```
//**************************************************
// FONA_MEGAprog17a
// 05.09.16
// Tests new old phone using FONA 3G & Arduino MEGA
// Operates basic functions of a phone
// Gets number from dial and connects
// Hangs up when cradle switch is pressed
// Recieves calls
// Plays sounds etc with Adafruit Music Maker
// Additional sound cues in this version
// Stores data on SD card
// Speed dial numbers etc
// Thanks to Adafruit for FONA library and other coding
//***************************************************

// First the Music Maker

#include <SPI.h>
#include <Adafruit_VS1053.h>
#include <SD.h>
#include <Wire.h>

// 0x4B is the default i2c address for MAX9744 audio amplifier
#define MAX9744_I2CADDR 0x4B

// These are the pins used for the Music Maker
#define BREAKOUT_RESET  9      // VS1053 reset pin (output)
#define BREAKOUT_CS     10     // VS1053 chip select pin (output)
#define BREAKOUT_DCS    8      // VS1053 Data/command select pin (output)
// These are the pins used for the music maker shield
#define SHIELD_RESET  -1      // VS1053 reset pin (unused!)
#define SHIELD_CS     7      // VS1053 chip select pin (output)
#define SHIELD_DCS    6      // VS1053 Data/command select pin (output)

// These are common pins between breakout and shield
#define CARDCS 4     // Card chip select pin
// DREQ should be an Int pin, see http://arduino.cc/en/Reference/attachInterrupt
#define DREQ 3       // VS1053 Data request, ideally an Interrupt pin

// create shield-example object!
Adafruit_VS1053_FilePlayer musicPlayer =
  Adafruit_VS1053_FilePlayer(SHIELD_RESET, SHIELD_CS, SHIELD_DCS, DREQ, CARDCS);


// Next the FONA
#include "Adafruit_FONA.h"

#define FONA_RX 14
#define FONA_TX 15
#define FONA_RST 22

// this is a large buffer for replies
char replybuffer[255];


// Hardware serial
HardwareSerial *fonaSerial = &Serial3;

Adafruit_FONA_3G fona = Adafruit_FONA_3G(FONA_RST);

boolean flag = true;
long counter = 0;
uint8_t type;
const byte maxDigit = 14;
char tele_num_str1[maxDigit];      // storage for dialed number
byte digit = 0;
byte count = 0;
boolean calling = false;
File myFile;
byte thevol = 63;
byte n = 0;                       // network status
const byte cradle = 47;

//**********************************************************

void setup() {

  // pin 50 when used as input and taken low prevents musicplayer from playing!
  pinMode(46, INPUT_PULLUP);   //end of pulse train switch, HIGH at end
  pinMode(48, INPUT_PULLUP);   //pulse train switch - pulse is HIGH
  pinMode(cradle, INPUT_PULLUP);  //handset cradle switch LOW handset OFF cradle
  pinMode(44, INPUT_PULLUP);  //call switch LOW when pressed
  pinMode(26, OUTPUT);        //speaker series resistor relay - high closed
  pinMode(24, INPUT);         //ring indicator
  pinMode(38, OUTPUT);        //green LED
  pinMode(40, OUTPUT);        //red LED
  digitalWrite(38, HIGH);

  Wire.begin();
  while (!Serial);

  Serial.begin(115200);
  //-------------------------------------------------------------------------


  delay(2000);
  // initialise the music player
  if (! musicPlayer.begin()) { // initialise the music player
     Serial.println(F("Couldn't find VS1053, do you have the right pins defined?"));
     while (1);
  }
  Serial.println(F("VS1053 found"));

  musicPlayer.sineTest(0x44, 500);    // Make a tone to indicate VS1053 is working

  if (!SD.begin(CARDCS)) {
    Serial.println(F("SD failed, or not present"));
    while (1);  // don't do anything more
  }
  Serial.println("SD OK!");

  // Set volume for left, right channels. lower numbers == louder volume!
  musicPlayer.setVolume(10,10);

  setvolume(thevol);

  // This option uses a pin interrupt. No timers required! But DREQ
  // must be on an interrupt pin.
   if (! musicPlayer.useInterrupt(VS1053_FILEPLAYER_PIN_INT))
    Serial.println(F("DREQ pin is not an interrupt pin"));


  //-----------------------------------------------------------------------------

  Serial.println(F("FONA basic test"));
  Serial.println(F("Initializing....(May take 3 seconds)"));

  fonaSerial->begin(4800);
  if (! fona.begin(*fonaSerial)) {
    Serial.println(F("Couldn't find FONA"));
    while (1);
  }
  type = fona.type();
  Serial.println(F("FONA is OK"));
  Serial.print(F("Found "));
  switch (type) {

    case FONA3G_E:
      Serial.println(F("FONA 3G (European)")); break;
    default:
      Serial.println(F("???")); break;
  }

  // Print module IMEI number.
  char imei[15] = {0}; // MUST use a 16 character buffer for IMEI!
  uint8_t imeiLen = fona.getIMEI(imei);
  if (imeiLen > 0) {
    Serial.print("Module IMEI: "); Serial.println(imei);
  }

  fona.setGPRSNetworkSettings(F("giffgaff.com"), F("giffgaff"), F(""));
  musicPlayer.sineTest(0x44, 500);    // Make a tone to indicate FONA initialised
}
//*********************************************************************
void flushSerial() {
  while (Serial.available())
    Serial.read();
}
//********************************************************************

void loop() {


  if(digitalRead(44) == LOW){              //call switch pressed triggers stored numbers routine
    if(n == 0){                            //unless there is no signal

      musicPlayer.playFullFile("nosignal.mp3");

    }
    else{
      num_fromfile();                      //function to select stored number
    }

 }
  counter++;
  if(counter == 100000){                         //poll the network every so often
    counter = 0;
    n = fona.getNetworkStatus();
    if(n == 1){
      flag = !flag;
      if(flag){
        digitalWrite(40, LOW);                  // flash red led if refgistered on network
      }
      else{
        digitalWrite(40, HIGH);
      }
    }
  }

  if(digitalRead(cradle) == LOW){                       //hanset off cradle - for call out or ringing?

     Serial.println("handset off cradle");

      if(n == 0){                                     //warn if not registered on network

       if (! musicPlayer.playFullFile("nosignal.mp3")) {

        Serial.println("Could not open no signal file ");             // same with playFullFile
       }
      else{
        Serial.println("no signal");
       }
    }


    byte stat = fona.getCallStatus();                 //check call status to see in dialling out or receiving
   if(stat == 0 && n == 1){
    dialout();
   }
   if(stat == 3){
    answercall();
   }

  }


}                //end of main loop

//---------------------------------------------------------------------------

void dialout(){

// won't open this file!? - not playFullFile either

  if (! musicPlayer.startPlayingFile("dialtone.ogg")) {
        Serial.println("Could not open file ");
        Serial.print("dialtone.ogg");
        Serial.println();
      }

  //musicPlayer.sineTest(0x44, 500);
  digit = 0;
 if(digitalRead(cradle) == LOW && fona.getCallStatus() == 0){       //handset off cradle and not receiving call
  for(byte x = 0; x < maxDigit+1; x++){          //erase any existing number
       tele_num_str1[x] = '\0';
      }


 // routine to get number from dial
      count = 0;
   while(digitalRead(cradle) == LOW){           // handset still off cradle


    while(digitalRead(46) == LOW){           // detect current pulse train being generated
      delay(20);                            // allow for bounce
     if(digitalRead(48) == HIGH){            // register a pulse
        digit++;
        while(digitalRead(48) == HIGH){      // wait for end of current pulse
        delay(20);                          // allow for bounce
        }
      }
    }

    if(digit != 0){                       //10 pulses represents zero
      if(digit == 10){
        digit = 0;
      }

     tele_num_str1[count] = digit + '0';   // change digit to character

      digit = 0;
      count++;
      if(count > 12){                     // limit no of digits to 12
        count = 12;
      }
    }

  if(digitalRead(44) == LOW){             // call switch pressed, dialling must be done

    tele_num_str1[count + 1]= '\0';
    Serial.print(tele_num_str1);
    Serial.println();

    break;                                //break out of while loop
  }

 }

     if(digitalRead(cradle) == LOW){            //handset still off cradle
        calling = true;
        flushSerial();
        Serial.print(F("Call #"));
        Serial.println();
        Serial.print(F("Calling "));
        Serial.println(tele_num_str1);
        if (!fona.callPhone(tele_num_str1)) {
          Serial.println(F("Failed"));
        } else {
          Serial.println(F("Sent!"));
        }
      }

        while(digitalRead(cradle) == LOW){      //wait for cradle switch to be pressed to hang up

        }
        delay(50);

       if(calling){                         // now hang up
        if (! fona.hangUp()) {
          Serial.println(F("Failed"));
        } else {
          Serial.println(F("HANG UP OK!"));
        }
        calling = false;
       }

  // flush input
  flushSerial();
  while (fona.available()) {
    Serial.write(fona.read());
  }
  }
}

//------------------------------------------------------------------------------------

void answercall(){

 if (! fona.pickUp()) {
          Serial.println(F("Failed"));
        } else {
          Serial.println(F("OK!"));
        }
while(digitalRead(cradle) == LOW){   // handset off cradle
}
// hang up!
        if (! fona.hangUp()) {
          Serial.println(F("Failed"));
        } else {
          Serial.println(F("OK!"));
        }


}

/////////////////////////////////////////////////////////////////////////////////////////

// Setting the volume is very simple! Just write the 6-bit
// volume to the i2c bus. That's it!
boolean setvolume(int8_t v) {
  // cant be higher than 63 or lower than 0
  if (v > 63) v = 63;
  if (v < 0) v = 0;

  Serial.print("Setting volume to ");
  Serial.println(v);
  Wire.beginTransmission(MAX9744_I2CADDR);
  Wire.write(v);
  if (Wire.endTransmission() == 0)
    return true;
  else
    return false;
}

/////////////////////////////////////////////////////////////////////////////////////////////

void num_fromfile(){

  for(byte x = 0; x < maxDigit + 1; x++){          //erase any existing number
       tele_num_str1[x] = '\0';
      }
      count = 0;


 musicPlayer.sineTest(0x44, 500);         // Make a tone to indicate FONA initialised
 delay(500);                              // these two lines are needed for some reason to stop
 musicPlayer.stopPlaying();               // the tone going on indefinitely till the next call to musicPlayer


 while(digitalRead(46) == HIGH){     //wait for dial to be operated
   if(digitalRead(cradle) == LOW){       //bailout if handset lifted before number selected
    break;
   }
 }
  delay(50);
 //get number of preset
  digit = 0;
  while(digitalRead(46) == LOW){           // detect current pulse train being generated
      delay(20);                             // allow for bounce
     if(digitalRead(48) == HIGH){            // register a pulse
        digit++;
        while(digitalRead(48) == HIGH){      // wait for end of current pulse
        delay(20);                          // allow for bounce
        }
      }
    }

    if(digit != 0){                         //10 pulses represents zero
      if(digit == 10){
        digit = 0;
      }
    }


 // .txt files contain number, .mp3 files contain name
 if(digit == 1){
    dialit("one.txt", "one.mp3");
      }
 if(digit == 2){
   dialit("two.txt", "two.mp3");
  }
  if(digit == 3){
    dialit("three.txt", "three.mp3");
   }
 if(digit == 4){
   dialit("four.txt", "four.mp3");
  }
  if(digit == 5){
    dialit("five.txt", "five.mp3");
   }

  //menu.mp3 contains list of names and index numbers
  if(digit == 0){
     if (! musicPlayer.playFullFile("menu.mp3")) {
        Serial.println("Could not open file ");
        Serial.print("menu.mp3");
        Serial.println();
      }

   }


  }



////////////////////////////////////////////////////////////////////////////////

void dialit(char num_file[], char txt_file[]){


myFile = SD.open(num_file);
 if(myFile){
  byte index = 0;
  while(myFile.available()){
    tele_num_str1[index] = myFile.read();
    index++;
  }
 }
 myFile.close();
 Serial.println(tele_num_str1);
 delay(200);

 if (! musicPlayer.playFullFile(txt_file)) {          //play name
        Serial.println("Could not open file ");
        Serial.print(txt_file);
        Serial.println();
      }
  musicPlayer.reset();

 while(digitalRead(cradle) == HIGH){
  //wait for handset to be lifted - that triggers dialling
 }
 delay(50);

 calling = true;
        flushSerial();
        Serial.print(F("Call #"));
        Serial.println();
        Serial.print(F("Calling "));
        Serial.println(tele_num_str1);
        if (!fona.callPhone(tele_num_str1)) {
          Serial.println(F("Failed"));
        } else {
          Serial.println(F("Sent!"));
        }

  while(digitalRead(cradle) == LOW){      //wait for cradle switch to be pressed to hang up

        }
        delay(50);
         byte x = 0;
       while(calling && x < 11){                // now hang up (have 11 goes at it!)

        if (! fona.hangUp()) {
          Serial.println(F("Failed"));
        } else {
          Serial.println(F("HANG UP OK!"));
          calling = false;
        }
        delay(300);
        x++;

       }

  // flush input
  flushSerial();
  while (fona.available()) {
    Serial.write(fona.read());
  }

}
```