Braille vocalization keyboard

Wanderson da Silva Maciel Filho, Maurílio Nunes Vieira
Departamento de Eletrônica/Escola de Engenharia/UFMG - Brazil

Beatriz Fonseca Torres, Regina Célia Passos Ribeiro de Campos
Faculdade de Educação/UFMG - Brazil

1. Introduction

This report describes the development of the prototype of a keyboard to aid the literacy of a visually impaired child. One goal is the learning of the Braille symbols for letters and numbers. The user is enrolled in a class with students without visual loss and another goal is the inclusion of the disabled in this class. For this, the keyboard allows not only the sound reproduction of the characters but also the registration and visualization of keystrokes typed for accompaniment of the teacher or of the colleagues (Figure 1).

Figure 1: Visually impaired child using the prototype.

Figure 2 shows details of the prototype. The keys are arranged alphabetical group and numeric groups. Wooden circles with the Braille symbols were glued over the keys. There are two loudspeakers (SP1 and SP2), a liquid crystal display (LCD) and switches to mute and adjust the volume (Vol- and Vol +). The user can also hear the sound of the letters through a headset whose connector is below the mute key. The four black circular buttons on the bottom (unidentified) are reserved for future improvements of the prototype.

Figure 2: User interface aspects of the prototype. See details in the text.

Figure 3 presents the inside of the keyboard, highlighting the LDC i2c controller board, the power supplies (AC/DC), and the main control board. This version of the keyboard does not use batteries.

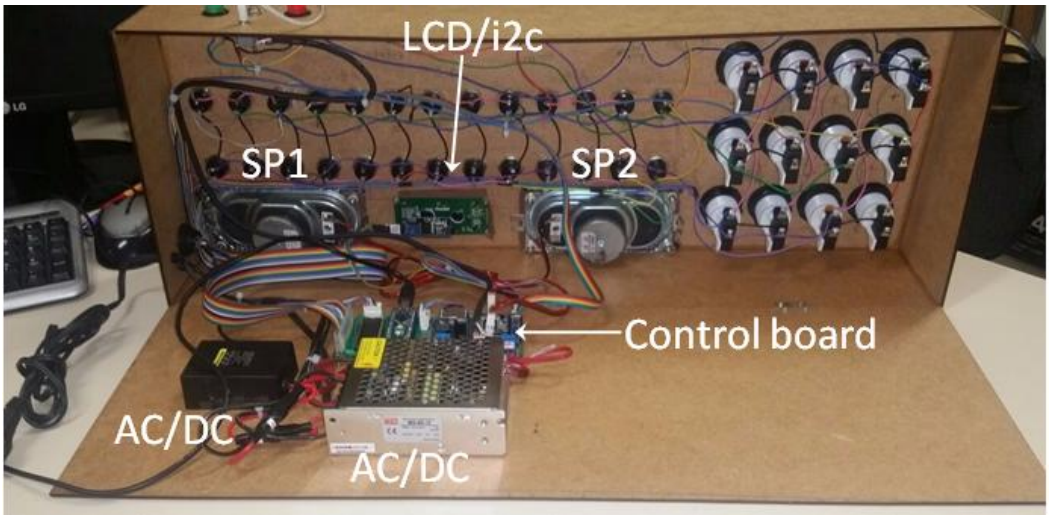

Figure 3: Interior of the prototype. See text for details.

Figure 4 shows the control board and its main components: the Arduino Nano, the keyboard decoder (CD 4515), the audio processor (DfPlayer mini), and the power amplifier (Pam 8610). Keyboard decoding could have been implemented with obsolete PS2 serial keyboard controllers, but the lack of standardization in the layout of the keyboard matrices motivated the use of a circuit based of the CD 4515, a 4 to 16 decoder.

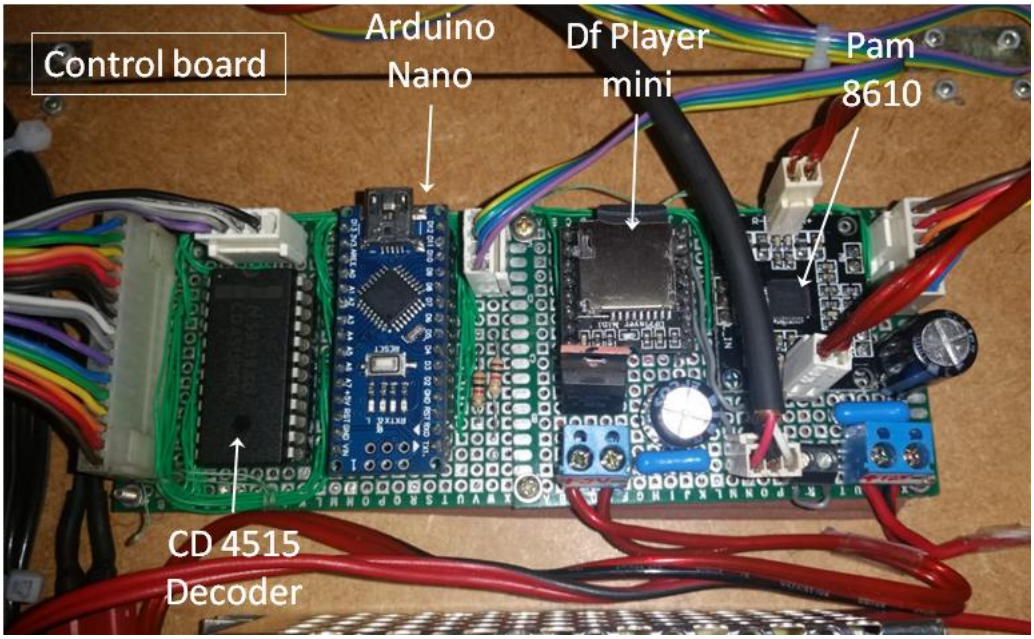

Figure 4: Main control board. See text for details.

2. Operation

Basically (Figure 5), the microprocessor scans the keyboard, detects if any key has been pressed and plays the corresponding audio file (mp3 or wav). The typed character is also displayed on the LCD. Keys allow the adjustment of audio loudness.

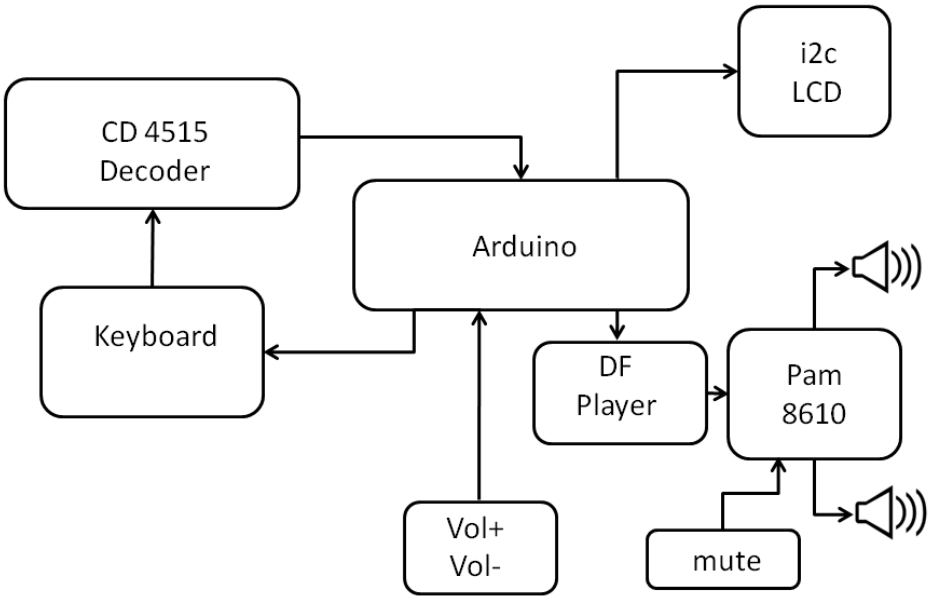

Figure 5: Schematic diagram of the control electronics. See text for details.

3. Main control board

3.1. Hardware

The schematic diagram of the control board is shown in Figure 6. U2 (Arduino Nano) does the main control of functions. U1 (CD4515) decodes the 4 binary addressing lines (Arduino digital outputs D8 ... D11) into up to 16 individual drive lines (Y0 ... Y15). In the project, 14 outputs (Y0 ... Y13) are used, each line addressing a 3-key group. In this way, Y0 addresses keys A, B or C; Y1 addresses keys D, E, or F, and so on. All alphanumeric keys have momentary contact. The 3-key groups are interconnected by the return lines (Ret1, Ret2 and Ret3), as shown in the schematic diagram. These return lines are Arduino digital inputs that allow the detection, by software, of the key activated in the respective 3-key group.

Once a pressed key is detected, the Arduino software will control the playback of the audio files and the display of the characters on the LCD.
A code will be sent to DFPlayer to reproduce the respective file. All audio files have been previously recorded, edited and stored on a MicroSD card housed in the DFPlayer. For complete playback control, volume control functions were performed on Arduino and not using the DFPlayer pins. Audio signals in DFPlayer output have adequate power for earphones, which can be connected to J1, but require amplification for playback on loud speakers so that all students in the class room listen to the sound. This amplification is performed by Pam8610, a 15 W per channel stereo class D amplifier. It was necessary isolated the power supply (12 V × 5 A) of this amplifier to avoid interference with other circuits. Attempts to use filters, etc., were not successful to avoid crosstalk noise. The 9 V × 1 A power source is connected to Arduino, which, in turn, powers the CD451 through the + 5V output. The 9V power source is also connected to U3 (LM 7805 regulator) to provide 5V x 1A (+ 5Vpwr) for the LCD display and the DFPlayer.

Arduino also control the display of characters on the LCD (16 columns x 2 lines). This allows the teacher or colleagues to follow the actions of the user.

3.2. Software

The comments in the source code, attachment, give the details of the hardware control, links to libraries, etc. Direct access to the microprocessor ports (PORTB) resulted in a compact code. The keys are addressed through variables alfabeto [][] and Order [][], which include other keys (ENT, SPC, DEL, ?) aimed at text edition. In the future, resources can be developed to aid to writing learning, such as the management of playing syllables, words, etc.

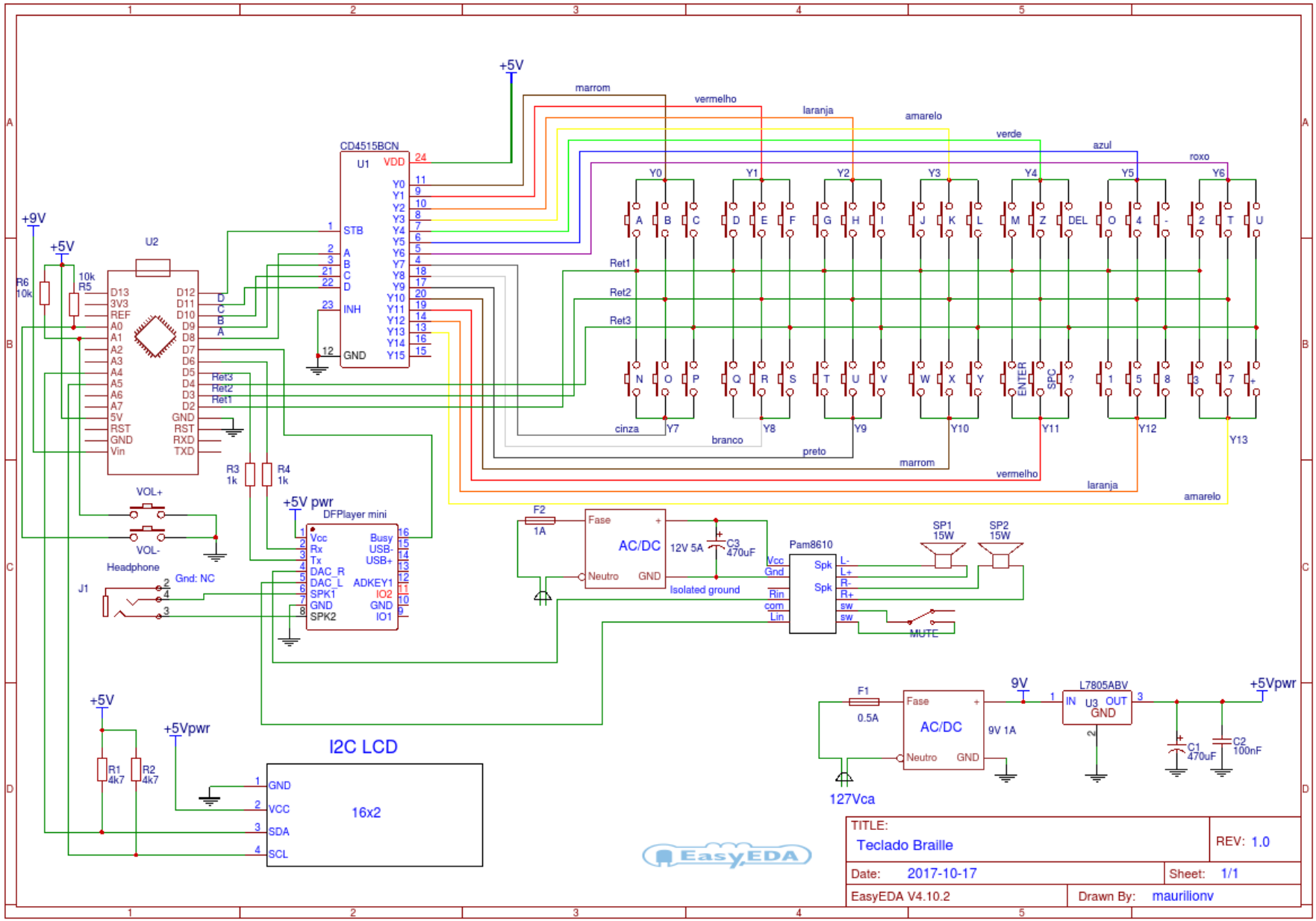

Figure 6: Detailed schematic diagram

4. Source code

Available from

https://drive.google.com/open?id=1iKNauiLZP-CNw-wMV79ck8yiDLUtnyXp

```
  /*   UFMG / Esc.Engenharia  / Depto Eletrônica
   *   Prof. Maurílio Nunes Vieira, Estudante Wanderson Maciel
   *
   *  UFMG / Fac Educação/ LAPOA - GEINE
   *  Profa. Regina Célia Passos Ribeiro de Campos,  Estudante Beatriz Fonseca Torres
   */

// i2c LCD
#include <Wire.h>
#include <LiquidCrystal_I2C.h>

// MP3 Player
#include <SoftwareSerial.h>
#include "DFPlayer_Mini_Mp3.h"

// CD4515 (4 to 16 decoder) address lines
#define strobe 12
#define A       8        // D8  = PORTB PB0 ... Data 1 => 4515/pin 2  (lsb)
#define B       9        // D9  = PORTB PB1 ... Data 2 => 4515/pin 3
#define C       10       // D10 = PORTB PB2 ... Data 3 => 4515/pin 21
#define D       11       // D11 = PORTB PB3 ... Data 4 => 4515/pin 22 (msb)
#define strobe  12       //        => 4515 pino 1

//                    ------------------------------------------
//           PORTB:   PB7  PB6  PB5   PB4  PB3   PB2   PB1   PB0
// Arduino UNO/NANO:            D13   D12  ~D11  ~D10  ~D9   D8
// CD 4515                          Srobe  D     C     B    A
//                    ------------------------------------------

// kebyboard return lines
#define Ret1    2                    // keyboard return 1
#define Ret2    3                    // keyboard return 2
#define Ret3    4                    // keyboard return 1

#define DEL     127                  // ASCII for "delete"
#define ENT     13                   // ASCII for CR = Carriage Return ('enter')
#define SPC     32                   // ASCII for Space

#define COL     14                   // nb of columns in alfabeto[3][COL]

// Mp3Player
#define Rx      5
#define Tx      6
#define Busy    7
#define VOL     15                    // default Volume (0 ... 30)
#define VolUp   1                     // A1 input (pull up resistor)
#define VolDown 0                     // A0 input (pull up resistor)
#define EQ      2                     // 0/1/2/3/4/5  =  Normal/Pop/Rock/Jazz/Classic/Bass

void mp3_set_EQ (uint16_t eq);
// i2C LCD Display at address 0x3f
LiquidCrystal_I2C lcd(0x3f,2,1,0,4,5,6,7,3, POSITIVE);

// creates character for "enter" (carriage return)
byte EnterSymbol[8] = {
  B00000,
  B00001,
  B00001,
  B00101,
  B01001,
  B11111,
  B01000,
  B00100
};

// MP3 Player
SoftwareSerial mySerial(Rx, Tx);
```

```
void setup(){
  // LCD i2C   powered by extenal supply
  // i2c uses pins A4 and A5 (with 4.7 kohm pull up resistors)
  lcd.begin (16,2);
  lcd.setBacklight(HIGH);
  lcd.setCursor(0,0);
  lcd.clear();
  lcd.blink();
  lcd.print("UFMG - Out/2017");
  lcd.setCursor(0,1);
  lcd.print(" Inicializando ");
  lcd.createChar(ENT,EnterSymbol);

  // CD4515 connections
  pinMode(A, OUTPUT);                // Data 1 A (pino 2 4515)
  pinMode(B, OUTPUT);                // Data 2 B (pino 3 4515)
  pinMode(C, OUTPUT);                // Data 3 C (pino 21 4515)
  pinMode(D, OUTPUT);                // Data 4 D (pino 22 4515)
  pinMode(strobe, OUTPUT);           // Strobe (pino 1 4515)

  // keyboard return lines
  pinMode(Ret1, INPUT_PULLUP);       // selected by alfabeto[0,i], i = 0 ... 11
  pinMode(Ret2, INPUT_PULLUP);       // selected by alfabeto[1,i], i = 0 ... 11
  pinMode(Ret3, INPUT_PULLUP);       // selected by alfabeto[2,i], i = 0 ... 11

  // MP3 Player
  // http://www.picaxe.com/docs/spe033.pdf
  /* MP3 Player
    http://www.picaxe.com/docs/spe033.pdf
    ".mp3" ou ".wav" files:
     - MUST BE in folder /mp3
     - MUST BE numbered with 4 numbers:
       0001.mp3   (a)
       0002.mp3   (b)
        ...       ...
       0038.mp3   (-)

     - File name ca have other text after inintial 4 numbers:
       0001a.wav      (a)
       0002b.wav      (b)
        ...           ...
       0038menos.wav  (-)
  */

  /*************************************************
  **Arduino          DFPlayer:
  *Pin5 -- 1kohm --   TX (3);
  *Pin6 -- 1kohm --   RX (2);
  *pin7               BUSY (16)
  *5V                 1 (Vcc)
  *GND                7 (Gnd)
  *                   6 headphone +
  *                   8 headphone -
  *                   4 DAC_R  => PAM 8610 power amp
  *                   5 DAC_L  => PAM 8610 power amp
  *************************************************/
  pinMode(Busy, INPUT);
  Serial.begin (9600);
  mySerial.begin (9600);
  mp3_set_serial (mySerial);  //set softwareSerial for DFPlayer-mini mp3 module
  delay(5000); // delay 1ms to set volume
  mp3_set_volume (VOL); // value 0~30; default VOL = 25
  delay(10);

  mp3_set_EQ (EQ);    // default Equalizer  0/1/2/3/4/5  Normal/Pop/Rock/Jazz/Classic/Bass


  // PAM 8610 (2 x 15W)?
  // https://tronixlabs.com.au/breakout-boards/amplifier/mini-amplifier-breakout-board-2-x-3w-class-d-pam8403/

  hello(2000);            // mensagens iniciais 2000 = delay in ms

}
```

```
/*   saída do 4515
               Y0 =    0    1    2    3    4      5    6    7    8     9   10    11     12    13 */

char alfabeto[3][COL]={{'a', 'd', 'g', 'j', 'm',   '0',  '2', 'n', 'q', 't', 'w', ENT,   '1',  '3'},
                       { 'b', 'e', 'h', 'k', 'z',   '4',  '6', 'o', 'r', 'u', 'x', SPC,   '5',  '7'},
                       { 'c', 'f', 'i', 'l', DEL,   '-',  '9', 'p', 's', 'v', 'y', '?',   '8',  '+'}};

uint16_t Oder[3][COL] = {1,   4,   7,  10,   13,   27,   29,  14,  17,  20,  23,  40,    28,   30,
                         2,   5,   8,  11,   26,   31,   33,  15,  18,  21,  24,  41,    32,   34,
                         3,   6,   9,  12,   39,   38,   36,  16,  19,  22,  25,  42,    35,   37};

/*
ordem dos arquivos de áudio (podem ser inseridos helps controlados, por exemplo, com pulsadas do botão '?'
a b c d e f g h i j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z  0  1  2  3  4  5  6  7  8  9  +  -  del ent spc ?
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39  40  41  42

*/

int lcd_count = 0;    // Global variable (usar static?)
void loop() {
  Address4515();
  VolCntr();
}


void Address4515() {
  int  j;
  byte i;

  for( i=0; i < COL; i++) {
     /*
     4515 input: 0000 DCBA   (i)
                 0000 0000   (0)
                 0000 0001   (1)
                 0000 0010   (2)
                 0000 ...    ...
                 0000 1100   (COL)
   */
    digitalWrite(strobe, HIGH);             // Enable addressing

    PORTB = PORTB & B11110000;              // clear Data1 ... Data4 of CD4515
    PORTB = PORTB | i;                      // Address CD 4515

    digitalWrite(strobe, LOW);              // disable addressing

    j = KeyScan();                          // key pressed?

    if(j<3){                                // yes
      if(alfabeto[j][i] != DEL && alfabeto[j][i] != ENT && alfabeto[j][i] != SPC && alfabeto[j][i] != '?' ) {
         LcdDisplay(j,(int)i);
         Audio(Oder[j][i]);
       }
    }
  }
}



int KeyScan()
/*
 * CD4515 applies LOW to addressed output
*/
{
 int j;
 for(j = 0; j<3; j++){
  if(digitalRead(j+Ret1) == 0) {
    delay(200);                             // empirical
    while(digitalRead(j+Ret1) == 0) {
      // wait while key is pressed
    }
    break;
  }
 }
 return j;  // j = 3 ==> no key was pressed
}
```

```
void LcdDisplay(int j, int i) {
  // lcd_count: global variable
  if(lcd_count==16){                      // last column?
    lcd.setCursor(0,1);                   // begin of 2nd line
  }

  if(lcd_count > 31){                     // is last position in LCD?
    lcd.clear();                          // clear and move cursor to (0,0)
    lcd_count=0;
  }

  if(alfabeto[j][i] == ENT) {
    lcd.print((char) ENT);
    lcd_count++;
  }
  else {
    lcd.print(alfabeto[j][i]);
    lcd_count++;
  }

  if(i==25 && lcd_count != 16 && lcd_count != 32) {
     lcd.print(" ");
  }
  delay(50);
}


void Audio(uint16_t num) {
 do{
 }while (!digitalRead(Busy));
 mp3_play(num);
 delay(10);                               // Is this line necessary?
}


void hello(int tempo) {
  // mensagens iniciais
  // lcd_count: global variable

  lcd.clear();
  lcd.noBlink();
  lcd.setCursor(0,0); lcd.print("Alfabeto Vocali-");
  lcd.setCursor(0,1); lcd.print("zador em Braille");
  delay(tempo);

  lcd.clear();
  lcd.setCursor(0,0); lcd.print("FAE/LAPOA/GEINE");
  lcd.setCursor(0,1); lcd.print(" Profa Regina");
  delay(tempo);

  lcd.clear();
  lcd.setCursor(0,0); lcd.print("FAE/LAPOA/GEINE");
  lcd.setCursor(0,1); lcd.print("Beatriz Fonseca");

  delay(tempo);
  lcd.clear();
  lcd.setCursor(0,0); lcd.print("DElt/Cefala/LSI");
  lcd.setCursor(0,1); lcd.print(" Prof Maurilio");
  delay(tempo);

  lcd.clear();
  lcd.setCursor(0,0); lcd.print("DElt/Cefala/LSI");
  lcd.setCursor(0,1); lcd.print("Wanderson Maciel");
  delay(tempo);

  lcd.clear();
  lcd.blink();
  lcd_count=0;                          // global variable
}
```

```
void VolCntr(){
  static int vol = VOL;                // initialize first time only
  int x;

  if (x = analogRead(VolUp) < 100) {  // check VolUp key
    do{
      if ((vol++) > 30) {
        vol = 30;
      }
      mp3_set_volume (vol); // value 0~30;
      delay(100);
      // LcdDisplay(2,13); // +

    } while ((x = analogRead(VolUp) < 100));
  }

  if (x = analogRead(VolDown) < 100) {  // check VolDown key
    do{
      if ((vol--) < 0) {
        vol = 0;
      }
      mp3_set_volume (vol); // value 0~30;
      delay(100);
      // LcdDisplay(2,5); // -

    } while ((x = analogRead(VolDown) < 100));
  }
}
```