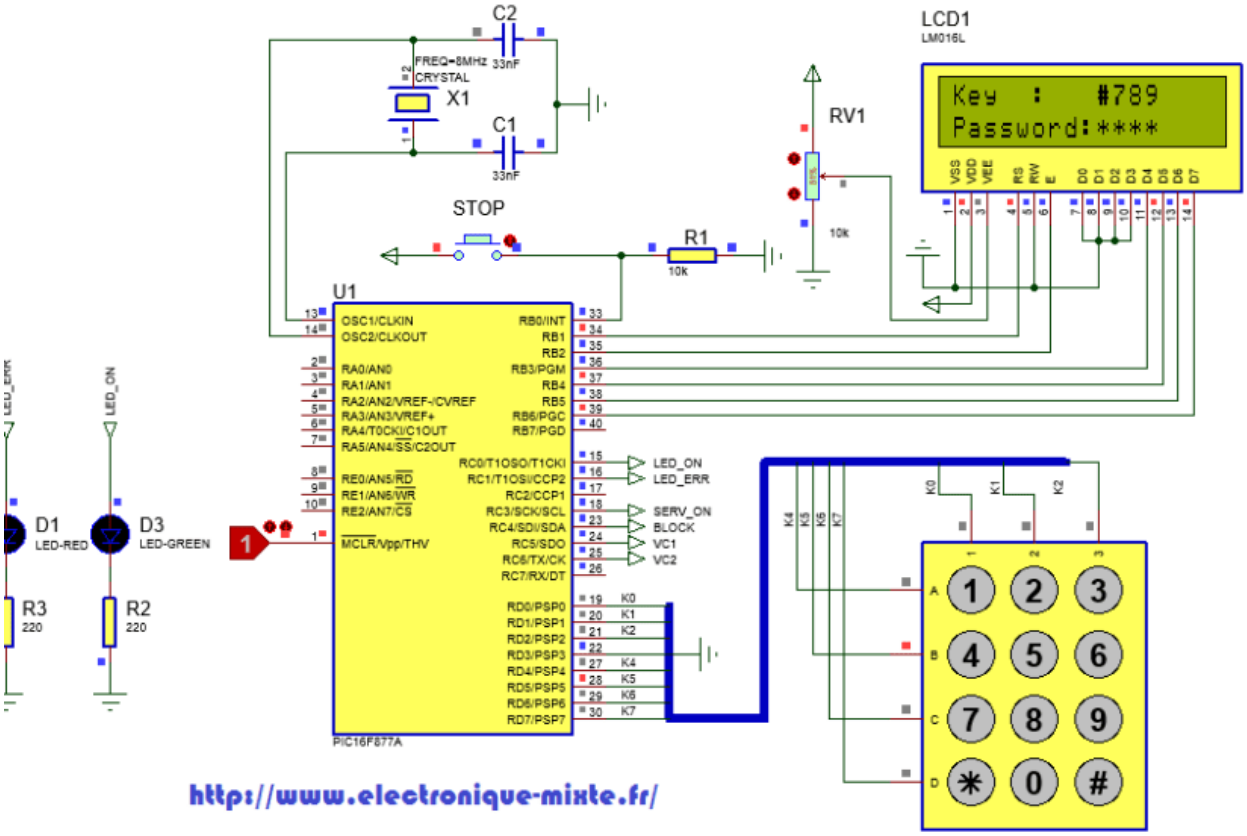
> Découvrez notre Chaîne YouTube "Ingénierie et Projets"

> Découvrez notre Chaîne Secondaire "Information Neuronale et l'Ingénierie du Cerveau"



Objectifs du projet électronique

1. Savoir comment utiliser le clavier 3×4 en utilisant des fonctions très simple sur MikroC
2. Savoir générer des tonalités différentes pour créer des mélodies avec des fonctions déjà

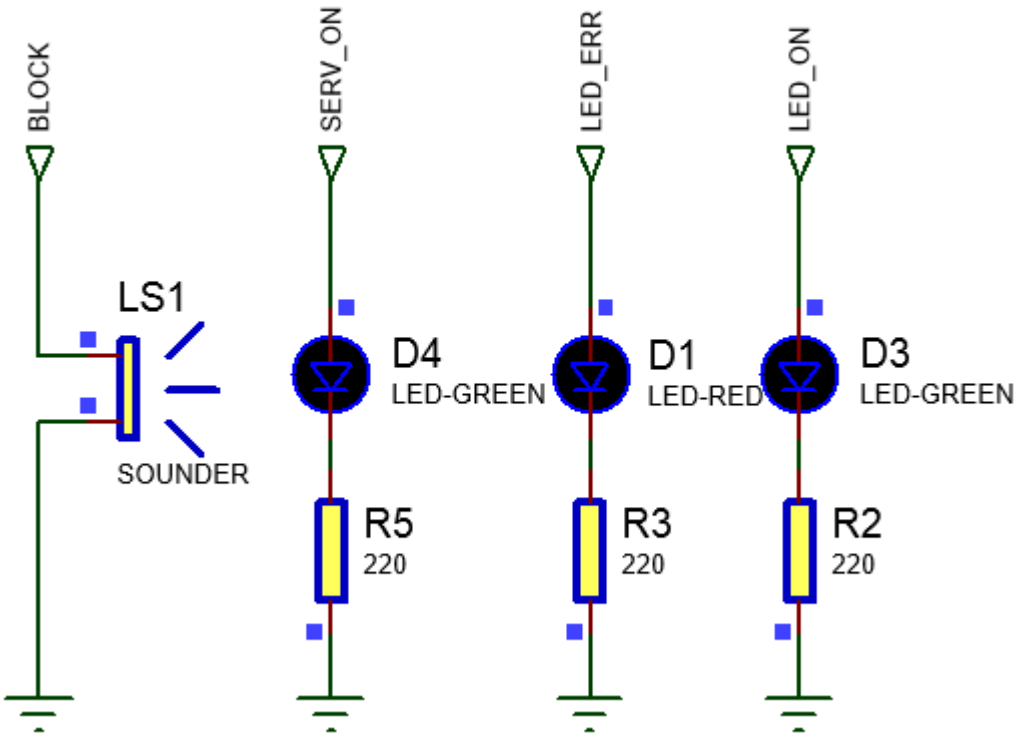
existantes
3. Utilisation du pont H complet pour commencer le sens de rotation du moteur à CC
4. Autres astuces de programmations

Principe de fonctionnement

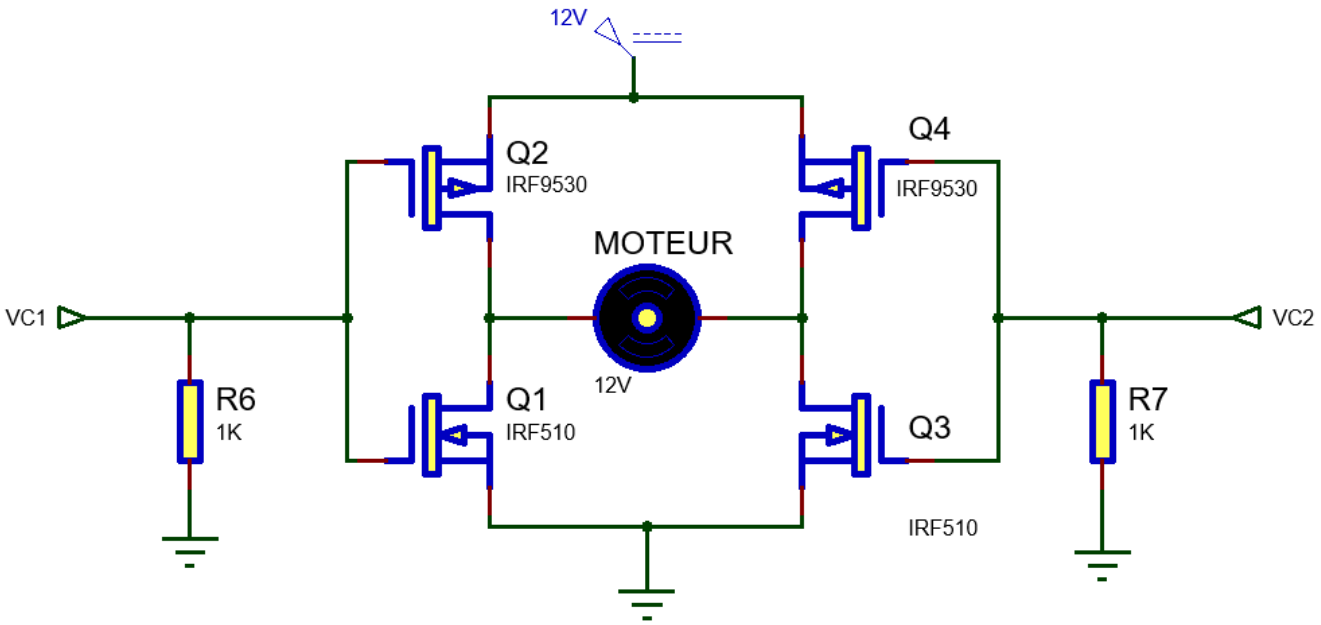
Le projet est constitué d'un clavier 4×3, un afficheur LCD 16×2, des LED d'état, un speaker et un bouton poussoir.

1. Clavier : Permet d'ecrire le mot de passe, la taille du mot est variable en fonction égal à la valeur du paramètre LengthPassWord, par défaut LengthPassWord=5. Le clavier aussi permet de réactiver le clavier dans le cas de dépassement de nombre de tentatives permises. Le caractère de déblocage = Enable_PW.
2. Afficheur LCD : Affichage des caractères instruit par le clavier et l'état du mot de passe Erreur' ou 'Correct'
3. LED d'état : 3 LED d'état : Erreur, Mot de passe correct et ouverture de la porte
4. Speaker : Alarme en cas de dépassement de nombre des tentatives permises
5. Bouton poussoir : Arrêt d'alarme. Le bouton permet juste de bloquer l'alarme, en cas d'appuie sur l'un des boutons du clavier, l'alarme se met en marche de nouveau. Pour débloquer l'alarme il faut taper le caractère de réactivation du clavier.

Dans le cas d'introduction d'un mot de passe correct, le moteur tourne dans le sens 1 pendant 5 secondes, freiner pendant 3 secondes puis tourner dans le sens 2 pendant 5 secondes. Les temps (frein, rotation sens1/2) sont réglables par les paramètres suivant dans le programme: Temps_ms_sens1, Temps_ms_sens2 et Temps_ms_door.



Circuit de puissance de commande du sens de la rotation du moteur



1. Sens 1 : VC1= ‘1’, CV2=’0′
2. Sens 2 : VC1= ‘0’, CV2=’1′
3. Frein : VC1= ‘0’, CV2=’0′ ou VC1= ‘1’, CV2=’1′

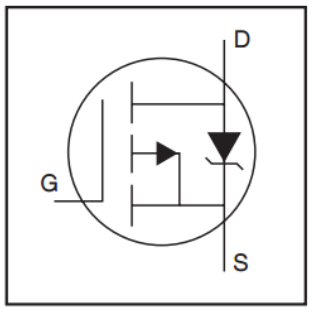
La courant maximal supporté par le transistor doit être supérieur au courant maximal demandé par la charge(le moteur). La puissance maximale supportée par le transistor de commutation IRF9530 égale à 1.4Kw(100V,14A).

International
IOR Rectifier

IRF9530NPbF

HEXFET® Power MOSFET

- Advanced Process Technology
- Dynamic dv/dt Rating
- 175°C Operating Temperature
- Fast Switching
- P-Channel
- Fully Avalanche Rated
- Lead-Free



V_{DSS} = -100V

$R_{DS(on)}$ = 0.20Ω

I_D = -14A

Notions de la musique sur le micro (Sonnerie d'Alarme & sonnerie d'ouverture de la porte)

Je ne suis pas un expert de la musique mais je peut vous aider pour développer votre propre piano et créer des mélodie à base du microcontrôleur.

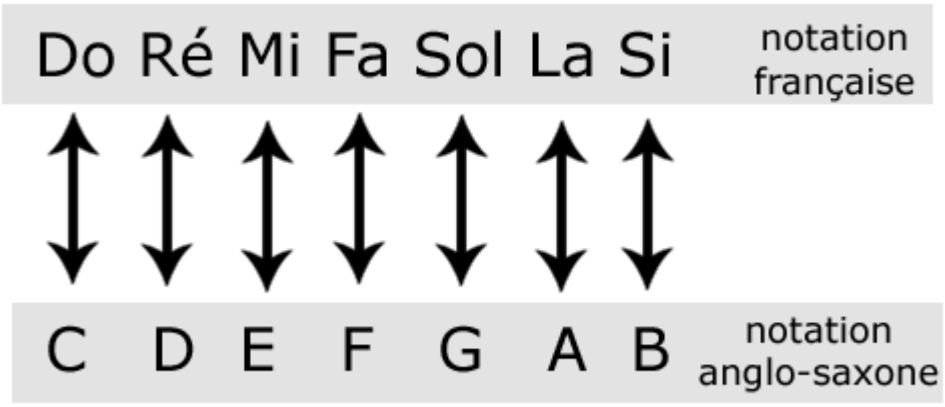
Tout son musical (ou note) possède une fréquence fondamentale (nombre de vibrations par seconde calculé en hertz) correspondant à sa hauteur. Deux notes dont les fréquences fondamentales ont un rapport qui est une puissance de deux (c'est-à-dire la moitié, le double, le quadruple...) donnent deux sons très similaires et portent le même nom. Cette observation permet de regrouper toutes les notes qui ont cette propriété dans la même catégorie de hauteur.

Dans la musique occidentale, les catégories de hauteurs sont au nombre de douze. Sept d'entre elles sont considérées comme les principales et ont pour noms : do, ré, mi, fa, sol, la et si. L'intervalle compris entre deux hauteurs dont la fréquence de l'une vaut le double (ou

la moitié) de l'autre s'appelle une octave. Pour distinguer deux notes de même nom dans deux octaves différentes, on numérote les octaves et donne ce numéro aux notes correspondantes : par exemple, le la3 a une fréquence de 440 hertz dans la norme internationale. Cette fréquence de référence est donnée par un diapason.

Fréquences des notes (en hertz) dans la gamme tempérée

Note/octave	0	1	2	3	4	5	6	7
do ou *si*♯	32,70	65,41	130,81	261,63	523,25	1046,50	2093,00	4186,01
do♯ ou *ré*♭	34,65	69,30	138,59	277,18	554,37	1108,73	2217,46	4434,92
ré	36,71	73,42	146,83	293,66	587,33	1174,66	2349,32	4698,64
ré♯ ou *mi*♭	38,89	77,78	155,56	311,13	622,25	1244,51	2489,02	4978,03
mi ou *fa*♭	41,20	82,41	164,81	329,63	659,26	1318,51	2637,02	5274,04
fa ou *mi*♯	43,65	87,31	174,61	349,23	698,46	1396,91	2793,83	5587,65
fa♯ ou *sol*♭	46,25	92,50	185,00	369,99	739,99	1479,98	2959,96	5919,91
sol	49,00	98,00	196,00	392,00	783,99	1567,98	3135,96	6271,93
sol♯ ou *la*♭	51,91	103,83	207,65	415,30	830,61	1661,22	3322,44	6644,88
la	55,00	110,00	220,00	440,00	880,00	1760,00	3520,00	7040,00
la♯ ou *si*♭	58,27	116,54	233,08	466,16	932,33	1864,66	3729,31	7458,62
si ou *do*♭	61,74	123,47	246,94	493,88	987,77	1975,53	3951,07	7902,13



Exemple d’application en format Anglo-saxone :

Happy birthday to you
C4 C4 D4 C4 F4 E4

Happy birthday to you
C4 C4 D4 C4 G4 F4

A vous de faire la conversion des notes en fréquence

Maintenant on va passer à l’implémentation sur le micro des notes. MikroC dispose d’une fonction spéciale qui permet de générer un signal carré avec une fréquence donnée pendant un durée précis. La fonction s’appelle Sound_play(Freq_Value, Duree_ms).

Exemple :

- Note C4 pendant 1s : Sound_play(523,1000)
- Note F4 pendant 400ms : Sound_play(698, 400)

> *Note : Avant d’utiliser la fonction Sound_play, il faut affecter un pin et un port pour sortir le son grâce à la fonction Sound_Init(&PORT, Num_Pin) [Sound_init(&PORTD,4)]*

Pour Plus des infos sur la limitation en fréquence et des exemples d’utilisation, vous pouvez consulter le lien suivant.

Programme MikroC

```
#define                 LengthPassWord  5       // Nombre des caractère du mot de pa
sse
#define                 NumCodeRepeat   2       // Nombre de tenation
#define                 Enable_PW       8       // Caractère de réactivation du clav
ier

#define                 Temps_ms_sens1  5000    // Temps de rotation dans le sens +
#define                 Temps_ms_sens2  5000    // Temps de rotation dans le sens -
#define                 Temps_ms_door   3000    // Temps d'ouvertur de la porte


// LCD module connections
sbit LCD_RS at RB1_bit;
sbit LCD_EN at RB2_bit;
sbit LCD_D4 at RB3_bit;
sbit LCD_D5 at RB4_bit;
sbit LCD_D6 at RB5_bit;
sbit LCD_D7 at RB6_bit;

sbit LCD_RS_Direction at TRISB1_bit;
sbit LCD_EN_Direction at TRISB2_bit;
sbit LCD_D4_Direction at TRISB3_bit;
sbit LCD_D5_Direction at TRISB4_bit;
sbit LCD_D6_Direction at TRISB5_bit;
sbit LCD_D7_Direction at TRISB6_bit;

// Initialisation du clavier dans le port D
char  keypadPort at PORTD;

unsigned short kp, count=0,i;
unsigned short CntWrongPW=0;
unsigned short Password[LengthPassWord]={1,2,3,4,5};


void Tone1() {
  Sound_Play(659, 250);   // Frequency = 659Hz, duration = 250ms
}

void Tone2() {
  Sound_Play(698, 250);   // Frequency = 698Hz, duration = 250ms
```

```
}

void Tone3() {
  Sound_Play(784, 250);   // Frequency = 784Hz, duration = 250ms
}

void Melody() {           // Plays the melody "Yellow house"
  Tone1(); Tone2(); Tone3(); Tone3();
  Tone1(); Tone2(); Tone3(); Tone3();
  Tone1(); Tone2(); Tone3();
  Tone1(); Tone2(); Tone3(); Tone3();
  Tone1(); Tone2(); Tone3();
  Tone3(); Tone3(); Tone2(); Tone2(); Tone1();
}

void Melody_Happy_BD(void)
{
    unsigned int Notes_Hz[6]={262,262,294,262,349,330};
    unsigned int Duration_ms[6]={200,200,400,400,400,500};
    unsigned int i;

    for(i=0;i<6;i++)
    {
       sound_play(Notes_Hz[i],Duration_ms[i]);
       delay_ms(100);
    }
}

void Melody_alarme_1(void)
{
    Sound_Play(400, 100);
    delay_ms(50);
    Sound_Play(600, 50);
    delay_ms(50);
    Sound_Play(800, 100);
    delay_ms(50);
    Sound_Play(700, 50);
    delay_ms(50);
    Sound_Play(500, 50);
}

void Melody_alarme_2(void)
{
    Sound_Play(400, 100);
```

```
    delay_ms(50);
    Sound_Play(500, 70);
    delay_ms(50);
    Sound_Play(600, 80);
    delay_ms(50);
    Sound_Play(700, 48);
    delay_ms(50);
    Sound_Play(800, 102);
    Sound_Play(900, 150);
    delay_ms(50);
    Sound_Play(1000, 40);
}
void Melody_alarme_3(void)
{
    Sound_Play(1500, 100);
    delay_ms(50);
    Sound_Play(2000, 50);
}

unsigned short GetKeyPressed(void)
{
    kp=0;
    do
      //kp = Keypad_Key_Press();
      kp = Keypad_Key_Click();
    while (!kp);

    switch (kp)
    {
      case  1: kp = 49; break; // 1
      case  2: kp = 50; break; // 2
      case  3: kp = 51; break; // 3
      case  4: kp = 65; break; // A
      case  5: kp = 52; break; // 4
      case  6: kp = 53; break; // 5
      case  7: kp = 54; break; // 6
      case  8: kp = 66; break; // B
      case  9: kp = 55; break; // 7
      case 10: kp = 56; break; // 8
      case 11: kp = 57; break; // 9
      case 12: kp = 67; break; // C
      case 13: kp = 42; break; // *
      case 14: kp = 48; break; // 0
      case 15: kp = 35; break; // #
```

```
      case 16: kp = 68; break; // D
    }
    return kp ;
}

void Sens_1_motor(void)
{
    PORTC.F5=1;
    PORTC.F6=0;
    delay_ms(Temps_ms_sens1);
}

void Sens_2_motor(void)
{
    PORTC.F5=0;
    PORTC.F6=1;
    delay_ms(Temps_ms_sens2);
}

void Frein_motor(void)
{
    PORTC.F5=0;
    PORTC.F6=0;

    // Ou bien
    //PORTC.F5=1;
    //PORTC.F6=1;
}


void main()
{

  TRISB = 0x01;
  TRISC = 0x00;
  PORTC=  0x00;

  Keypad_Init();                          // Init clavier
  Sound_Init(&PORTC, 4);                  // Init Sound
  Lcd_Init();                             // Init LCD
  Lcd_Cmd(_LCD_CLEAR);                    // Effacer LCD
  Lcd_Cmd(_LCD_CURSOR_OFF);               // Désactiver le curseur

  Lcd_Out(1, 1, "1");
```

```
Lcd_Out(1, 1, "Key  :");                 // Write message text on LCD
Lcd_Out(2, 1, "Password:");


while(1)
{
  for(i=0;i<LengthPassWord;i++)
  {
      kp=GetKeyPressed();
      Lcd_Chr(1, 10+i, kp);
      Lcd_Chr(2, 10+i, '*');
      if(Password[i]+48==kp  ) count++;
  }

  if (count==LengthPassWord)
  {
    count = 0;
    Lcd_Out(2,10,"Correct");
    // LED ON
    PORTC.F0=1;

    // DOOR ON
    PORTC.F3=1;
    Melody_Happy_BD();

    // Ouverture de la porte
    Sens_1_motor();
    Frein_motor();
    delay_ms(Temps_ms_door);
    Sens_2_motor();
    Frein_motor();

    // INIT
    PORTC=0x00;
  }
  else
  {
    count++;
    Lcd_Out(2,10,"Erreur");
    PORTC.F1=1;
    delay_ms(1000);
    PORTC=0x00;

    CntWrongPW++;
```

```
      if(CntWrongPW==NumCodeRepeat)
      {
         CntWrongPW=0;
         count=0;
         while(kp=GetKeyPressed()-48!=Enable_PW)
         {
             Lcd_Out(2,1,"Password Blocked");
             Lcd_Out(1,1,"Password Blocked");
             PORTC.F1=1;

             while(1)
             {
                 Melody_alarme_1();
                 //Melody_alarme_2();
                 //Melody_alarme_3();
                 //Melody();
                 if(PORTB.F0==1) break;
             }
             PORTC=0x00;

         }
      }
    }

    delay_ms(1000);

    Lcd_Cmd(_LCD_CLEAR);
    Lcd_Out(1, 1, "Key  :");
    Lcd_Out(2, 1, "Password:");

  }
}
```

Télécharger gratuitement le fichier du projet : Serrure codée à base du microcontrôleur PIC16F877 (Schéma ISIS + Code MikroC)

- Projet électronique Sérrure codée électronique à base du micro 16F877

****

Un petit commentaire de vous, un Grand encouragement pour nous

Click to rate this post!
[Total: 3 Average: 4.3]

> Nous Soutenir