**Név: Schmelczer András Neptun kód: CVTNXD**

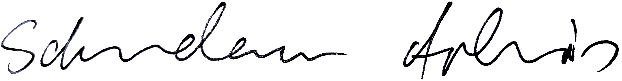
**Beadás dátuma: 2020. 05. 19.**

**Házi feladat címe: Szimulációs játék**

**Mikrokontrollerek alkalmazástechnikája házi feladat**

ATtiny85-ös mikrokontrollerrel szeretnék megvalósítani egy infrás távirányítóval irányítható játékot. A játék futásához nem kell emberi beavatkozás, magától is történnek benne események, viszont, ha fogunk egy távirányítót, akkor az egyik karaktert (a három közül) irányíthatjuk vele. A videójáték űrhajós tematikájú, amiben fejlődni az aszteroidák szétlövésével lehetséges.

A feladatot önállóan, meg nem engedett segítség nélkül oldottam meg:



……………………………….

aláírás

# Részletes specifikáció

A játék futásához nem kell emberi beavatkozás, magától is történnek benne események, viszont, amennyiben fogunk egy távirányítót, akkor az egyik karaktert irányíthatjuk vele. A rendszer 3.3 V-ról működtethető, fogyasztása nem haladja meg a 15 mW-ot.

Infrán keresztüli NEC protokollal történő kommunikációval hét különböző kérést küldhetünk. A kijelző fényereje állítható növekvő és csökkenő irányban, a képernyőn látható egyik karakter mozgatható négy irányban, illetve a képernyőn látható karakter késztethető cselekvésre.

A rendszerrel továbbá UART-on keresztül is lehet kommunikálni. Küldeni a távirányító által küldhető parancsok karakteres megfelelőjét lehet. Ezek az előző sorrenddel megegyezően a következők: ’+’, ’-’, ’w’, ’a’, ’s’, ’d’, ’ ’. Visszafele pedig egy képkocka kirajzolásának ideje érkezik pár másodperces időközönként.

Maga a játék az űrben játszódik. Ezen belül egy űrhajóban található a játékos által irányított karakter és pár másik, a gép által irányított karakter is. A karakterek az űrhajót irányíthatják (ehhez az irányítópulthoz kell menni, és megnyomni az akciógombot). Az űrhajóval lőni is lehet, ehhez egy másik irányítópulthoz kell menni. Az űrhajó egy részének a képernyőn mindig látszania kell, így azt nem lehet bármilyen távolra elirányítani.

Vizuálisan oldalnézetű képeket látunk. Az űrhajó kétszintes, ezek között, illetve ezeken belül tudnak a karakterek közlekedni. A két szintet egy létra köti össze, amelyen fel és lefele is lehet közlekedni. A lefele mozgásnak a létrán kívül nincs következménye. A felfele mozgást a létrán kívül végrehajtva egy ugró animációt láthatunk.

Ütközésnek számít az és csak az, amikor az űrhajó legalább egy sötét pixele fedésben van az aszteroida egy sötét pixelével. Az űrhajó ütközhet az aszteroidákkal, ilyenkor az energiája csökken. Az energia lövés hatására is csökken. Erről visszajelzést egy az űrhajó spritejába integrált csík szolgáltat. Ha a csík hosszúsága nulla, a játéknak vége, és egy effekt lejátszása után elölről kezdődik. Ezt elkerülendő a fedélzet megfelelő pontján kell az akciógombot megnyomni, ilyenkor újra feltöltődik a csík. Néha megjelennek aszteroidák, melyek szétlőhetők.

A spriteok animálhatók és az átlátszóság is biztosított. A háttérben csillagok mozognak véletlenszerű konfigurációkban. Utóbbiaknak nincs hatása a játékmenetre. Követelmény továbbá, hogy legalább 50 képkockát kirajzoljon másodpercenként legalább 64 x 32 pixeles felbontásban.

A játéknak figyelembe kell vennie az eddig felrobbantott aszteroidák számát, és ennek függvényében újabb funkciókat kell a játékos számára biztosítani. Ezek a funkciók sorrendben a következők: energia növelés lehetősége, lövés lehetősége, első segítő végül pedig a második segítő.

A segítők másik karaktereket jelölnek, akik egy többszintes szabályrendszert követve próbálják a játék előrehaladását segíteni. Minden funkció elérhető számukra, ami a játékos rendelkezésére is áll.

# Funkcionális blokkvázlat



## Részletesebben

A specifikáció alapján az előző oldalon található blokkvázlatot készítettem. Sokat gondolkoztam a megfelelő absztrakciós szinten. Végül amellett döntöttem, hogy a játéklogika fáradtságos részletezése helyett annak csak a specifikációban kiemelt fő részeit tüntetem fel. Az eszköz viselkedésére már ennyiből is kellően nagy pontossággal lehet következtetni, a pontos implementáció és felbontás pedig rendkívül sokféle lehet. Szerencsére a többi követelmény elég egyértelműen vetíthető funkcionális blokkokra.

Mivel parancsokat több forrásból is kaphat a rendszer így a komplexitás csökkentése végett, azokhoz közös feldolgozóegységet célszerű használni. A kommunikációs protokollok bonyolultsága miatt ezekhez kezelőket is célszerű alkalmazni, így a rendszer többi része könnyebben veheti igénybe őket. Ezeknek a kezelőknek időzítőre is szükségük lesz.

Többször említésre került a követelmények közt a véletlenszerűségre igénye, ezek megvalósításához szükséges egy véletlenszám-generátor is. Továbbá explicit nem szerepelt a perzisztens tároló szükségessége, viszont az animált sprite-ok tárolását ezzel megoldhatjuk.

Az üzleti logika felel a követelményrendszer nagyobb részének teljesítéséért. Ennek kiemelendő funkciói a játékállapotának és a játék objektumoknak a felügyelete. Emellett a többi komponens működésének összeegyeztetése is ezen egység feladata.

# Hardver szoftver szétválasztás

Ezek után a hardver szoftver szétválasztás többnyire egyértelmű lesz. A ki és bemenetek (infra, UART, LCD) természetesen hardveresen kerülnek megvalósításra. Az időzítők és tárhely pedig a mikrokontrollerek felépítéséből következően szintén hardveresen manifesztálódnak. Jó hír, hogy ezeken kívül minden funkció megvalósítható szoftveresen is. A felbontást vizuálisan a következő oldalon láthatjuk.

## Diagram

A hardver által megvalósított részeket kék háttérszínű téglalapokkal jelöltem.

## 

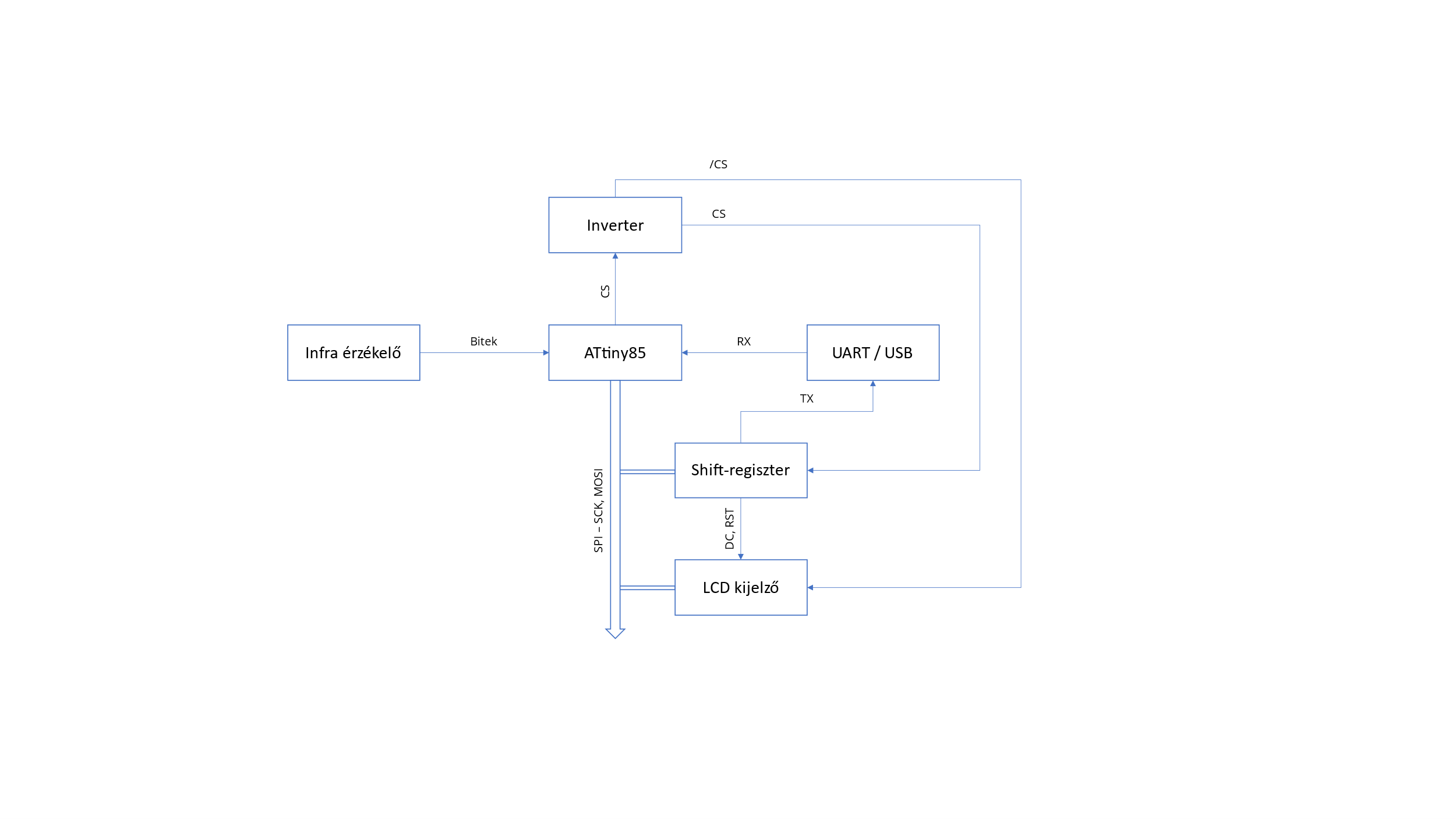
## Hardver blokkvázlat

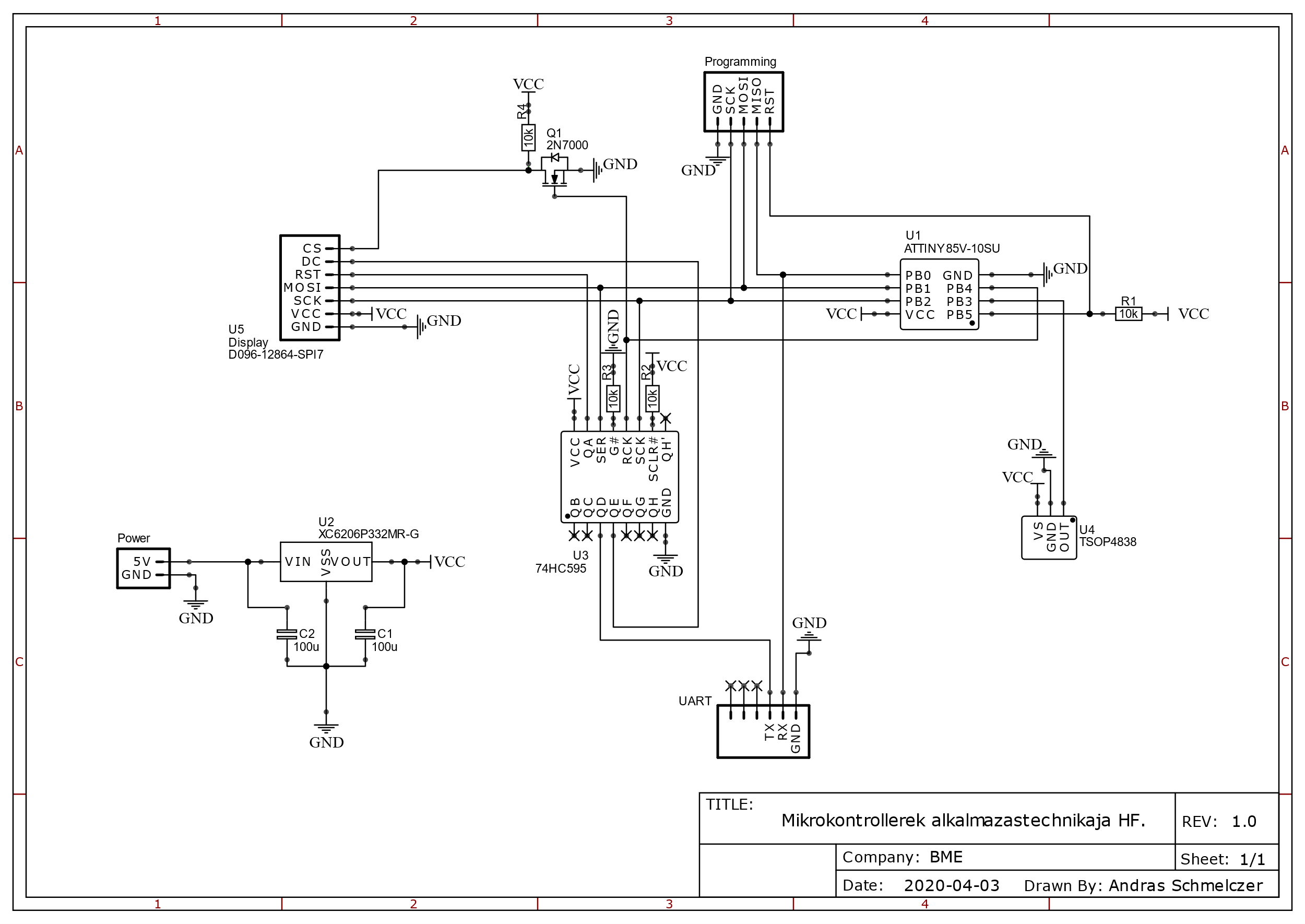
Maga a mikrokontroller a korábban említett ATtiny85-ös lesz. Ez egy kellően olcsó és kisméretű eszköz, meglepően magas teljesítménnyel. Hátránya a kisméretű RAM és programmemória. Ezeket ügyes programozással kell áthidalni.

Egy 128 x 64 pixel felbontású SPI-on kommunikáló kijelző szolgál kimenetként, egy TSOP szériás infra érzékelő pedig bemenetként. A kijelző felbontása (és beépített hardveres gyorsítása) lehetővé teszi a kétszeres upscaling használatát, így elég 64 x 32 pixelnyi képet kiszámolni. Fejlesztés alatt a mikrokontroller debugWIRE interface-én kerül sor a debugolásra. A tárgykövetelmények miatt UART-os kommunikációra is szükség lesz, viszont a lábak elfogytak, így a kimeneti lábakat egy shift-regiszterrel kell bővíteni, amivel kvázi SPI-on keresztül lehet kommunikálni.

A (kimeneti) portbővítés az eddigi funkcionális leírásokban nem szerepelt, hiszen ez nem a specifikációban részletezett funkcionalitáshoz szükséges, hanem egyszerűen a mikrokontroller kis lábszáma által okozott akadályt szünteti meg.

A két SPI lábakat használó periféria miatt szükséges a chip-select jelek használata, ezt egy inverter alkalmazásával biztosítom.

A mikrokontroller beépített USI-ja az SPI kezelésre lesz konfigurálva, lévén ez az időkritikusabb a relatíve nagy sávszélességet igénylő képkockák átküldése miatt. Az UART és az infra kezelésére pedig maradnak a pin change és timer interruptok.

Kapcsolási rajz

# Szoftver rendszerterv

A teljes szoftveres dokumentációt mérete miatt nem illesztettem be ebbe a dokumentációba. A kommentek és a kód felépítése alapján doxygen-nel és pdlatex-kel generált dokumentáció a software\_documentation.pdf fájlban található meg nyomtatható verzióban. Böngésző segítségével célszerűbb a kereshető software\_documentation/index.html fájlon keresztül elérhető dokumentációt használni.

Ebben a részben csak átfogóbb képet szeretnék adni a rendszerről. A részleteket az előbb említett helyen lehet megtalálni.

## Elvek

A szoftver architektúrájának tervezésekor fő szempontok voltak a modularitás, hordozhatóság és a teljesítmény. Nem elhanyagolható igény továbbá a kis méret sem. Mindezek teljesítése végett több alkalommal ismerhető fel az objektum-orientált szemlélet.

Ezek miatt a programban nincs globálisan elérhető állapot, a komponensek saját állapotukat felügyelik, velük függvényhívásokkal lehet kommunikálni.

## Hardverközeli

Madártávlatból három fő részre osztható a kódbázis. Legalul helyezkedik el a hardware-access, ami a legalapvetőbb hardveres funkciókat látja el egy kényelmesen használható API-val. A következő hardver funkciókért felelős modulok találhatók itt.

* Energiahatékonyság
* SPI (sebesség végett blokkoló működés)
* Időzítés (megszakítás alapú működés)
* Digitális kimenet

Másik mikrokontrollerre váltás esetén a változtatások túlnyomó része az előbbi elemekben várható.

Ez a réteg rendelkezik egy Façade jellegű interfésszel, aminek két felelőssége van. Egyrészt deklarálja a hardware-access által biztosított funkciókat. Másrészt a teljes hardveres inicializálásért felel. Utóbbira a kis méret és az egyszerű használat miatt van szükség. A program elején inicializáljuk a HAL-t, utána pedig bárhol includeolva a megfelelő headert, a teljes hardver funkcionalitást elérjük.

## Driverek

A következő réteg a driver réteg. Az itteni modulok lazán csatoltak és egyetlen közös tulajdonságuk, hogy a HAL-t használják funkcionalitásuk megvalósításához.

A következő komponensek helyezkednek el itt.

* Kijelző
* Infra (megszakítás alapú működés)
* Alvásütemezés
* UART (megszakítás alapú működés)

Ezek többnyire egy egyszerű interfész (callback függvény) átadásával inicializálhatók. Erre a laza csatoltság miatt van szükség, igazából ez a dependency inversion principle C nyelvi megvalósítása.

A legmagasabb szintű funkciók az alkalmazás réteg-ben találhatók. Maga a játék és annak logikája itt kerül megvalósításra. Ez saját belső architektúrával rendelkezik, amit a későbbiekben fogok kifejteni.

Végül egy fontos komponens maradt még. A *mediator* az alkalmazás és a driverek között helyezkedik el. Biztosítja, hogy az előbbi két rétegen belül és azok közt megvalósulhasson a kommunikáció. Felelőssége a komponensek orkesztrációja. Mindezt a megfelelő fejlécű függvények átadásával éri el a megfelelő inicializáló függvényhívásokban.

## Alkalmazáslogika

A fő szempont a magasszintű kód írásának lehetővé tétele és a kompakt méret. Ezért a játékobjektumok közt egy prototípus alapú öröklési rendszer került kialakításra. Minden típus rendelkezik egy prototípussal, ami a méretét, a rajzoló és a frissítő függvényét tartalmazza. Maguk a prototípusok a programmemóriában kerülnek tárolásra. Az objektumok rendelkeznek egy mutatóval a prototípusukra, egy pozícióval és egy union-nel, amiben a típus specifikus állapotukat tárolják.

Az objektumok tárolására az objektumtároló szolgál. Ez igazából egy heterogén kollekció pár hasznos függvénnyel. Például segítségével lekérdezhetők az objektumok típus alapján vagy az alapján, hogy adott pixel pozíciót tartalmazzák-e. Továbbá az objektumok frissítését és rajzolását lehet innen kezdeményezni.

Az időközönkénti új objektumok létrehozását az esemény generátor kezdeményezi. Emellett a játékmenetért (indítás, újraindításért) felelős komponens is ebben a rétegben kerül megvalósításra.

Az összes játékobjektum típus rendelkezik saját komponenssel. Adott feladatukat a program többi részét használva igen magas szintű kóddal lehet megvalósítani. Önálló komponense van továbbá az AI-nak, ami az élő felhasználó által használható interfészen keresztül irányíthatja a karaktereket.

Megemlíthető, hogy a kirajzolásra szánt spritesheetek is itt kerülnek definiálásra a kijelző által elvárt speciális formátumban. Ezek az EEPROM-ban kerülnek tárolásra.

## Segédfüggvények

Egyéb sztenderd könyvtár jellegű segédfunkciók is helyet kapnak a kódbázisban. Ezek működése elég magától értetődő. Felsorolás jelleggel a következő komponensekről van szó.

* Véletlenszámgenerátor
* Vektor (2 dimenziós)
* Téglalap
* Makrók
  + Null
  + Bit műveletek

# Felhasználói leírás és kezelési útmutató

Az eszköz bekapcsolásához szükséges egy 3.3 V-os tápfeszültséget biztosítani számára. Ezután az UART – USB átalakító csatlakoztatva a számítógéphez, egy terminál alkalmazással (például PuTTY) csatlakozhatunk az eszközhöz.

Ilyenkor az OLED kijelzőn egy karaktert láttunk az űrhajóban állni. Ezt a karaktert kétféleképpen irányíthatjuk. Egyrészt használhatjuk az imént elindított soros kommunikációt, vagy az eszközhöz tartozó infrás távirányítót is. Előbbi esetén a ’w’, ’a’, ’s’, ’d’ és szóköz billentyűk lenyomásával mozgathatjuk a karaktert, illetve vehetjük rá, hogy interakcióba lépjen a környezetével. A ’+’ és ’-’ billentyűk lenyomásával a kijelző fényerejét változtathatjuk. Távirányító esetében ezeknek a ’2’, ’4’, ’8’, ’6’, ’5’, ’+’ és ’-’ billentyűk felelnek meg. Bizonyos időközönként a soros porton egy ’szám ms’ alakú üzenet érkezik, ami megadja, hogy az előző ilyen üzenet óta mennyi időbe telt a leglassabban előállított kép kirajzolása.

A játék célja, az aszteroidák felrobbantása, miközben az űrhajó épséget megőrizzük. Aszteroidák felrobbanthatóak, ha az űrhajóval ütköznek, vagy lelövik őket. Az űrhajó mozgatásához annak jobb felső részéhez érve kell interakció gombot nyomni. Ettől kezdve a karakter helyett az űrhajót mozgatják a gombnyomások. Visszaváltani az interakció gomb ismételt lenyomásával lehetséges.

Kellő mennyiségű felrobbantott aszteroida után az űrhajó felszereltsége nő. Az újonnan megjelenő tárgyakkal is interakcióba lehet lépni. Bal alul az űrhajót javíthatjuk (a jelenlegi állapotát jobb középen elhelyezett csíkon láthatjuk). Jobb alul található az ágyú. A felszereltség növekedése mellett gép által irányított űrhajósók is megjelenhetnek, akik elvárható pontosággal végzik a szükséges feladatokat a játék célja eléréséért.

Ha a korábban említett csík hossza nullára csökken, egy animáció lejátszása után a játék elölről kezdődik.

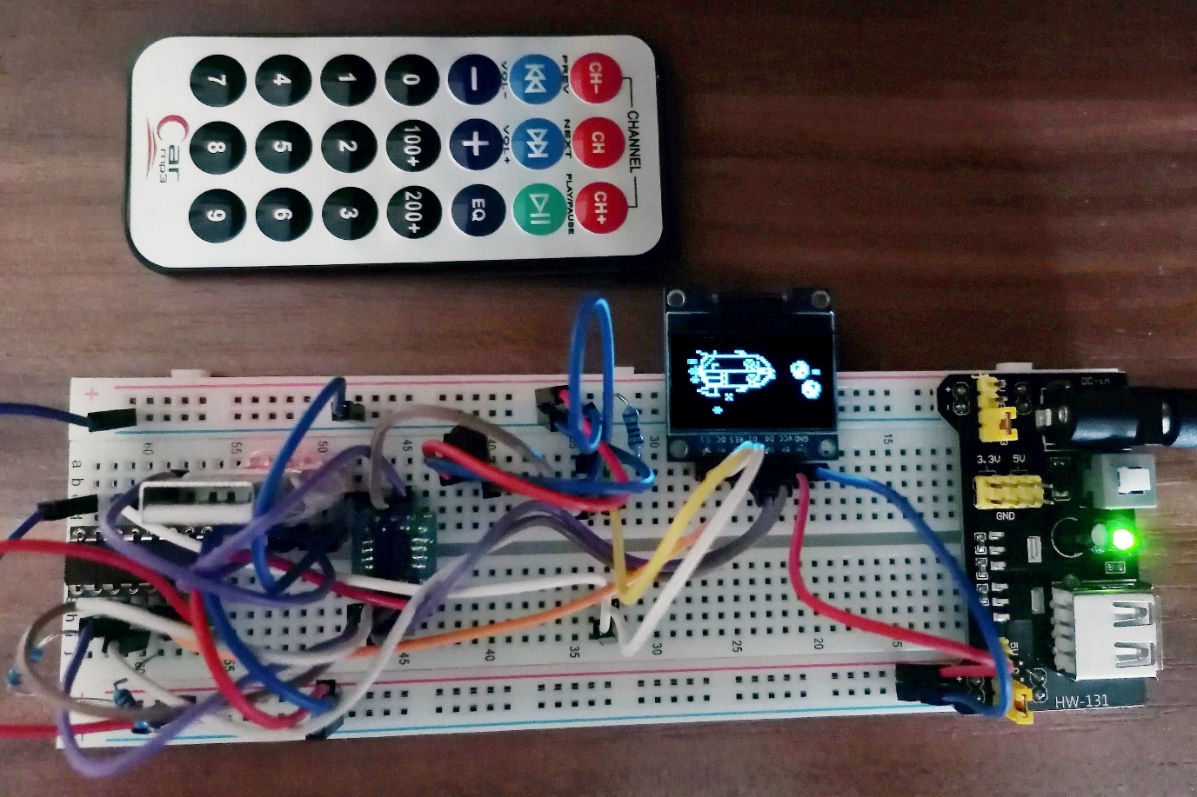
Bemérési tapasztalatok

Többször volt szükséges átépíteni a hardvert amiatt, mert az olcsó és kisméretű breadboard-ok csatlakozói nagyon hamar elhasználódnak. Emiatt legközelebb mindenképpen egy komolyabb breadboard-on kezdeném el a prototipízálást.

Emellett a fejlesztés első fázisában fejlesztőkártyát (vagy külső debuger-t) használnék a beépített debugolási lehetőségek miatt. Továbbá célszerű, ha mindenből legalább egy tartalék van nálunk, erre nemegyszer volt szükségem.

Alkatrészek kapcsán legpozitívabban a TSOP-ban csalódtam, nagyon jólszűrt és erős jelet ad még akkor is, ha az infra forrása relatíve messze van tőle.

# Elkészült eszköz

Egy kép a dokumentációban részletezett eszköz breadboardon elkészült prototípusáról.

Az előző eszköz alapján készített, továbbfejlesztett konfiguráció pedig ezen a képen látható. Utóbbiról több információ a <https://schmelczer.dev> oldalon található.

# Datasheet borítók

A következőkben a felhasznált hardverek datasheetjeinek első oldalát másoltam be. Ezek a következő chipekhez tartozó boritók:

* 2N7000G
* 74HC595
* D096128-SPI-7
* ATTINY85-V
* TSOP48

