DOKUMENTÁCIÓ	
Tantárgy neve	Informatika II
Mérés száma	Nagy házi feladat
Mérés megnevezése	Soros posti forgalomgenerátor SLLCP protokollon keresztül
Hallgató neve	Fodor Attila
Hallgató NEPTUN kódja	GRXOFY
Kurzus	KVM-T4L-07
Mérésvezető	• Borsos Döníz • Dombora Sándor • Sándor Tamás

Tartalom

Feladatválasztás 2

Specifikáció 2

1. Fő elemei és részegységei a programnak 2
2. Felhasználói beviteli lehetőségek 3
3. Kimenetek 3
4. Hibakezelések 3
5. File típusok és adattípusaik 3

Programozói dokumentáció 4

1. Blokkdiagram 4
2. Projekt felépítése: 4
3. Szükséges programok, környezet 7
4. Felhasznált könyvtárak 7

Ábrajegyzék

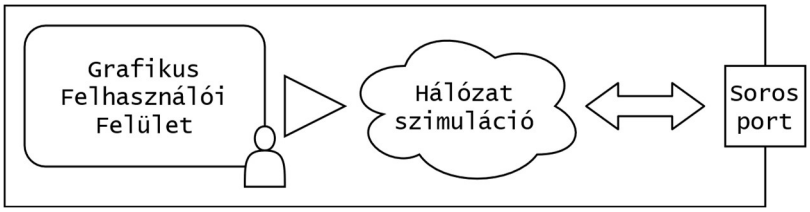
1. ábra - Megvalósítandó program elvi működése 2

2. ábra - Blokkdiagram 4

3. ábra - Modulok kapcsolata, a nyíl hegye mutat a beimportált modulra 7

Feladatválasztás

Egy fejlesztő eszköz a TDK munkámhoz. Képes legyen az őszi félévben beadott dolgozatomban specifikált protokollkészlet alapvető funkcióit ellátani. A különbség annyi, hogy a hálózati csomagokat soros porton keresztül fogadja és továbbítja. Továbbá képes legyen egyszerűbb CSV állományban tárolt CueList-et lejátszani és a megfelelő csomagba ágyazva soros porton keresztül továbbítani azt. A soros porton keresztül elérhető csomópont állapota és konfigurációs felülete egy külön ablakban jelenjen meg a tőle begyűjtött adatok alapján.



1. ábra - Megvalósítandó program elvi működése

Specifikáció

1. Fő elemei és részegységei a programnak

- **Grafikus ablak kezelő:** A felhasználó ennek az eredményét látja, ezen keresztül működteti a programot.
  - Konfigurációs felület
  - Hálózat felület
  - Konzol ablak
- **Soros port kezelő:** A futtató környezettől kéri le a soros portokat, egyikhez csatlakozik, annak a kommunikációját kezeli
  - Adó és vevő szálak
- **Protokoll kezelő:** SLLCPv1.0 és SLLCPv1.4 részleges, de elegendő implementációja.
- **Csomópont vezérlő:** SLLCP adminisztrációs funkciói érhetőek el egy külön, kifejezetten adott csomópontnak dedikált párbeszédpanelen keresztül.
  - Adatmegjelenítő felület
  - Hálózat felület
  - Adminisztrátori felület
- **CueList olvasó és lejátszó:** Fájlkezelés és szekvenciális végrehajtás egy párhuzamosan futó szál segítségével.

2. Felhasználói beviteli lehetőségek

Gombokkal, legördíthető listákkal és beviteli mezők segítségével konfigurálhatja be a programot. Hozzáfér a soros port kezeléséhez, beállíthatja a protokoll üzemmódját és hálózat felderítő funkcióját.

CueList-het tölthet be és elindíthatja lejátszását.

Színeket választhat a palettáról és a kiválasztott DMX-csatornákon kiküldheti azokat. Továbbá az első 8 DMX-csatorna keverő formájában is elérhető csúszó potenciométereken keresztül.

3. Kimenetek

A futó folyamatokról folyamatos visszajelzések szolgálnak időbélyeggel ellátva a konzol ablakon belül. A küldött és fogadott üzenetekről OpCode-dal és megnevezéssel ellátott állapot üzenetek jelennek meg.

4. Hibakezelések

Adott helyzetben nem elérhető funkciók meghívásának hatására figyelmeztető üzenet jelenik meg, mely indoklással is szolgál a megelőzött hibáról.

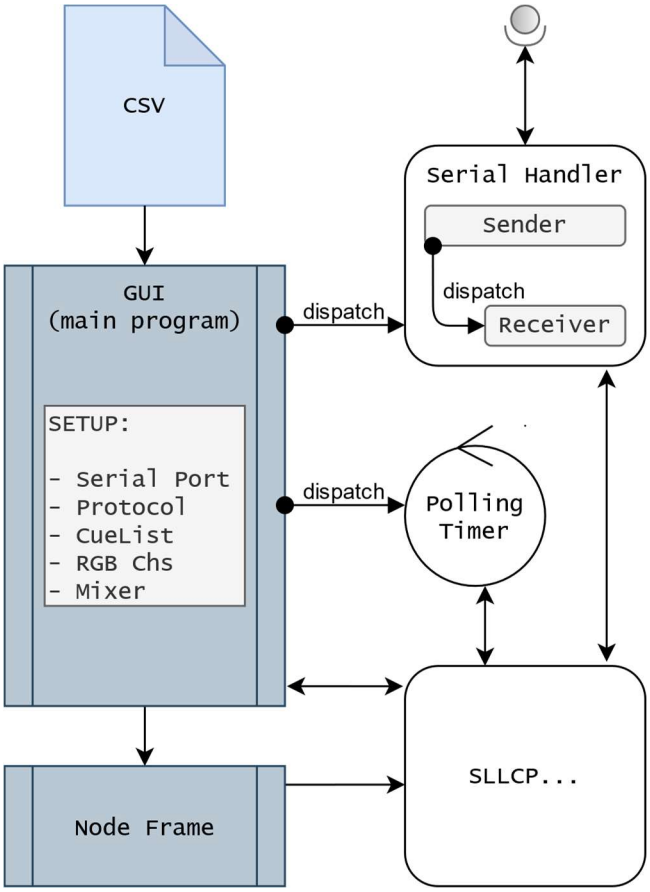
A beolvasott fájlok nem kerülnek validálásra végrehajtás előtt, így sérült állományból váratlan kimeneti csomag generálódhat.

5. File típusok és adattípusaik

CSV állományt olvas be a program, melynek 1. sorában szerepel a lista hossza, az érintett csatornák száma és a frame-k kiküldése közt eltelt idő. A 2. sorban felsorolásra kerülnek az érintett csatornák. A 3. sortól kezdődően a 2. sor sorrendje szerint minden frameben felsorolásra kerülnek a csatornák aktuális értékei. Annyi ilyen sornak kell lennie, mint, amennyit a fejléc sorában a lista hossza állított.

Programozói dokumentáció

1. Blokkdiagram



2. ábra - Blokkdiagram

2. Projekt felépítése:

Az elkészült program 9 modulból és 58 függvényből épül fel. Összesen 1136 sor forráskód.

`./venv/player.py`

Globális változók: `data, cue`

Osztály: `PlayerTimer(object)`

Tagfüggvényei:

- `__init__(function, *args, **kwargs)`
- `_run()`
- `_start()`
- `start(interval)`
- `stop()`

Függvények:

- `load()`
- `play()`

./venv/serial_receiver.py

Osztály: `SerialReceiver(threading.Thread)`

Tagfüggvényei:

- `__init__`
- `run`
- `stop`

./globals.py

Globális változók: `color, available_lengths, dmx_length, active_port, file_path, polling_timer, serial_node, dmxData, last_seen`

Függvény:

- `init_tk_globals()`

./gui.py

Globális változók: `main_window, txt, com_select, poll_nud, file_path, r_ch_nud, g_ch_nud, b_ch_nud, len_select, sc_values`

Függvények:

- `channel_set(ch, val)`
- `asd()`
- `send_dmx_data()`
- `collect_serial()`
- `color_picker()`
- `set_color()`
- `file_search()`
- `start_polling()`
- `on_closing()`
- `init_gui()`

./main.py

Osztály: `StdoutRedirect(object)`

Tagfüggvényei:

- `__init__(widget, tag)`
- `write(msg)`
- `flush()`
- `enable_timestamp()`

Függvények:

- `goodbye()`
- `signal_handler()`

./nodes.py

Globális változók: `node_window, len_select, right_txt, node_len, echo_opt, serial_node, capabilities`

Osztály: `SllcpNode`

Tagfüggvényei:

- `__init__(man, mod, dmx, wif, eth, dev, idi, ido, imi, imo, ilo, iso)`

- `print_to(widget)`

Függvények:

- `set_len()`
- `send_opt()`
- `send_restart()`
- `on_closing()`
- `print_capabs()`
- `open_node()`
- `open_node_window()`

./polling_timer.py

Osztály: `PollingTimer(object)`

Tagfüggvényei:

- `__init__(function, *args, **kwargs)`
- `_run()`
- `_start()`
- `start()`
- `stop()`

./serial_packet_sender.py

Globális változók: `ser, receiver`

Függvények:

- `init_serial()`
- `send_request_serial(oc)`
- `send_setter_serial(mode)`
- `multiline_serial(packet)`
- `send_pollreply_serial()`
- `send_capab_serial()`
- `send_datas_serial(length, intf, dmx_data)`
- `close_serial()`

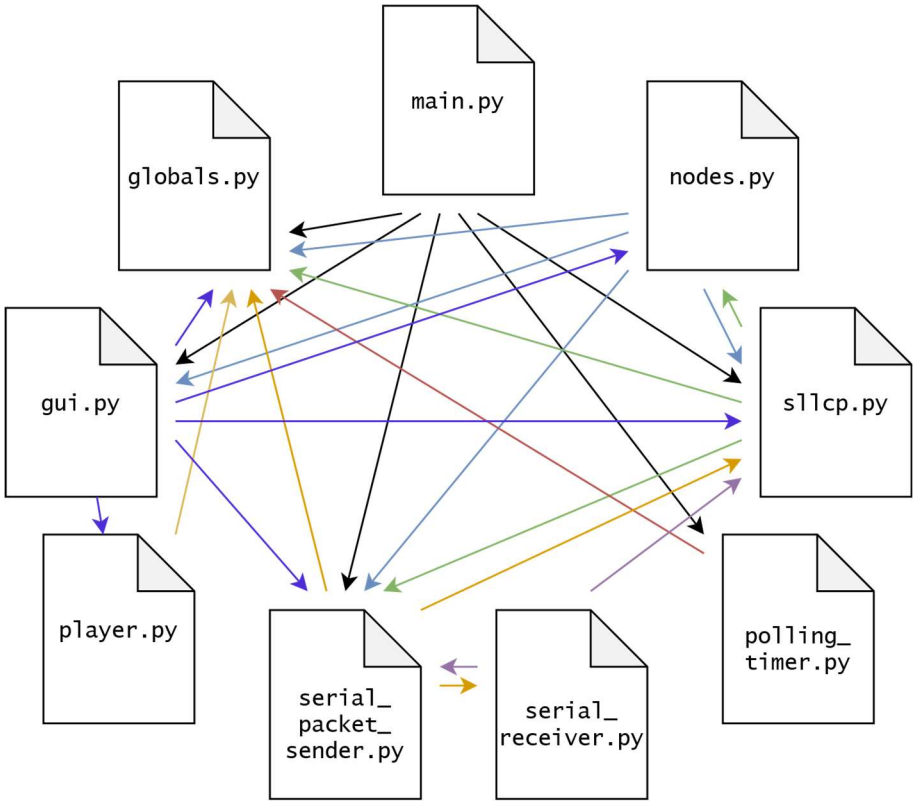
./sllcp.py

Globális változók: `version, enum_opcodes, enum_devices, header, packet, fwdIp, dmxLn`

Osztályok: `OpCodes, DevCodes`

Függvények:

- `filling_generator(amount)`
- `packet_reset(v)`
- `get_seq_id()`
- `len_to_opcode(s)`
- `simple_packet(op_code)`
- `set_mode_packet(mode_code)`
- `dmx_length_switch(oc)`
- `dmx_packet(length, intf, dmx_data)`
- `pollreply_packet()`
- `capability_packet()`
- `answer_received(incoming)`



3. ábra - Modulok kapcsolata, a nyíl hegye mutat a beimportált modulra

3. Szükséges programok, környezet

A program Python 3.9 veziójú keretrendszerre készült a PyCharm 2020.3.3 (Community Edition) Build #PC-203.7148.72 IDE használatával `venv` környezetben.

4. Felhasznált könyvtárak

A modulok a következő, a keretrendszerhez tartozó modulokat importálják be:

- csv
- datetime
- functools
- serial
- sys
- time
- tkinter
- threading

A program fejlesztése karakteres felületen kezdődött, az onnan megmaradt (de nem használt) funkciók miatt a következő modulok kerülnek még beimportálásra:

- argparse
- math
- signal

Ezek továbbra is a projekt részét képezik a tervezett tovább fejlesztések érdekében.