#include "FS.h"

#include "SD.h"

#include "SPI.h"

#include <Keypad.h>

#include <WiFi.h>

#include "Audio.h"

const byte ROWS = 4;

const byte COLS = 4;

char hexaKeys[ROWS][COLS] = {

{ '1', '4', 's', 'e' },

{ '2', '5', 'n', 'u' },

{ '3', '6', 'C', 'c' },

{ '\*', 'b', '#', 'd' }

};

byte rowPins[ROWS] = { 35, 36, 37, 38 };

byte colPins[COLS] = { 39, 40, 41, 42 }; // 18-sck, 5-cs, 23-mosi, 19-miso for sd

Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);

#define SD\_CS 10

#define SPI\_MOSI 11

#define SPI\_MISO 13

#define SPI\_SCK 12

// Digital I/O used

#define I2S\_DOUT 6

#define I2S\_BCLK 5

#define I2S\_LRC 4

Audio audio;

struct audioMessage {

uint8\_t cmd;

const char\* txt;

uint32\_t value;

uint32\_t ret;

const char\* lang;

} audioTxMessage, audioRxMessage;

enum : uint8\_t { SET\_VOLUME,

GET\_VOLUME,

CONNECTTOHOST,

CONNECTTOSD,

CONNECTTOSPEECH,

GET\_AUDIOSTATUS };

QueueHandle\_t audioSetQueue = NULL;

QueueHandle\_t audioGetQueue = NULL;

void CreateQueues() {

audioSetQueue = xQueueCreate(10, sizeof(struct audioMessage));

audioGetQueue = xQueueCreate(10, sizeof(struct audioMessage));

}

struct language {

char userdata;

String kerneldata;

};

language eng[] = {

{ ' ', "E000000" },

{ '!', "E011101" },

{ '\"',"E000010" },

{ '#', "E001111" },

{ '$', "E110101" },

{ '%', "E100101" },

{ '&', "E111101" },

{ '\'',"E001000" },

{ '(', "E111011" },

{ ')', "E011111" },

{ '\*', "E100001" },

{ '+', "E001101" },

{ ',', "E000001" },

{ '-', "E001001" },

{ '.', "E000101" },

{ '/', "E001100" },

{ '0', "E001011" },

{ '1', "E010000" },

{ '2', "E011000" },

{ '3', "E010010" },

{ '4', "E010011" },

{ '5', "E010001" },

{ '6', "E011010" },

{ '7', "E011011" },

{ '8', "E011001" },

{ '9', "E001010" },

{ ':', "E100011" },

{ ';', "E000011" },

{ '<', "E110001" },

{ '=', "E111111" },

{ '>', "E001110" },

{ '?', "E100111" },

{ '@', "E000100" },

{ 'A', "E100000" },

{ 'B', "E110000" },

{ 'C', "E100100" },

{ 'D', "E100110" },

{ 'E', "E100010" },

{ 'F', "E110100" },

{ 'G', "E110110" },

{ 'H', "E110010" },

{ 'I', "E010100" },

{ 'J', "E010110" },

{ 'K', "E101000" },

{ 'L', "E111000" },

{ 'M', "E101100" },

{ 'N', "E101010" },

{ 'P', "E111100" },

{ 'Q', "E111110" },

{ 'R', "E111010" },

{ 'S', "E011100" },

{ 'T', "E011110" },

{ 'U', "E101001" },

{ 'V', "E111001" },

{ 'W', "E010111" },

{ 'X', "E101101" },

{ 'Y', "E101111" },

{ 'Z', "E101011" },

{ '[', "E010101" },

{ '\\',"E110011" },

{ ']', "E110111" },

{ '^', "E000110" },

{ '\_', "E000111" }

};

#define root\_menu\_cnt 7

#define write\_f\_cnt 3

#define practice\_writing\_cnt 3

#define learn\_writing\_cnt 3

#define read\_f\_cnt 3

#define practice\_reading\_cnt 3

#define learn\_reading\_cnt 3

#define audio\_book\_cnt 3

#define calculator\_cnt 4

#define wireless\_cnt 2

#define settings\_cnt 5

String set\_data = "E000000", s;

String dataa, buff, phone\_number;

const char\* ssid = "rakib";

const char\* password = "rakib@2023";

bool capslock = false;

bool numlock = false;

bool btn\_cancel\_isdown = false;

bool btn\_up\_isdown = false;

bool btn\_down\_isdown = false;

bool btn\_accept\_isdown = false;

bool btn\_backspace\_isdown = false;

bool ctrl= false;

enum pagetype { root\_menu,

write\_f,

read\_f,

audio\_book,

calculator,

wireless,

sim,

settings,

doc\_writing,

practice\_writing,

learn\_writing,

practice\_alphabet\_writing,

practice\_number\_writing,

practice\_specialcharecter\_writing,

learn\_alphabet\_writing,

learn\_number\_writing,

learn\_specialcharecter\_writing,

doc\_reading,

practice\_reading,

learn\_reading,

practice\_alphabet\_reading,

practice\_number\_reading,

practice\_specialcharecter\_reading,

learn\_alphabet\_reading,

learn\_number\_reading,

learn\_specialcharecter\_reading,

bengali\_book,

english\_book,

math\_book,

addition,

subtraction,

multiplication,

division,

wifi,

bluetooth,

call,

message,

language,

voice\_output,

braille\_output,

touch\_pen,

power };

enum pagetype c\_page = root\_menu;

void setup() {

pinMode(SD\_CS, OUTPUT);

digitalWrite(SD\_CS, HIGH);

SPI.begin(SPI\_SCK, SPI\_MISO, SPI\_MOSI);

SPI.setFrequency(1000000);

Serial.begin(115200);

WiFi.begin(ssid, password);

SD.begin(SD\_CS);

if (!SD.begin()) {

Serial.println("Card Mount Failed");

return;

}

uint8\_t cardType = SD.cardType();

if (cardType == CARD\_NONE) {

Serial.println("No SD card attached");

return;

}

if (cardType == CARD\_MMC) {

Serial.println("MMC");

} else if (cardType == CARD\_SD) {

Serial.println("SDSC");

} else if (cardType == CARD\_SDHC) {

Serial.println("SDHC");

} else {

Serial.println("UNKNOWN");

}

// writeFile(SD, "/text.txt", " ");

if (!SD.exists("/text.txt")) {

Serial.println("new file");

writeFile(SD, "/text.txt", " ");

}

Serial.printf("Total space: %lluMB\n", SD.totalBytes() / (1024 \* 1024));

Serial.printf("Used space: %lluMB\n", SD.usedBytes() / (1024 \* 1024));

audioInit();

log\_i("current volume is: %d", audioGetVolume());

}

void loop() {

switch (c\_page) {

//root

case root\_menu: root\_menu\_page(); break;

case write\_f: write\_f\_page(); break;

case read\_f: read\_f\_page(); break;

case audio\_book: audio\_book\_page(); break;

case calculator: calculator\_page(); break;

case wireless: wireless\_page(); break;

case sim: sim\_page(); break;

case settings: settings\_page(); break;

//write

case doc\_writing: doc\_writing\_page(); break; //write to text file

case practice\_writing: practice\_writing\_page(); break;

case learn\_writing: learn\_writing\_page(); break;

//practice write

case practice\_alphabet\_writing: practice\_alphabet\_writing\_page(); break;

case practice\_number\_writing: practice\_number\_writing\_page(); break;

case practice\_specialcharecter\_writing: practice\_specialcharecter\_writing\_page(); break;

//learn write

case learn\_alphabet\_writing: learn\_alphabet\_writing\_page(); break;

case learn\_number\_writing: learn\_number\_writing\_page(); break;

case learn\_specialcharecter\_writing: learn\_specialcharecter\_writing\_page(); break;

//read

case doc\_reading: doc\_reading\_page(); break; //read to text file

case practice\_reading: practice\_reading\_page(); break;

case learn\_reading: learn\_reading\_page(); break;

//practice read

case practice\_alphabet\_reading: practice\_alphabet\_reading\_page(); break;

case practice\_number\_reading: practice\_number\_reading\_page(); break;

case practice\_specialcharecter\_reading: practice\_specialcharecter\_reading\_page(); break;

//learn read

case learn\_alphabet\_reading: learn\_alphabet\_reading\_page(); break;

case learn\_number\_reading: learn\_number\_reading\_page(); break;

case learn\_specialcharecter\_reading: learn\_specialcharecter\_reading\_page(); break;

//audio book

case bengali\_book: bengali\_book\_page(); break;

case english\_book: english\_book\_page(); break;

case math\_book: math\_book\_page(); break;

//calculator

case addition: addition\_page(); break;

case subtraction: subtraction\_page(); break;

case multiplication: multiplication\_page(); break;

case division: division\_page(); break;

//wireless

case wifi: wifi\_page(); break;

case bluetooth: bluetooth\_page(); break;

//sim

case call: call\_page(); break;

case message: message\_page(); break;

//settings

case language: language\_page(); break;

case voice\_output: voice\_output\_page(); break;

case braille\_output: braille\_output\_page(); break;

case touch\_pen: touch\_pen\_page(); break;

case power: power\_page(); break;

}

}

//root menu

void root\_menu\_page(void) {

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" welcome to root menu"));

//audioConnecttoSD("voice\_over/root\_menu.mp3");

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("write"));

print\_selected(2, sub\_pos);

Serial.println(F("read"));

print\_selected(3, sub\_pos);

Serial.println(F("audio book"));

print\_selected(4, sub\_pos);

Serial.println(F("calculator"));

print\_selected(5, sub\_pos);

Serial.println(F("wireless"));

print\_selected(6, sub\_pos);

Serial.println(F("sim"));

print\_selected(7, sub\_pos);

Serial.println(F("settings"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/write.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/read.mp3");

} else if (menu\_selected(3, sub\_pos)) {

audioConnecttoSD("C/menu/eng/audio\_book.mp3");

}

if (menu\_selected(4, sub\_pos)) {

audioConnecttoSD("C/menu/eng/calculator.mp3");

} else if (menu\_selected(5, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/wireless.mp3");

} else if (menu\_selected(6, sub\_pos)) {

audioConnecttoSD("C/menu/eng/sim.mp3");

} else if (menu\_selected(7, sub\_pos)) {

audioConnecttoSD("C/menu/eng/settings.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == root\_menu\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = root\_menu\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = write\_f; return;

case 2: c\_page = read\_f; return;

case 3: c\_page = audio\_book; return;

case 4: c\_page = calculator; return;

case 5: c\_page = wireless; return;

case 6: c\_page = sim; return;

case 7: c\_page = settings; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//write section

void write\_f\_page(void) {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" write "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("doc writing"));

print\_selected(2, sub\_pos);

Serial.println(F("practice writing"));

print\_selected(3, sub\_pos);

Serial.println(F("learn writing"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/document\_writing.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/practice\_writing.mp3");

} else if (menu\_selected(3, sub\_pos)) {

audioConnecttoSD("C/menu/eng/learn\_writing.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == write\_f\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = write\_f\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = root\_menu;

return;

;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = doc\_writing; return;

case 2: c\_page = practice\_writing; return;

case 3: c\_page = learn\_writing; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

// doc writing section

void doc\_writing\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" doc writing "));

print\_divider();

print\_selected(1, 1);

Serial.println(F("write to text file"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_accept\_isdown) {

String st = braille\_input();

char udata = kerneldata\_to\_userdata(st);

Serial.println(udata);

if (numlock && numlock\_f(udata)) appendFile(SD, "/text.txt", (String)(udata));

if (!numlock) {

if (!capslock && (int)udata > 64 && (int)udata < 92) {

udata = (int)udata + 32;

}

appendFile(SD, "/text.txt", (String)(udata));

}

btn\_accept\_isdown = false;

}

if (btn\_backspace\_isdown) {

read\_for\_backspace(SD, "/text.txt");

int length = s.length();

s[length - 1] = '\0';

Serial.println(s);

writeFile(SD, "/text.txt", s);

btn\_backspace\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = write\_f;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//practice writing section

void practice\_writing\_page(void) {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" practice writing "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("practice alphabet writing"));

print\_selected(2, sub\_pos);

Serial.println(F("practice number writing"));

print\_selected(3, sub\_pos);

Serial.println(F("practice special charecter writing"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/practice\_alphabet\_writing.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/practice\_number\_writing.mp3");

} else if (menu\_selected(3, sub\_pos)) {

audioConnecttoSD("C/menu/eng/practice\_special\_character\_writing.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == practice\_writing\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = practice\_writing\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = write\_f;

return;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = practice\_alphabet\_writing; return;

case 2: c\_page = practice\_number\_writing; return;

case 3: c\_page = practice\_specialcharecter\_writing; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//practice alphabet writing

void practice\_alphabet\_writing\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("practice alphabet writing"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = practice\_writing;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//practice number writing

void practice\_number\_writing\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("practice number writing"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = practice\_writing;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//practice special charecter writing

void practice\_specialcharecter\_writing\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("practice special charecter writing"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = practice\_writing;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//learn writing section

void learn\_writing\_page(void) {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" learn writing "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("learn alphabet writing"));

print\_selected(2, sub\_pos);

Serial.println(F("learn number writing"));

print\_selected(3, sub\_pos);

Serial.println(F("learn special charecter writing"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/learn\_alphabet\_writing.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/learn\_number\_writing.mp3");

} else if (menu\_selected(3, sub\_pos)) {

audioConnecttoSD("C/menu/eng/learn\_special\_character\_writing.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == learn\_writing\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = learn\_writing\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = write\_f;

return;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = learn\_alphabet\_writing; return;

case 2: c\_page = learn\_number\_writing; return;

case 3: c\_page = learn\_specialcharecter\_writing; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//learn alphabet writing

void learn\_alphabet\_writing\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("learn alphabet writing"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = learn\_writing;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//learn number writing

void learn\_number\_writing\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("learn number writing"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = learn\_writing;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//learn special charecter writing

void learn\_specialcharecter\_writing\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("learn special charecter writing"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = learn\_writing;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//read

void read\_f\_page(void) {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("reading "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("doc reading"));

print\_selected(2, sub\_pos);

Serial.println(F("practice reading"));

print\_selected(3, sub\_pos);

Serial.println(F("learn reading"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/document\_reading.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/practice\_reading.mp3");

} else if (menu\_selected(3, sub\_pos)) {

audioConnecttoSD("C/menu/eng/learn\_reading.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == read\_f\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = read\_f\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = root\_menu;

return;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = doc\_reading; return;

case 2: c\_page = practice\_reading; return;

case 3: c\_page = learn\_reading; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

// doc reading section

void doc\_reading\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" doc reading "));

print\_divider();

print\_selected(1, 1);

Serial.println(F("read to text file"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = read\_f;

return;

}

if (btn\_accept\_isdown) {

readFile(SD, "/text.txt");

btn\_accept\_isdown = false;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//practice reading section

void practice\_reading\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" practice reading "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("practice alphabet reading"));

print\_selected(2, sub\_pos);

Serial.println(F("practice number reading"));

print\_selected(3, sub\_pos);

Serial.println(F("practice special charecter reading"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/practice\_alphabet\_reading.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/practice\_number\_reading.mp3");

} else if (menu\_selected(3, sub\_pos)) {

audioConnecttoSD("C/menu/eng/practice\_special\_character\_reading.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == practice\_reading\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = practice\_reading\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = read\_f;

return;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = practice\_alphabet\_reading; return;

case 2: c\_page = practice\_number\_reading; return;

case 3: c\_page = practice\_specialcharecter\_reading; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//practice alphabet reading

void practice\_alphabet\_reading\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("practice alphabet reading"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = practice\_reading;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//practice number reading

void practice\_number\_reading\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("practice number reading"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = practice\_reading;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//practice special charecter reading

void practice\_specialcharecter\_reading\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("practice special charecter reading"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = practice\_reading;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//learn reading section

void learn\_reading\_page(void) {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" learn reading "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("learn alphabet reading"));

print\_selected(2, sub\_pos);

Serial.println(F("learn number reading"));

print\_selected(3, sub\_pos);

Serial.println(F("learn special charecter reading"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/learn\_alphabet\_reading.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/learn\_number\_reading.mp3");

} else if (menu\_selected(3, sub\_pos)) {

audioConnecttoSD("C/menu/eng/learn\_special\_character\_reading.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == learn\_reading\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = learn\_reading\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = read\_f;

return;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = learn\_alphabet\_reading; return;

case 2: c\_page = learn\_number\_reading; return;

case 3: c\_page = learn\_specialcharecter\_reading; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//learn alphabet reading

void learn\_alphabet\_reading\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

const char\* ch;

uint8\_t sub\_pos = 65;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("learn alphabet reading"));

Serial.println();

print\_divider();

}

keypress\_detect();

if(btn\_accept\_isdown)

{

String str\_data=(String)((char)(sub\_pos));

String data=userdata\_to\_kerneldata(str\_data.c\_str());

String bi\_str=data.substring(1);

String ch\_name = "C/en/alp/" + str\_data + ".mp3";

ch = ch\_name.c\_str();

audioConnecttoSD(ch);

Serial.println(bi\_str);

//disp.writeBinStr(bi\_str);

btn\_accept\_isdown=false;

}

if (btn\_down\_isdown) {

if (sub\_pos == 90) {

sub\_pos = 65;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 65) {

sub\_pos = 90;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = learn\_reading;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//learn number reading

void learn\_number\_reading\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("learn number reading"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = learn\_reading;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//learn special charecter reading

void learn\_specialcharecter\_reading\_page(void) {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

print\_divider();

print\_selected(1, 1);

Serial.println(F("learn special charecter reading"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = learn\_reading;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//audio\_book

void audio\_book\_page(void) {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("listen audio book "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("bengali book"));

print\_selected(2, sub\_pos);

Serial.println(F("english book"));

print\_selected(3, sub\_pos);

Serial.println(F("math book"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/bengali\_book.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/english\_book.mp3");

} else if (menu\_selected(3, sub\_pos)) {

audioConnecttoSD("C/menu/eng/math\_book.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == audio\_book\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = audio\_book\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = root\_menu;

return;

;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = bengali\_book; return;

case 2: c\_page = english\_book; return;

case 3: c\_page = math\_book; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void bengali\_book\_page() {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" bengali book"));

print\_divider();

print\_selected(1, 1);

Serial.println(F("play bengali.wav"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = audio\_book;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void english\_book\_page() {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

bool update\_display = true;

int sub\_pos = 1;

uint32\_t loopstart;

const char\* filename;

const char\* page\_num;

audioConnecttoSD("C/menu/eng/play\_english\_for\_kids.mp3");

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" english book"));

print\_divider();

print\_selected(1, 1);

Serial.println(F("play English for kids"));

Serial.println();

print\_divider();

}

String page\_name = "C/menu/eng/page" + (String)(sub\_pos) + ".mp3";

page\_num = page\_name.c\_str();

audioConnecttoSD(page\_num);

keypress\_detect();

if (btn\_cancel\_isdown) {

audioConnecttoSD("C/menu/eng/stop.mp3");

c\_page = audio\_book;

return;

}

if (btn\_accept\_isdown) {

String fname = "D/books/english\_ for\_kids/" + (String)(sub\_pos) + ".mp3";

filename = fname.c\_str();

audioConnecttoSD(filename);

btn\_accept\_isdown = false;

}

if (btn\_down\_isdown) {

if (sub\_pos == 4) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = 4;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void math\_book\_page() {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" math book"));

print\_divider();

print\_selected(1, 1);

Serial.println(F("play math.wav"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = audio\_book;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void calculator\_page(void) {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome to calculator "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("addition"));

print\_selected(2, sub\_pos);

Serial.println(F("subtraction"));

print\_selected(3, sub\_pos);

Serial.println(F("multiplication"));

print\_selected(4, sub\_pos);

Serial.println(F("divison"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/addition.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/subtraction.mp3");

} else if (menu\_selected(3, sub\_pos)) {

audioConnecttoSD("C/menu/eng/multiplication.mp3");

} else if (menu\_selected(4, sub\_pos)) {

audioConnecttoSD("C/menu/eng/division.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == calculator\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = calculator\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = root\_menu;

return;

;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = addition; return;

case 2: c\_page = subtraction; return;

case 3: c\_page = multiplication; return;

case 4: c\_page = division; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void addition\_page() {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" addition"));

print\_divider();

print\_selected(1, 1);

Serial.println(F("A+B"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = calculator;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void subtraction\_page() {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" subtraction"));

print\_divider();

print\_selected(1, 1);

Serial.println(F("A-B"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = calculator;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void multiplication\_page() {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" multiplication"));

print\_divider();

print\_selected(1, 1);

Serial.println(F("A\*B"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = calculator;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void division\_page() {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" division"));

print\_divider();

print\_selected(1, 1);

Serial.println(F("A/B"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = calculator;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void wireless\_page(void) {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome to wireless section "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("wifi"));

print\_selected(2, sub\_pos);

Serial.println(F("bluetooth"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/wifi.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/bluetooth.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == wireless\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = wireless\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = root\_menu;

return;

;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = wifi; return;

case 2: c\_page = bluetooth; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void wifi\_page() {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome to wifi section "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("enter wifi ssid"));

print\_selected(2, sub\_pos);

Serial.println(F("enter wifi password"));

print\_selected(3, sub\_pos);

Serial.println(F("press enter to set credential"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/wifi\_ssid.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/wifi\_password.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == 3) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = 3;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = wireless;

return;

;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1:

Serial.println(F(" enter ssid"));

while (Serial.available() == 0) {

String ssid = Serial.readString();

if (ssid != NULL) {

Serial.println(ssid);

return;

}

if (btn\_cancel\_isdown) break;

}

case 2:

Serial.println(F(" enter password"));

case 3:

Serial.println(F(" \*set credential"));

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void bluetooth\_page() {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome to bluetooth section "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("turn on bluetooth"));

print\_selected(2, sub\_pos);

Serial.println(F("turn off bluetooth"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/bluetooth\_on.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/bluetooth\_off.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == 2) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = 2;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = wireless;

return;

;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: Serial.println(F(" \*turned on bluetooth"));

case 2: Serial.println(F(" \*turned off bluetooth"));

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//sim section

void sim\_page(void) {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome to sim section "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("call"));

print\_selected(2, sub\_pos);

Serial.println(F("message"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/call.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/message.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == 2) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = 2;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = root\_menu;

return;

;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = call; return;

case 2: c\_page = message; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void call\_page() {

btn\_accept\_isdown = false;

btn\_cancel\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

const char\* num;

audioConnecttoSD("C/menu/eng/input\_number.mp3");

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" call"));

print\_divider();

print\_selected(1, 1);

Serial.println(F("call to x"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_accept\_isdown) {

numlock = true;

String st = braille\_input();

char digit = kerneldata\_to\_userdata(st);

if (numlock && numlock\_f(digit)) phone\_number += (String)(digit);

Serial.println(phone\_number);

String num\_name = "C/en/num/" + (String)(digit) + ".mp3";

num = num\_name.c\_str();

audioConnecttoSD(num);

btn\_accept\_isdown = false;

}

if (btn\_backspace\_isdown) {

int length = phone\_number.length();

phone\_number[length - 1] = '\0';

btn\_backspace\_isdown = false;

}

if(phone\_number.length() == 11)

audioConnecttoSD("C/menu/eng/make\_call.mp3");

if (ctrl&&btn\_accept\_isdown) {

make\_call(phone\_number);

btn\_accept\_isdown = false;

phone\_number = "";

btn\_accept\_isdown = false;

ctrl=false;

}

if (btn\_cancel\_isdown) {

c\_page = sim; return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void message\_page() {

btn\_cancel\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F(" message"));

print\_divider();

print\_selected(1, 1);

Serial.println(F("message to x"));

Serial.println();

print\_divider();

}

keypress\_detect();

if (btn\_cancel\_isdown) {

c\_page = sim;

return;

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//settings

void settings\_page(void) {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome to settings "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("language"));

print\_selected(2, sub\_pos);

Serial.println(F("voice output"));

print\_selected(3, sub\_pos);

Serial.println(F("braille output"));

print\_selected(4, sub\_pos);

Serial.println(F("touch pen"));

print\_selected(5, sub\_pos);

Serial.println(F("power"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/language.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/voice\_output.mp3");

} else if (menu\_selected(3, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/braille\_output.mp3");

} else if (menu\_selected(4, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/touch\_pen.mp3");

} else if (menu\_selected(5, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/power.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == settings\_cnt) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = settings\_cnt;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = root\_menu;

return;

;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: c\_page = language; return;

case 2: c\_page = voice\_output; return;

case 3: c\_page = braille\_output; return;

case 4: c\_page = touch\_pen; return;

case 5: c\_page = power; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//language

void language\_page() {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome to language section "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F("english language"));

print\_selected(2, sub\_pos);

Serial.println(F("bangla language"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/english\_language.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/bengali\_language.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == 2) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = 2;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = settings;

return;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: set\_data = "E000000"; return;

case 2: set\_data = "B000000"; return;

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//voice output

void voice\_output\_page() {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome to voice output section "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F(" voice output enable"));

print\_selected(2, sub\_pos);

Serial.println(F("voice output disable"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/voice\_enable.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/voice\_disable.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == 2) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = 2;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = settings;

return;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: Serial.println(F(" \*voice output enabled"));

case 2: Serial.println(F(" \*voice output disabled"));

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//braille output

void braille\_output\_page() {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome to braille output section "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F(" braille display output enable"));

print\_selected(2, sub\_pos);

Serial.println(F("braille display output disable"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/display\_enable.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/display\_disable.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == 2) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = 2;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = settings;

return;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: Serial.println(F(" \*braille output enabled"));

case 2: Serial.println(F(" \*braille output disabled"));

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//touch pen section

void touch\_pen\_page() {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome to touch pen section "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F(" touch pen enable"));

print\_selected(2, sub\_pos);

Serial.println(F("touch pen disable"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/touch\_pen\_enable.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/touch\_pen\_disable.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == 2) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = 2;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = settings;

return;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: Serial.println(F(" \*pen input enabled"));

case 2: Serial.println(F(" \*pen input disabled"));

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

//power

void power\_page() {

btn\_cancel\_isdown = false;

btn\_up\_isdown = false;

btn\_down\_isdown = false;

btn\_accept\_isdown = false;

bool update\_display = true;

uint32\_t loopstart;

uint8\_t sub\_pos = 1;

while (true) {

loopstart = millis();

if (update\_display) {

update\_display = false;

clear\_screen();

Serial.println(F("welcome power section "));

print\_divider();

print\_selected(1, sub\_pos);

Serial.println(F(" power off"));

print\_selected(2, sub\_pos);

Serial.println(F("restart"));

Serial.println();

Serial.println();

print\_divider();

if (menu\_selected(1, sub\_pos)) {

audioConnecttoSD("C/menu/eng/power\_off.mp3");

} else if (menu\_selected(2, sub\_pos)) //C/en/alp/A.mp3

{

audioConnecttoSD("C/menu/eng/restart.mp3");

}

}

keypress\_detect();

if (btn\_down\_isdown) {

if (sub\_pos == 2) {

sub\_pos = 1;

} else {

sub\_pos++;

}

update\_display = true;

btn\_down\_isdown = false;

}

if (btn\_up\_isdown) {

if (sub\_pos == 1) {

sub\_pos = 2;

} else {

sub\_pos--;

}

update\_display = true;

btn\_up\_isdown = false;

}

if (btn\_cancel\_isdown) {

c\_page = settings;

return;

}

if (btn\_accept\_isdown) {

switch (sub\_pos) {

case 1: Serial.println(F(" \*power off"));

case 2: ESP.restart();

}

}

while (millis() - loopstart < 25) {

delay(2);

}

}

}

void print\_selected(uint8\_t x, uint8\_t y) {

if (x == y) Serial.print(F("-> "));

else Serial.print(F(" "));

}

bool menu\_selected(uint8\_t x, uint8\_t y) {

if (x == y) return true;

else return false;

}

void clear\_screen() {

for (uint8\_t i = 0; i < 100; i++) Serial.println();

}

void print\_divider() {

for (uint8\_t i = 0; i < 20; i++) Serial.print("--");

Serial.println();

}

String braille\_input() {

String data = set\_data;

for (;;) {

if (customKeypad.getKeys()) {

for (int i = 0; i < LIST\_MAX; i++) // Scan the whole key list.

{

if (customKeypad.key[i].stateChanged) // Only find keys that have changed state.

{

switch (customKeypad.key[i].kstate) { // Report active key state : IDLE, PRESSED, HOLD, or RELEASED

case PRESSED:

char ch = customKeypad.key[i].kchar;

if (ch > 48 && ch < 55)

data[((String)ch).toInt()] = '1';

else if (customKeypad.key[i].kchar == 'e') {

return data;

}

break;

}

}

}

}

}

}

char kerneldata\_to\_userdata(String data) {

for (int i = 0; i < sizeof(eng) / sizeof(eng[0]); i++) {

if (data.compareTo(eng[i].kerneldata) == 0) {

//if(eng[i].userdata!="B000000")

return eng[i].userdata;

}

}

}

String userdata\_to\_kerneldata(String data) {

for (int i = 0; i < sizeof(eng) / sizeof(eng[0]); i++) {

if (data.compareTo((String)eng[i].userdata) == 0)

{

return eng[i].kerneldata;

}

}

}

void readFile(fs::FS& fs, const char\* path) {

Serial.printf("Reading file: %s\n", path);

File file = fs.open(path);

if (!file) {

Serial.println("Failed to open file for reading");

return;

}

Serial.print("Read from file: ");

s = ""; //audiostatus()

int cnt = 0;

while (file.available()) {

char charRead = file.read();

s += (String)charRead;

if (charRead == ' ') cnt++;

if (charRead == '.' || cnt == 25 || !file.available()) {

audioConnecttoSpeech(s.c\_str(), "en");

s = "";

cnt = 0;

while (audiostatus()) { Serial.println("audio playing"); }

}

}

Serial.println(s);

file.close();

}

void writeFile(fs::FS& fs, const char\* path, String message) {

Serial.printf("Writing file: %s\n", path);

File file = fs.open(path, FILE\_WRITE);

if (!file) {

Serial.println("Failed to open file for writing");

return;

}

if (file.print(message)) {

Serial.println("File written");

} else {

Serial.println("Write failed");

}

file.close();

}

void appendFile(fs::FS& fs, const char\* path, String message) {

File file = fs.open(path, FILE\_APPEND);

if (!file) {

Serial.println("Failed to open file for appending");

return;

}

if (file.print(message)) {

Serial.println("Message appended");

} else {

Serial.println("Append failed");

}

file.close();

}

void keypress\_detect() {

int cnt = 0;

char customKey = customKeypad.getKey();

if (customKey == 'e') {

btn\_accept\_isdown = true;

}

if (customKey == 'u') {

btn\_up\_isdown = true;

}

if (customKey == 'd') {

btn\_down\_isdown = true;

}

if (customKey == 'c') {

btn\_cancel\_isdown = true;

}

if (customKey == 'b' && c\_page == doc\_writing) {

Serial.println("backspace");

btn\_backspace\_isdown = true;

}

if (customKey == '\*') {

ctrl = true;

}

if (customKey == 'C') {

capslock = !capslock;

if (capslock) {

Serial.println("capslock on");

}

if (!capslock) {

Serial.println("capslock off");

}

}

if (customKey == 'n') {

numlock = !numlock;

if (numlock) {

Serial.println("numlock on");

}

if (!numlock) {

Serial.println("numlock off");

}

}

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// GSM

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void send\_sms(String phn\_num, String msg) {

Serial2.println("AT+CMGF=1");

waitForResponse();

Serial2.println("AT+CNMI=1,2,0,0,0");

waitForResponse();

String msg\_mode = "AT+CMGS=\"" + phn\_num + "\"\r";

Serial2.print(msg\_mode);

waitForResponse();

Serial2.print(msg);

Serial2.write(0x1A);

waitForResponse();

}

void make\_call(String phn\_num) {

phn\_num = "ATD" + phn\_num + ";";

Serial2.println(phn\_num);

delay(40000);

waitForResponse();

}

void waitForResponse() {

delay(1000);

while (Serial2.available()) {

buff = Serial2.readString();

Serial.print(buff);

}

//Serial2.read();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// numlock & backspace \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

bool numlock\_f(char udata) {

if (isDigit(udata))

return true;

else {

audioConnecttoSD("C/sounds/beep.mp3");

return false;

}

}

void read\_for\_backspace(fs::FS& fs, const char\* path) {

File file = fs.open(path);

Serial.print("Read from file: ");

s = "";

while (file.available()) {

char charRead = file.read();

s += (String)charRead;

//Serial.write(file.read());

}

Serial.println(s);

file.close();

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// A U D I O \_ T A S K \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

void audioTask(void\* parameter) {

CreateQueues();

if (!audioSetQueue || !audioGetQueue) {

log\_e("queues are not initialized");

while (true) { ; } // endless loop

}

struct audioMessage audioRxTaskMessage;

struct audioMessage audioTxTaskMessage;

audio.setPinout(I2S\_BCLK, I2S\_LRC, I2S\_DOUT);

audio.setVolume(20); // 0...21

while (true) {

if (xQueueReceive(audioSetQueue, &audioRxTaskMessage, 1) == pdPASS) {

if (audioRxTaskMessage.cmd == SET\_VOLUME) {

audioTxTaskMessage.cmd = SET\_VOLUME;

audio.setVolume(audioRxTaskMessage.value);

audioTxTaskMessage.ret = 1;

xQueueSend(audioGetQueue, &audioTxTaskMessage, portMAX\_DELAY);

} else if (audioRxTaskMessage.cmd == CONNECTTOHOST) {

audioTxTaskMessage.cmd = CONNECTTOHOST;

audioTxTaskMessage.ret = audio.connecttohost(audioRxTaskMessage.txt);

xQueueSend(audioGetQueue, &audioTxTaskMessage, portMAX\_DELAY);

} else if (audioRxTaskMessage.cmd == CONNECTTOSD) {

audioTxTaskMessage.cmd = CONNECTTOSD;

audioTxTaskMessage.ret = audio.connecttoSD(audioRxTaskMessage.txt);

xQueueSend(audioGetQueue, &audioTxTaskMessage, portMAX\_DELAY);

} else if (audioRxTaskMessage.cmd == GET\_VOLUME) {

audioTxTaskMessage.cmd = GET\_VOLUME;

audioTxTaskMessage.ret = audio.getVolume();

xQueueSend(audioGetQueue, &audioTxTaskMessage, portMAX\_DELAY);

} else if (audioRxTaskMessage.cmd == GET\_AUDIOSTATUS) {

audioTxTaskMessage.cmd = GET\_AUDIOSTATUS;

audioTxTaskMessage.ret = audio.isRunning();

xQueueSend(audioGetQueue, &audioTxTaskMessage, portMAX\_DELAY);

} else if (audioRxTaskMessage.cmd == CONNECTTOSPEECH) {

audioTxTaskMessage.cmd = CONNECTTOSPEECH;

audioTxTaskMessage.ret = audio.connecttospeech(audioRxTaskMessage.txt, audioRxTaskMessage.lang);

xQueueSend(audioGetQueue, &audioTxTaskMessage, portMAX\_DELAY);

} else {

log\_i("error");

}

}

audio.loop();

if (!audio.isRunning()) {

sleep(1);

}

}

}

void audioInit() {

xTaskCreatePinnedToCore(

audioTask, /\* Function to implement the task \*/

"audioplay", /\* Name of the task \*/

5000, /\* Stack size in words \*/

NULL, /\* Task input parameter \*/

2 | portPRIVILEGE\_BIT, /\* Priority of the task \*/

NULL, /\* Task handle. \*/

1 /\* Core where the task should run \*/

);

}

audioMessage transmitReceive(audioMessage msg) {

xQueueSend(audioSetQueue, &msg, portMAX\_DELAY);

if (xQueueReceive(audioGetQueue, &audioRxMessage, portMAX\_DELAY) == pdPASS) {

if (msg.cmd != audioRxMessage.cmd) {

log\_e("wrong reply from message queue");

}

}

return audioRxMessage;

}

void audioSetVolume(uint8\_t vol) {

audioTxMessage.cmd = SET\_VOLUME;

audioTxMessage.value = vol;

audioMessage RX = transmitReceive(audioTxMessage);

}

uint8\_t audioGetVolume() {

audioTxMessage.cmd = GET\_VOLUME;

audioMessage RX = transmitReceive(audioTxMessage);

return RX.ret;

}

bool audioConnecttohost(const char\* host) {

audioTxMessage.cmd = CONNECTTOHOST;

audioTxMessage.txt = host;

audioMessage RX = transmitReceive(audioTxMessage);

return RX.ret;

}

bool audioConnecttoSD(const char\* filename) {

audioTxMessage.cmd = CONNECTTOSD;

audioTxMessage.txt = filename;

audioMessage RX = transmitReceive(audioTxMessage);

return RX.ret;

}

void audio\_info(const char\* info) {

Serial.print("info ");

Serial.println(info);

}

bool audioConnecttoSpeech(const char\* speech, const char\* language) {

audioTxMessage.cmd = CONNECTTOSPEECH;

audioTxMessage.txt = speech;

audioTxMessage.lang = language;

audioMessage RX = transmitReceive(audioTxMessage);

return RX.ret;

}

bool audiostatus() {

audioTxMessage.cmd = GET\_AUDIOSTATUS;

audioMessage RX = transmitReceive(audioTxMessage);

return RX.ret;

}

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

// end audio portion \*

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*