Technická dokumentace

Vysvětlivky ke kódu semestrální práce do předmětu B4B35APO (léto 2016/2017)

Vypracoval: Miroslav Matocha

Osnova:

1. Kompilace a spuštění programu

Program se kompiluje pomocí *Makefile* přiloženého v archivu programu(příkaz *make*), ten odkazuje na všechny dodané knihovny a definuje defaultní značky pro kompilaci. Po úspěšném zkompilování se spustitelný soubor *main* nachází v kořenovém adresáři projektu.

Některé důležité hodnoty se nastavují již při kompilaci – výchozí jméno jednotky, ikonka apod. Tyto proměné a jejich inicializace se probírá v následující sekci.

2. Hlavní metoda a důležité konstanty

V hlavní části programu se nalézají tři metody. Jejich podrobný popis je uveden níže.

Konstnta MZAPO slouží k určení toho, jestli program běží na desce nebo na běžném PC.

void main()

Hlavní metoda, která spouští ostatní funkce z hlavní části a zároveň v ní běží hlavní smyčka programu.

Po zavolání funkce *init()* a kontroly jejího výstupu proběhne první prohledání oblasti pomocí funkce *getBroadcasters()* výsledek je uložen do proměnné *areaInfo*. Potom se spouští hlavní smyčka programu. Ta čte z paměti hodnoty potenciometrů, převádí je na směr otočení a podle něho upravuje index, který ukazuje na vybranou jednotku. Vše zaznamenává do obrázku, který jednou za cyklus vykreslí. Program zároveň vysílá svůj aktuální stav a jednou za RESARCH_TIME aktualizuje přehled jednotek, vzorek který nabírá má velikost SAMPLE_SIZE, vzorek při prvním spuštění má velikost START_SAMPLE. Ve smyčce se zároveň kontroluje stisk tlačítka, podle kterého program buď skončí, nebo zavolá funkci *runSettings()* s hodnotami právě vybrané jednotky.

void broadcastMe(int socket)

Funkce přijímá odkaz na aktuální socket a vysílá hodnoty uložené při kompilaci programu. Funkce hlídá pravidelný broadcast podle konstanty CAST_TIME.

int init()

Funkce se stará o inicializaci hodnot jednotky jako je text, nebo ikona. Zároveň ustanovuje síťový socket a pokud běží na desce(MZAPO) uloží odkaz na adresy ovládacích prvků a obrazovky. Funkce vrací 1 nebo 0, které značí úspěšnost inicializace.

void runSettings(char* ip, unsigned char* mem_base, unsigned char* lcd_base, int socket, InfoMessage* message)

Funkce která dostává z hlavní smyčky odkaz na socket, adresy ovládacích prvků a poté adresu a poslední přijatou zprávu vybrané jednotky. Podle stisků tlačítek nastavuje řídící proměné, která mají vliv na mód odesílání zprávy nebo na měněné hodnoty. Konkrétně podle proměné *state* řídí formát odesílané zpravy(Increment, Direct Set, ...viz Uživatelská příručka). *flag* potom určuje, jestli hodnota čtěná z potenciometrů ovlivňuje hodnoty osvětlení stropu nebo zdí. Mód, hodnoty a graf funkce pravidelně zapisuje do obrázku a zobrazuje na LCD displeji.

3. Popis jednotlivých knihoven

Knihovny jsou členěné podle fuknčního obsahu a jejich určení by měl sdělovat název. Podle konvence jsou hlavičky a implementace oddělené.

3.1 graphics.h

Knihovna, která implementuje vykreslování jednotlivých obrazovek a složitějších grafických celků.

Image* createMenuScreen(AreaInfo* area, int index)

Funkce přijímá struktura, která obsahuje informace o všech uzlech v oblasti a index právě vybraného uzlu. Pro každý uzel do obrázku zapíše jméno, ikonku a aktuální hodnoty. K vybrané jednotce přikreslý šipku a strukturu s obrázkem vrátí.

Image* createDetailScreen(char* ip, int index, unsigned char* wallsRGB, unsigned char* ceilingRGB, InfoMessage* message)

Funkce, která přijímá informace o právě zobrazované jednotce a zapisuje je do obrázku. Mezi ně patří jméno jednotky získané ze zprávy a její IP. Dále přijímá odkaz na pole hodnot pro stěny a strop, tyto hodnoty zapisuje do grafu. Vrací zapsaný obraz.

Image* createResearchScreen()

Fukce vrací strukturu *Image* do které zapíše text, který upozorňuje na úvodní hledání.

Image* paintIcon(Image* img, int16_t* rgb, int x, int y, int posx, int posy)

Funkce která vykreslí ikonu do obrazu na který dostane odkaz. Přijímá pole reprezentující ikonu, pozici ikony a její rozměry.

Image* paintGraph(Image* img, int posx, int posy, unsigned char* values)

Funkce vykresluje do obrazu graf na vybrané pozici. Hodnoty jsou také předávané jako odkaz na jejich pole.

Image* paintBar(Image* img, int posx, int posy, int x, int y, uint32_t rgb)

Funkce vykresluje do obrazu obdelník určené barvy. Parametry určují jeho rozměry a pozici.

3.2 intercom.h

Knihovna implementuje pokročilejší prvky komunikačního protokolu. Prohledávání oblasti, odesílání zpráv a podobně.

struct AreaInfo

Struktura obsahuje informace o okolí aktivní jednotky. Adresy jednotlivých uzlů, jejich aktuální zprávy a počet unikátních vysílačů.

int broadcastInfo(int socket, unsigned char* walls, unsigned char* ceiling, char* text, int16_t* image)

Funkce přijímá informace o jednotce v podobě jednotlivých hodnot, Ty převede na strukturu *InfoMessage,* kterou převede do network-order a pokusí se ji odvysílat, vrací jedničku nebo nulu v závislosti na jejím úspěchu.

int sendEdit(int socket, unsigned char* walls, unsigned char* ceiling, char* ip, int type)

Funkce přijímá informace o úpravě, nebo nastavení jednotlivých hodnot a typu zprávy. Ty převede na strukturu *EditMessage,* kterou převede do network-order a pokusí se ji odvysílat na příslušnou adresu, vrací jedničku nebo nulu v závislosti na jejím úspěchu.

AreaInfo* getBroadcasters(int socket, int numOfMessages)

Funkce si naalokuje místo na určený počet zpráv, ty potom přijme a vytřídí podle unikátních jmen. Následně zaalokuje strukturu *AreaInfo* a zjištěné údaje do ní překopíruje. Vrací strukturu s informací o aktuální oblasti.

AreaInfo* sortAreaByName(AreaInfo* area);

Vrací strukturu, která má stejné vstupy jako předaná, avšak setříděné podle jména.

3.3 messages.h

Knihovna definuje především formáty zpráv, poté soubor funkcí pro jejich zobrazování a převedení jednotlivých polí na správné pořadí bytů

struct InfoMessage

Zpráva obsahující obecné informace o jednotce, její ikonu, jméno a jednotlivé hodnoty osvětlení

struct EditMessage

Struktura pro odeslání zprávy s úpravou nebo nastavením jednotlivých položek. Určuje se podle připojené hlavičky.

struct MessageHead

Hlavička společná pro všechny tipy zpráv. Obsahuje typ protokolu, zprávy a kontolní číslo ALC1.

Knihovna poté obsahuje tři soubory funkcí. První pro tvoření samotných struktur z jednotlivých údajů, druhou pro tisk přijatých nebo odesílaných struktur a poslední pro převod struktur do odpovídajícího pořadí bytů. Implementace struktur je samovysvětlující, proto bych ji nerad věnoval prostor v tomto dokumentu.

3.4 netcom.h

Knihovna, která implementuje komunikaci na základní úrovni, nemá informace o tom jaká data přenáší. Definuje SEND_PORT a BIND_PORT.

int initCommunication()

Knihovna která nastavuje jednotlivé hodnoty spojení a vrací odkaz na vytvořený socket(případně 0 při selhání).

void* receiveBytes(int SOCKET, int bytesNum, char* address)

Funkce která přijímá odkaz na socket, počet bytů, který má přijmout a pointer na lokaci, kam má umístit adresu odesílatele. Vrací odkaz na přijmuté data.

int sendBytes(int SOCKET, void* bytes, char* ip, int length)

Funkce která přebírá odkaz na socket, posílaná data, adresu adresáta a počet bytů, které má přenést. Vrací 1 nebo 0 v závislosti na úspěchu poslání.

int broadcast(int SOCKET, void* bytes, int length)

Funkčně stejná jako *sendBytes()* ale bez konkrétního adresáta – vysílá do celé sítě.

3.5 imgwrite.h

Knihovna která definuje strukturu pro uchovávání grafiky a jednotlivé metody pro zápis grafických primitiv.

struct Image

Struktura obsahuje informace o rozměru obrazu a pole hodnot jednotlivých pixelů.

void showScreen(char* filename, Image* img);

Zapisuje obraz do souboru s předaným jménem.

Image* createTextScreen(int posx, int posy, char* text)

Vrátí obrazovku s předaným textem na vybrané pozici. Barva textu definována jako TEXT_R/G/B.

Image* createBlankScreen(int r, int g, int b)

Vytvoří obrazovku jedné barvy. Konkrétně podle konstant BASE_R/G/B.

Image* writeText(Image* img, int x, int y, char* text)

Zapíše předaný text do obrázku na vybrané pozici.

Image* setPixel(Image* img, int x, int y, int r, int g, int b)

Image* paintPixel(Image* img, int x, int y, int16_t color);

Nastaví pixel v obrazu na předanou barvu.

Image* writeLetter(Image* img, int x, int y, unsigned char c)

Zapíše písmeno na stanovenou pozici. Více o implementaci v sekci Third-party knihovny.

3.6 utils.h

Knihovna zvládá často používané konverze. A jiné základní funkce.

uint8_t* getValueIncrement(uint32_t old, uint32_t new)

Ze dvou hodnot vrátí pole delt o které se dané hodnoty liší. Používá konverzi z jedné 32bit hodnoty na pole o velikosti jednoho byte.

int getIndexIncrement(int lastVal, int curVal)

Ze dvou hodnot vrátí směr o který se líší. Počítá s overflow.

int* getUnique(char** ips, int length)

Z pole stringů vybere ty unikátní. Vrací pole 0 a 1, které značí unikátní stringy.

uint32_t charToNumRGB(unsigned char* RGB)

unsigned char* numToCharRGB(uint32_t RGB)

Konverze hodnot RGB.

3.7 control.h

Knihovna k čtení ovládacích prvků desky. Slouží k inicializaci adres potenciometrů, a čtení jejich hodnot, případně stlačení. Chování a parametry metod popisuje hlavičkový soubor.

3.8 screen.h

Obdobná knihovna jako *control.h* sloužící pro inicializaci obrazovky a zápis na ní, obsahuje jednu konverzní metodu. Implementace je znovu jednoduchá a využívá hlavně knihovny třetích stran.

4. Third-party knihovny

V projektu jsem použil několik knihoven třetích stran, hlavně pro definici grafických primitiv a komunikaci s hardwarem jednotky.

mzapo_parlcd.h

Knihovna sloužící k přístupu a zápis na adresy kontrolující jednotlivé prvky obrazovky.

mzapo_phys.h

Knihovna pro práci s pamětí, mapování oblastí apod.

mzapo_regs.h

Definice adres jednotlivých registrů řídících jednotku.

Všechny tři tyto knihovny byli poskytnuty vyučujícími předmětu B4B35APO. Jsou dostupné na adrese: https://cw.fel.cvut.cz/wiki/courses/b35apo/start

font_types.h

Definuje strukturu fontu a poskytuje přístup ke konkrétnímu fontu.

font_rom8x16.h

Deklarace hodnot pixelů a ostatních informací o fontu.

Knihovny Microwindows/Nano-X library by Greg Haerr

5. Dodatek

Dohromady kód tvoří implementaci funkčnosti popsané v uživatelské příručce.

Aktuální verze k nalezení na https://github.com/Shade254/APO_lightcontrol