Arduino FM Radio Receiver

By Jacob Lee

Fitchburg State University
Microprocessors Course CSC 3600
Prof. Kevin Autin
May, 16 2016

Table of Contents

Introduction

For the computer science course microprocessors we needed to create a project idea, wire it, code it and present it for the end of the semester. The guidelines for this project were as follows

- At least two peripheral devices one of which needs to communicate over I2C or SPI
- Have an input and output
- A data processing step
- Code has at least three functions

We could not use 3rd party libraries and could only use the following libraries from arduino (Wire, SPI, Serial, Software Serial and Servo) other libraries needed to be approved by the professor. We had many platforms and peripherals to work with some of which we used in the labs. Using devices we had no experienced with was encouraged.

Project Goals

The goal for my project was to make something interesting and fun, and challenge myself by using devices I had little to no experience with. I came up with an FM radio receiver. The idea is that you type the station number into a number keypad (0-9, *, #) using the * as the decimal point and # as the submit button. The station will be shown on the LCD screen along with signal strength and if it's receiving in Stereo or not. I also wanted a smooth user interface so when you type the station in it shows the numbers on the LCD as they are typed. For even more smoothness when you stop typing a station number it has a 5 second timeout period and returns to the old station and screen.

Project concepts

The core of the radio is an Arduino Duemilanove. The Arduino is connected to 3 peripheral devices, an I2C FM receiver chip, I2C LCD screen, and a matrixed membrane number keyboard(0-9, *, #). The idea is that you have a radio chip to receive the radio station and signal information. An LCD screen to display the station, signal strength and whether or not we are receiving in Stereo. To change station you use a matrixed keypad.

Hardware Methods

Choosing hardware was easy I have an arduino, and had used the FM radio chip briefly before and knew it would do exactly what I needed as for the Lcd it was also I2C like the fm chip and would help reduce pins required, also we used them in lab. The membrane matrixed keypad seemed like a more challenging input device for the radio station then and up and down button.

Platform

The platform I choose to use is the Arduino Duemilanove. This Arduino is of older model but works perfectly for my project, the board has the following features.

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB of which 2 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

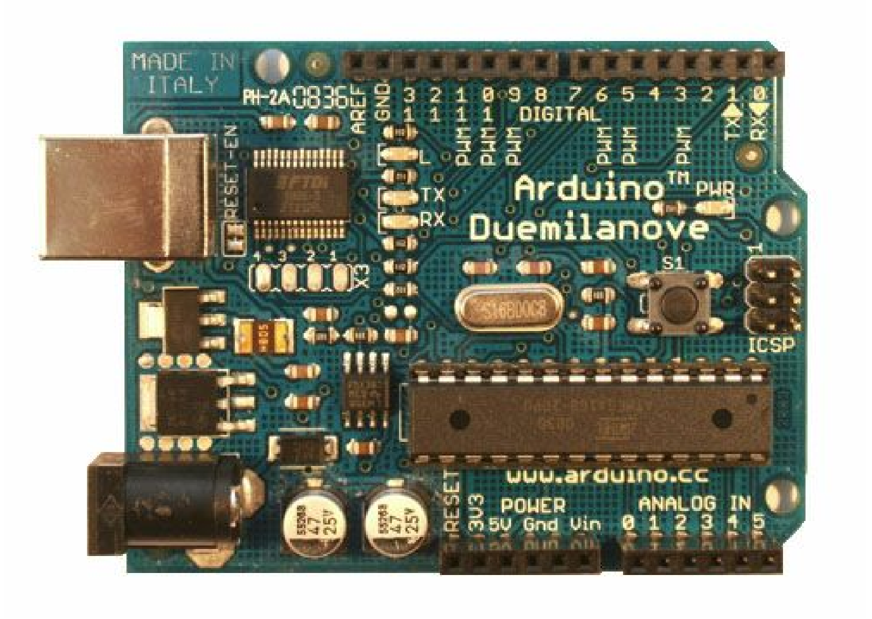

Figure 1

Devices

Peripherals connected where an I2C LCD, I2C TEA5767 FM Radio, Membrain 0-9, *, # matrixed keypad.

I2C LCD

Function: Displays characters on a backlit 20x4 LCD
Interface: I2C
Description: The Lcd has an I2C backpack for making communication with the 20x4 Lcd easier and requires less pins. Lcd allows for displaying information from the arduino.
Diagram(s):

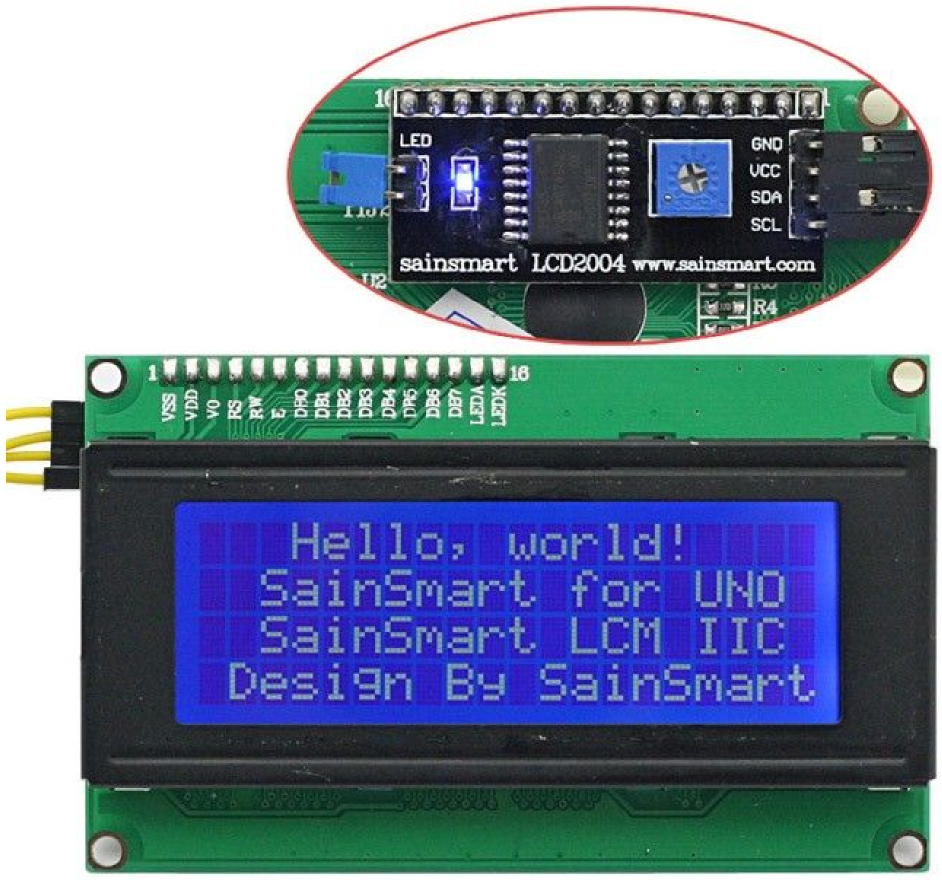

Figure 2

I2C TEA5767 Fm Radio

Function: Outputs low-level audio signals for programmed radio station, Return signal level out of 15 and whether or not it's receiving in stereo.
Interface: I2C
Description: The radio is one of the most important parts of the project it receives Fm signals and outputs the audio as low-level signals. Also it generates the signal strength and checks for stereo reception. This info is displayed on the LCD for the user.
Diagram(s):

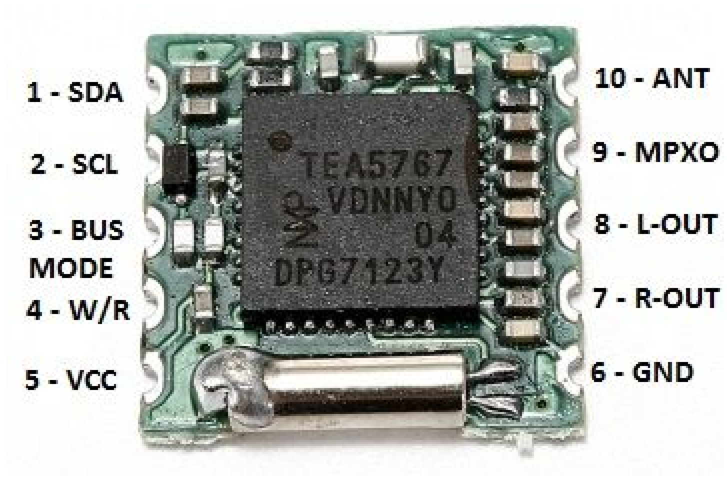

Figure 3

Membrane Matrixed keyboard

Function: Takes user input in the form of 12 membrane keys matrixed together with 4 rows and 3 columns. Has the numbers 0-9 with special keys * and #.
Interface: Matrixed 4x3 buttons
Description: The keypad has 12 keys in the 4x3 arrangement like a telephone with the * and # keys. This is how the user types in the station they want using the * as a decimal point and the # as the done/submit key.
Diagram(s):

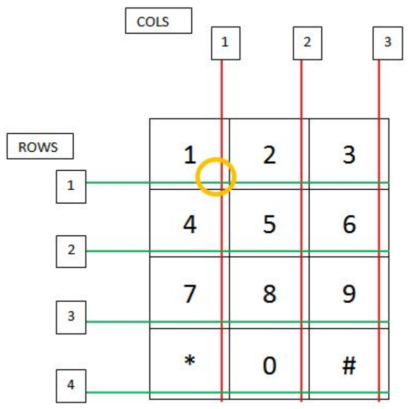

Figure 4

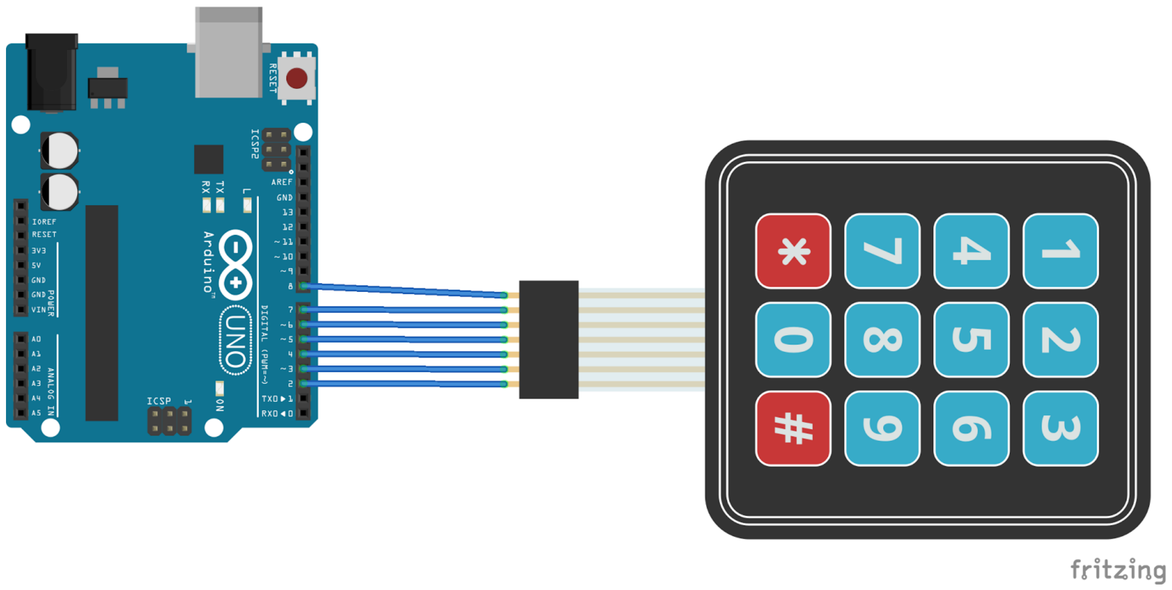

Figure 5

Software Development

Software was the most challenging part of the project, I took it one step at a time though and managed to include all the features I wanted and they worked flawlessly.

Design

I divided my program into 4 steps (Keypad functions, Basic radio functions, LCD functions, Final touches and features).

Keypad Functions

This is one of the most important parts of the program it's functions are responsible for checking if a key was pressed and getting the value. These functions are **keypad()** and **getKey(int val)**. This is the first thing I did when I started. They keypad function give back value between 0-15 if a key is pressed otherwise it returns 50, these values correspond to keys in no particular order. I had to record what each key's number was using the serial monitor. Next getKey was created which contains a switch statement using the parameter it's sent and returns the character that was pressed.

Basic Radio Functions

After getting the keypad functions working I moved on to trying to change the radio station with the keypad. I did this so I could make sure things were working before complicating things by adding the LCD. I did lots of debugging at this step using the serial monitor. The function created at this stage were a very early version of **chkBtns()** and a complete version of **setFrequency()**. chkBtns is the main function in the whole program that checks for a button push by calling keypad() and checking if the returned value of less than 50.

Lcd Functions

I had a bit of an advantage here from the functions we used in the lab for the lcd **sendData(byte dataByte)**, **sendCmd(byte cmdByte)** and **resetLCD()**. What we did not do in the lab was create a function that could send strings to the lcd instead of individual characters. This is one of the most important functions since everything displayed on the lcd is sent here. The function is **stringLcd(String text)**. It utilizes the toCharArray arduino function to convert the string to a char array and then loop through it passing each char to sendData().

Final Touches and Features

The final touches included getting the lcd formatted the way I wanted, adding the signal strength functions and the stereo function along with getting the timeout delay implemented. Functions that came along at this stage were **dispADC(), getAdcLvl(), getStereo(),** and **currentLcd()**.

Testing

Each time I made small changes or added a function I tested it to make sure it worked. I only added more function once everything I already had was working. Lots of debugging was done with the serial monitor by printing variables at different stages to pinpoint problem causing code. Some of the things that took the longest were chkBtns(), all keypad functions, and surprisingly stringLcd(). The keypad functions got pretty confusing easily, I had to do lots of debugging for those functions. chkBtns was a major point of constant issues since it is the most important function its took me a while to figure out how I wanted the user interface to work so I was constantly changing this function. Also that function contains the timeout delay which took a few hours of fiddling to get working. The stringLcd function took me a while even though it is simple in principle I started by trying to send a char array to the function as a parameter but found that sending a string and converting it worked much better for my purposes this took a while of fooling with but was a major stepping stone for the rest of the code. Mostly what I learned is just take things one step at a time and just keep connecting every part to the next until it all work together.

Conclusion

Overall I am quite happy with how the radio turned out especially in the short time period I build it in, around 9 hours of straight programming. I got all the features I originally thought of. One major feature that I would like to have added would be storing the radio station in the arduino's eeprom memory. This would have allowed me to have the radio stay on the same station even when power cycled. Also I wanted to scale the signal strength from 0-15 to 0-18 so that with the 2 side brackets it would fill an entire line of the lcd. I would have like to add more preventative measures against allowing a user to choose an invalid radio station, like only allowing odd numbers. I will take away from this course that you can figure it out if you persevere.

Evaluations

Final project I did 100% of everything!

Semester Lab Evaluation

Area	**Allen**	**Josh**	**Me**	**Irving**	**Total**
Code Development	50%	50%	50%	0%	150%
Code Analysis	50%	50%	50%	0%	150%
Data Sheet Interpretation	50%	50%	50%	0%	150%
Breadboarding	50%	50%	50%	0%	150%
Code Debugging	50%	50%	50%	0%	150%
Hardware Research	50%	50%	50%	0%	150%
Research	50%	50%	50%	0%	150%

Appendix

Source Code:

```
/**
   Jake Lee 2016

   FM Radio w/ LCD and Keypad

   This code untilizes an I2C LCD and a 4x3 matrixed number keypad and the I2C Fm radio
chip TEA5767

   The user can change the radio station by typing it in using the * as a period and # as the
   submit button. The LCD shows the current station and whether or not its in stereo. It also
   show the signal strength out of 15.

*/
//Wire library used for I2C comunication
#include <Wire.h>

//LCD defines for sending data and commands
#define ADDR 0x27
#define RSC 0
#define RSD 1
#define RW_READ 2
#define WR 0
#define EN_HIGH 4
#define EN_LOW 0
#define BL 8

//Radio frequency change variables
unsigned char frequencyH = 0;
unsigned char frequencyL = 0;
unsigned int frequencyB;
float frequency = 0;
float count=0.1;

//Keypad Functions variables
byte h=0,v=0;                           //variables used in for loops
const unsigned long period=50;          //little period used to prevent error
const unsigned long setPeriod = 5000;   // period of wait for
unsigned long kdelay=0;                 // variable used in non-blocking delay
unsigned long setDly=millis();          // var used for reseting screen after no more button
presses
```

```
const byte rows=4;                    //number of rows of keypad
const byte columns=3;                 //number of columnss of keypad
const byte Output[rows]={5,6,7,8};      //array of pins used as output for rows of keypad
const byte Input[columns]={2,3,4};      //array of pins used as input for columnss of keypad

//Other variables
int i=0;
char station[4];                      //Holds the buttons pushed
String masterLCD = "Frequency: ";       //used for holding last value presed on keypad when
chaging station
int lbpF=0;                           //Last button pushed timer flag

// function used to detect which button is used
byte keypad()
{
 static bool no_press_flag=0;         //static flag used to ensure no button is pressed
  for(byte x=0;x<columns;x++)          // for loop used to read all inputs of keypad to ensure no
button is pressed
  {
     if (digitalRead(Input[x])==HIGH);   //read every input if high continue else break;
     else
     break;
     if(x==(columns-1))   //if no button is pressed
     {
      no_press_flag=1;
      h=0;
      v=0;
     }
  }
  if(no_press_flag==1) //if no button is pressed
  {
    for(byte r=0;r<rows;r++) //make all outputs low
    digitalWrite(Output[r],LOW);
    for(h=0;h<columns;h++)  //check if one of inputs is low
    {
      if(digitalRead(Input[h])==HIGH) //if specific input is remain high (no press on it) continue
      continue;
      else    //if one of inputs is low
      {
          for (v=0;v<rows;v++)   //specify the number of row
          {
          digitalWrite(Output[v],HIGH);      //make specified output as HIGH
          if(digitalRead(Input[h])==HIGH)    //if the input that selected from first sor loop is change
to high
          {
            no_press_flag=0;                //reset the no press flag;
            for(byte w=0;w<rows;w++)        // make all outputs low
            digitalWrite(Output[w],LOW);
            return v*4+h;                   //return number of button
          }
          }
      }
    }
  }
 return 50;
}

//Return the coresponding character from the value returned from keypad
char getKey(int val){
  switch (val)
  {
    case 0:
      return '#';
      break;
    case 1:
      return '0';
      break;
    case 2:
      return '.';
      break;
    case 4:
      return '9';
      break;
    case 5:
      return '8';
      break;
    case 6:
      return '7';
      break;
    case 8:
      return '6';
      break;
    case 9:
      return '5';
      break;
    case 10:
      return '4';
      break;
    case 12:
      return '3';
      break;
    case 13:
      return '2';
      break;
    case 14:
      return '1';
      break;
    default:
    ;
  }
}


//Creates the whole lcd layout with frequency, stereo, and signal strength
void currentLcd(){
      resetLCD();
      String tmp = "Frequency: ";
      tmp.concat(frequency);   //Concat allows me to combine a string and float
      tmp.remove(tmp.length()-1); //Removes the extra 0 from the 2 decimal float
      stringLcd(tmp); //Sends to lcd
      getStereo(); //Displays ST if stereo reception
      dispADC(); //Displays the signal strength bar graph
      masterLCD="Frequency: "; //Resets masterLCD
}

void chkBtns(){
    if(millis()-kdelay>period) //used to make non-blocking delay
  {
    kdelay=millis();  //capture time from millis function
    int keyV=keypad(); //get button press value
    char key = getKey(keyV); //get char assosiated with button press value

    //This is the timout code. If you don't finish typing in your station or
    //don't hit enter it resets the screen by calling currentLCD() because
    //frequency has not be changed yet.
    if (millis()-setDly>setPeriod){ //This checks to see if the system time millis() - the last time you
pressed a button (setDly) is greater then the timout delay
      while (lbpF==1){ //This gets set to 1 when a button is pushed and 0 when the station is
entered
        currentLcd();
        masterLCD="Frequency: ";
        i=0; //necessary to reset the char array that holds button pushes or else it shows the
buttons pushed before the timeout
        lbpF = 0; //reset flag
      }
    }

    //MAGIC RIGHT HERE
    //This is where all the magic happens if keyV is less than 50 meaning that a button was
pushed
    if (keyV<50){
      setDly=millis(); //Get system time that a button was pushed
      lbpF=1; //Set timeout flag
      resetLCD(); //resets the lcd
      if (i<=5){ //stations can only be a max of 5 chars ex. 101.2
        i++;
        stringLcd(masterLCD += key); //add the key pressed to the lcd display
        station[i-1] = key; //add the key pressed to the station char array

      }
    }
     if (key == '#'){ //The station has be entered
         if (i==4){ //checks for a station that is only 4 chars long like 94.5 and adds an extra 0 to fill
the char array
         station[4] = '0';
         }
         float sVal=atof(station); //converts the char array station containing the station to a float
         frequency = sVal; //sets the global frequency
         setFrequency(); //set the frequency on the chip
         delay(100);
         currentLcd(); //shows the new frequency on the lcd
         lbpF=0; //resets timout flag
         i=0;  //resets station char array
        }

    }
}

//Sets the frequency on the TEA5767
void setFrequency()
{
  frequencyB = 4 * (frequency * 1000000 + 225000) / 32768;
  frequencyH = frequencyB >> 8;
  frequencyL = frequencyB & 0XFF;
  delay(100);
  Wire.beginTransmission(0x60);
  Wire.write(frequencyH);
  Wire.write(frequencyL);
  Wire.write(0xB0);
  Wire.write(0x10);
  Wire.write((byte)0x00);
  Wire.endTransmission();
  delay(100);
}

//Sends data bytes to the LCD
void sendData(byte dataByte) {
  byte highNibble = dataByte & 0xF0;
  byte lowNibble = dataByte << 4;
  Wire.beginTransmission(ADDR);
  Wire.write(byte(highNibble|BL|EN_HIGH|WR|RSD));
  Wire.write(byte(highNibble|BL|EN_LOW|WR|RSD));
  Wire.write(byte(highNibble|BL|EN_HIGH|WR|RSD));
  Wire.write(byte(lowNibble|BL|EN_HIGH|WR|RSD));
  Wire.write(byte(lowNibble|BL|EN_LOW|WR|RSD));
  Wire.endTransmission();
}

//Send command bytes to the LCD
void sendCmd(byte cmdByte) {
  byte highNibble = cmdByte & 0xF0;
  byte lowNibble = cmdByte << 4;
  Wire.beginTransmission(ADDR);
  Wire.write(byte(highNibble|BL|EN_HIGH|WR|RSC));
  Wire.write(byte(highNibble|BL|EN_LOW|WR|RSC));
  Wire.write(byte(highNibble|BL|EN_HIGH|WR|RSC));
  Wire.write(byte(lowNibble|BL|EN_HIGH|WR|RSC));
  Wire.write(byte(lowNibble|BL|EN_LOW|WR|RSC));
  Wire.endTransmission();
}

//Resets the lcd
void resetLCD() {
  sendCmd(0x00); // reset display
  sendCmd(0x01);
  sendCmd(0x02); // go home
  sendCmd(0x06); // entry mode
  sendCmd(0x0C); // display on
  delay(100);
}

//Sends strings to the lcd
void stringLcd(String text){
  char charBuf[text.length()+1]; //creates a char array of the strings length
  text.toCharArray(charBuf, text.length()+1); //converts the string to a char array
  for (int j = 0 ; j < text.length() ; j++) //loops though array
  {
    sendData(charBuf[j]); //sends each char to the lcd
  }
}

//Gets the stereo bit from the TEA5767 in byte 3, bit 7
//if its 1 then stereo
//if its 0 then no stereo
//displays to the lcd
void getStereo(){
 unsigned char buffer[5];
 Wire.requestFrom(0x60,5); //reading TEA5767
  if (Wire.available())
  {
    for (int i=0; i<5; i++) {
      buffer[i]= Wire.read();
    }

   byte stereo = buffer[2]&0x80; //mask out bit 7
   if (stereo == 0x80){
    stringLcd("  ST");
   }else{
    stringLcd("    ");
   }

  }
  delay(100);
}

//Gets the analog to digital conversion of the signal strength from the TEA5767 byte 4 bit 4-7
int getAdcLvl(){
 unsigned char buffer[5];
 Wire.requestFrom(0x60,5); //reading TEA5767
  if (Wire.available())
  {
    for (int i=0; i<5; i++) {
      buffer[i]= Wire.read();
    }
  int adcLvl = (buffer[3]>>4);
  return adcLvl;
}
}

//Displays the signal strength to the lcd in the form of a progress bar 0-15
```

```
void dispADC(){
  int lvl = getAdcLvl();
  int olvl = lvl;
  stringLcd(" [");
  for (int q=0; q<15; q++){
    if (lvl!=0){
      stringLcd("#");
      lvl--;
    }else{
      stringLcd(" ");
    }
  }
  stringLcd("]   SIG: ");
  stringLcd((String)olvl);
  stringLcd("/15");
}

void setup()
{
  //make pin mode of outputs as output
  for(byte i=0;i<rows;i++){pinMode(Output[i],OUTPUT);}
  //make pin mode of inputs as inputpullup
  for(byte s=0;s<columns;s++){pinMode(Input[s],INPUT_PULLUP);}

  Serial.begin(9600);
  Wire.begin();

  frequency =  94.5; //starting frequency
  currentLcd();      //Set the lcd
  delay(100);
  setFrequency();    //Set the frequency
  delay(100);


}
void loop()
{
  chkBtns(); //Check for button pushes
}
```