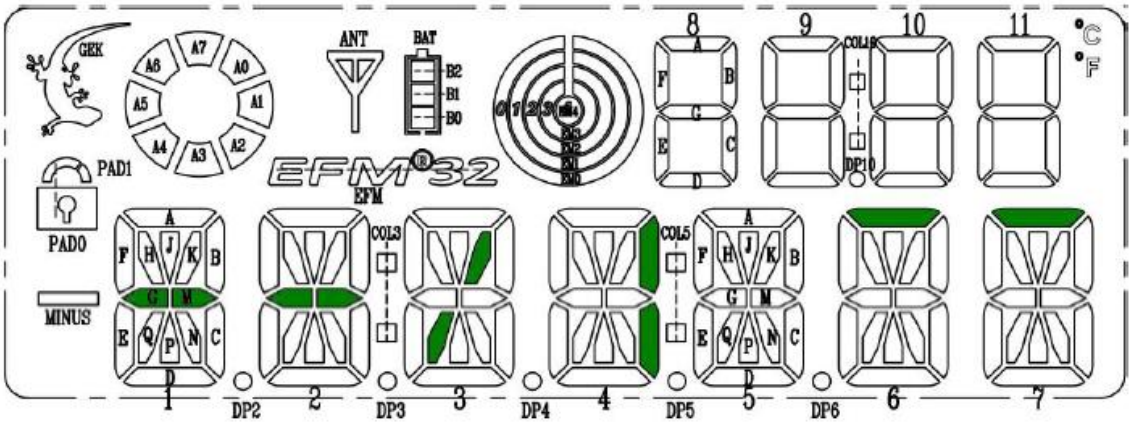
Párhuzamos- és Eseményvezérelt Programozás Beágyazott Rendszereken Házi feladat

Specifikáció

A házi feladatunk egy korábbi BAMBI házi feladat továbbfejlesztése. A játék maga a torpedóra hasonlít, a Gecko alfanumerikus szegmenskijelzőjén rejtve el van helyezve négy darab egyenként két szegmensből álló hajó, melyek nem hajolhatnak meg és nem érhetnek össze. Az egyes karakterek középső két szegmense egy szegmensnek számít. Egy érvényes elrendezés a fentiek alapján pl.:



1. ábra: Egy érvényes hajóelrendezés

A játékos a kurzor segítségével tud lépni. A kurzor aktuális helyét az aktuálisan kijelölt szegmens villogásával jelzi. A játékos lövést tud leadni egy-egy szegmensre. A lövést animáció kíséri, melyet a Gecko bal felső kör alakú szegmenskijelzője feltöltődéssel jelez. Találat esetén ugyanazen szegmenskijelző háromszor villog a feltöltődés után, ellenkező esetben nem történik semmi. A játékos az animációk közben is tud lépkedni és lövéseket leadni, bár ekkor a kurzort nem látja, és az összes animáció végbe fog menni, illetve ugyanarra a szegmensre többször is lehet lőni. A játék közben az addigi lövések számát a Gecko jobb felső sarkában lévő numerikus szegmenskijelzőn olvashatjuk. A játékos akkor nyer, ha eltalálta az összes elrejtett hajó minden szegmensét legalább egyszer.

A játék logikai működéséért és a kijelzésekért a Geckon futó FreeRTOS operációs rendszert használó kód felel. Az irányításért és a játék végét követően a statisztikai adatok kijelzéséért a Linux operációs rendszert használó PC-n futó kliensalkalmazás felel. A PC és a Gecko UART

kapcsolaton keresztül kommunikál egymással. A feladatunk IMSc feladatrészt is tartalmaz, melynek lényege, hogy két játékos tudjon egymás ellen versenyezni két különböző Geckon és kliens PC-n. A verseny győztese az, aki hamarabb kilövi az összes elrejtett hajót (megnyeri a játékot), ekkor a játéknak vége. A versenyzők az előre definiált 16 pálya közül ugyanazon kezdik meg a versenyt. Mindezekhez a kliensalkalmazások TCP kapcsolaton keresztül kommunikálnak egymással.

A FreeRTOS alkalmazás

A feladat 3 fő taskra és 1 interrupt függvényre oszlott.

Az UART0_RX_IRQHandler fogadja a soros porton érkező adatokat és egy semaphoreal jelzi az új adat jöttét. Az IRQ kezelő függvény jelzi az ütemező felé, hogy szeretné a hTaskUart0 taskot felébreszteni, ugyanis ez a task kezeli az adatokat.

A prvTaskUart0 az UART0_RX_IRQHandler által nyújtott semaphort elveszi és belerakja egy Queueba. Ennek a tasknak más feladata nincs. A 3 task közül neki van a legnagyobb prioritása, hogy a program a lehető leggyorsabban tudjon reagálni az új adatokra és hogy minimalizáljuk az adatvesztés kockázatát, hisz ha egy hosszabb task futna mielőtt még el lehetne menteni az új adatot, adatvesztés történik.

Prioritásban a következő task a prvTaskLCD task. Ő tulajdonképpen a ”fő” task. Kezeli a kezdeti pályaválasztó frame helyességét és egy case szerkezettel valósítja meg a kívánt működést. Mozgat, lő és leállít. Az adatokat a korábban említett Queue-ból veszi ki. Továbbá ő tartja számon a lövések és találatok számát és kezeli a kijelzőre való írást.

A kijelző logikája 4 struktúra tömbből áll. Egy a rejtett hajókat tartalmazza (MAP), egy a kurzor aktuális pozícióját (CURSOR_POS), egy az eltalált hajók helyét (HIT), egy pedig ezeknek valamely kombinációját, amit meg kell jeleníteni (FIELD). Ez tulajdonképpen a kurzor helye és az eltalált hajók struktúráinak szummája (FIELD = CURSOR_POS + HIT).

A legalacsonyabb prioritású task a prvTaskCursorBlink. Ez a task 750 milisecundumonként fut le. Ez egy empirikus szám. Próbálgatás során ez keltett egy kellemes villódzó hatást. A task tulajdonképpen felváltva kivonja, illetve hozzáadja a FIELD-hez a kurzor helyét. A folyamat

azért lesz megbízható, hogy nem veszi el egy hajó helyét a FIELD-ből, mert a prvTaskLCD mindig hozzáadja új kurzormozgatás esetén az eltalált hajókat a FIELD-hez.

A Linux alkalmazás(ok)

Mivel két kliensalkalmazás kommunikál egymással TCP kapcsolaton keresztül, így nem elég egyféle alkalmazás, hiszen a hálózati kapcsolathoz szükség van kommunikációs szempontból egy szervert megvalósító kliensalkalmazásra (továbbiakban: szerver) és egy klienst megvalósító kliensalkalmazásra (továbbiakban: kliens).

Mindkét alkalmazás bemenetként ugyanazon parancssori argumentumokat várja a következő sorrendben: a TCP szervert futtató PC IPv4 címe (xx.xx.xx.xx formátum), a kommunikációhoz használt socket portszáma, végül az adott PC-hez kapcsolódó Gecko kártya soros portjáért felelős fájl neve elérési útja (pl. `/dev/ttyACM0`). Ha a felhasználó ennél kevesebb paramétert ad meg, az adott program figyelmeztet, és leáll.

A parancssori paraméterek ellenőrzése után mindkét alkalmazás megkezdi a hálózati kommunikáció előkészítését. Ehhez a szerver létrehoz egy socketet (sikertelenség esetén hibaüzenettel leáll), bindol rá (sikertelenség esetén hibaüzenettel leáll), elkezd rajta hallgatózni, majd megpróbálja blokkoló módon fogadni a kliens csatlakozási kérelmét. A kliens hasonlóan létrehoz egy socketet (sikertelenség esetén hibaüzenettel leáll), majd megpróbál csatlakozni a szerverhez. Sikertelenség esetén hibaüzenettel leáll, ellenkező esetben a sikeres csatlakozást is üzenettel jelzi a felhasználó felé.

A kapcsolat létrejötte után a szerver a felhasználó inputját várja, vagyis a kiválasztott pálya számát (00-15). Mindezt addig várja, amíg a definiált tartományon belül nem érkezik egy pálya szám. Ezt követően – illetve a kliens ezzel párhuzamosan – megnyitják az alkalmazások a saját Geckoikkal történő soros porti kommunikációhoz a fájlleírót (sikertelenség esetén hibaüzenettel leállnak). A szerver a játék által kívánt formátumúvá (pl. xx00) alakítja a felhasználótól kapott pálya számát, majd elküldi azt a kliensnek. Ezt követően mindkét program kiküldi a játék kezdéséhez szükséges sztringet a soros portjára, majd a felhasználót értesítik a játék kezdetéről. A programok futásuk további részére a játék irányítása miatt kikapcsolják az echo funkciót a standard bemeneten (hogy ne írja ki folyton, hogy merre lépnek meg mikor lőnek), illetve azt is, hogy `\n` karaktert várjon a standard input minden bevitelhez.

Ezt követően a játék irányításáért felelős programrészek következnek mindkét alkalmazásban. Ez lényegében egy nagy `while` ciklust jelent, amiben a `select()` rendszerhívás segítségével

várakoznak egyszerre többféle inputra a programok. A standard inputról a kapott karaktereket elküldik a soros portra alapesetben, ezzel vezérlik a Geckokon futó játékot (lövés, mozgás). Ha a speciális kilépésért felelős karakter megkapja egy program, akkor elküldi azt a másik programnak is, majd a ciklusból való kilépésért felelős feltételváltozó értékét igazra állítva kilép a ciklusból. Amennyiben a soros portról kap adatot egy program, az a játék végét jelenti, hiszen a Gecko csak ekkor küld adatokat a PC felé (futási idő, lövések száma). Ekkor elküld egy győzelmet jelentő `w` karaktert a másik programnak, majd kilép a ciklusból. Ha a másik programtól kap adatot a socketen keresztül, akkor a fentiek szerint ez azt jelenti, hogy a játéknak vége, úgyhogy kiküldi a kilépésért felelős karaktert a saját soros portjára, majd kilép a ciklusból. A játék futása alatt a szerver program végig számontartja, hogy volt-e nyertes a játékban, illetve azt is meghatározza a végén, hogy ki az.

A játék végét követően statisztikai adatok kiírásával fejeződik be a programok futása. A szerver elküldi a kliensnek a győztest, majd, ha tényleg volt győztes, akkor a kliensnek küld egy `w` karaktert, ellenkező esetben egy `l` karaktert. Ha nem volt győztes, akkor a programok kiírják, hogy az adott programot megszakították, és azért lett vége. Ellenkező esetben a programok beolvassák a soros portjaikról a diagnosztikai adatokat. A játékidő meghatározása mindig a szervert futtató PC-hez csatlakozó Gecko mérése alapján történik. A szerver elküldi ezt, meg az ő lövéseinek számát a kliensnek, kliens pedig a saját lövéseinek számát a szervernek. A programok kiírják a győztes nevét, a játék időtartamát, és a játékosok lövéseinek számát.

Végezetül lezárásra kerülnek megnyitott fájlleírók, illetve visszaállításra kerülnek a standard input beállításai.