Nao robot fényképező alkalmazás

Bemutató: https://www.youtube.com/watch?v=2oRcpbcC5Qg

Összefoglaló:
A roboton futó alkalmazás visszaszámol 3-tól, kimondja, hogy „Cheese”, végül jelez a weboldalnak, hogy mehet a fényképezés. A weboldal ciklikusan kiolvassa a robot memóriájából a „kulcs” értékét, ami ha 1-re vált, akkor fényképez. A robot és a weboldal között egy Python program tart kapcsolatot, amelyik webszervert indít, kezeli a beérkező kéréseket, megformálja a parancsot, és elküldi a robotnak, a robot válaszát szintén feldolgozza, és visszaküldi a weboldalnak.

A robot alkalmazása:
A Visszaszámlálás doboz beépített modulokból van összetéve. Egy Say doboz kimondja, hogy 3, egy Wait vár 1 másodpercet, majd kapcsol a következő Say-re, amelyik 2-t mond, végül az utolsó Say 1-et mond, és ezután is vár 1 másodpercet. Ekkor a Visszaszámlálás aktiválja a kimenetét, ami egy Say dobozra kapcsolódik, amely a „Cheese” kimondása után a Kép készítése mondult indítja. Ez már egy saját Python szkript, így néz ki:

```
memory = ALProxy("ALMemory")
memory.insertData("kulcs", 1)
```

A memóriában a kulcs nevű bejegyzést 1-re állítja.
A Choreography projekt könnyen bővíthető, hogy a valós funkciót ellássa. Egy Movement tracker-rel követi a mozgó embereket, Nao marker felmutatására pedig elindítja a fényképezést. A három másodperces visszaszámlálásnak az is a funkciója, hogy legyen idő eltenni a markert. Sajnos ezt már nem tudtam élőben tesztelni.

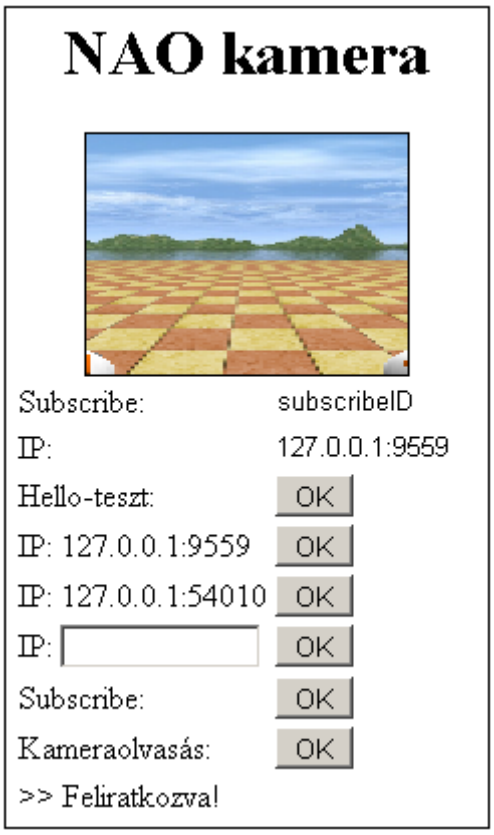
A Python program:
BaseHTTPServer könyvtár felhasználásával webszervert indít. A kommunikációs portot 8080-ra állítottam. A beérkező kéréseket a MyHandler osztály fogadja. GET kérésre HTML fejlécet küld, majd a lekérdezési címből kiveszi a változókat: ip, eval, call, oldal. Az ip-vel lehet megadni a robot címét, az eval a robotra küldendő parancs, a call-t már nem használom (eredetileg callback függvények hívására kellett volna, de azóta ezt a részt megoldottam a kliensoldali Javascript kódban), az oldal pedig a tartalom visszaadásának módját választja ki, ami lehet „query” – parancsra a válasz küldése (a válaszból levágja a robot által küldött <HTML><BODY><PRE> és </PRE></BODY></HTML> kódrészleteket), „memory” esetén kiolvassa a robot memóriájának „kulcs” nevű mezőjét, „memory_torlese” esetén ezt a kulcsot 0-ra állítja, minden egyéb esetben pedig a főoldalt (index.html) tölti be.

A kliensoldali kód:
A kód eleje statikus HTML, amelyik a keretet, címet, feliratokat és gombokat tartalmazza. Javascript függvények teszik dinamikussá a felületet. Az URL-ből kiolvassa a paraméterek értékét (IP-cím, feliratkozási azonosító), illetve amit beállítunk, az később az URL-ben is megjelenik (újratölti az oldalt). A kiolvasást a getUrlVars() függvény végzi. Van 2 előre beállított IP-cím, melyeket kiválasztva az ip1() és ip2() függvény fut le, illetve van lehetőség saját IP megadására, ezt pedig az ip3() kezeli. Az ip4() a függvényparaméterből olvassa be a beállítandó címet. A szerverinfo() a megadott szöveget a weboldal alján jeleníti meg (ez az alert() alternatívájaként lett létrehozva, hogy ne kelljen mindig az OK-ra kattintani a felugró ablakban).

A legfontosabb függvény a query(), amely a program írása során sokat változott. Jelenleg egy AJAX GET-kérést intéz a webszerver felé, amelyben megadja a Nao robot vezérléséhez szükséges információkat: az IP-címet, subscriber ID-t (kameraolvasáshoz), lefuttatandó parancsot (eval). A visszakapott adatot a callback() függvénynek adja át. Az AJAX metódust jQuery $.get() parancsával hívom meg.

A következő sorokban egy ciklikusan lefutó kód szerepel. Ezt egy setInterval() parancs végzi, és másodpercenként elküld egy memóriaolvasási kérést a webszervernek. Amennyiben a válasz értéke 1, ezt jelzésnek veszi, visszaállítja 0-ra, majd elindítja a kameraolvasás procedúráját, mely a subscribe() függvénnyel indul.

A subscribe() megnézi, hogy van-e már beállított sub változó (ebben tároljuk a kamerára való feliratkozáshoz tartozó azonosítót), ha nincs, alapértéket ad neki, majd megpróbál leiratkozni róla. Ez azért kell, nehogy többször feliratkozzunk, mert akkor a robot automatikusan megváltoztatja a subscribe ID-t. Ha ez lefutott, akkor a subscribe_callback() függvényt indítja, mely feliratkozik a „subscribeID”-ra, és indítja a subscribe_callback_callback() függvényt, amelyik már csak kliensoldalon szépíti az oldalt; szerverinfóként kiírja, hogy Feliratkozva!, megjeleníti a subscribe értékét, végül elindítja a kiolvas() függvényt, amelyik a kiválasztott subscriber ID-val lekéri a kameraképet. A kép feldolgozásakor fontos, hogy a webszerver formázza meg a robot válaszát, különben a JavaScript nem képes kezelni az óriási stringet. A szerver a következőket csinálja: levágja a <HTML><BODY>... kódrészleteket, végül kicseréli a \ szimbólumokat *-ra. Ez azért kell, mert a \ jel a JavaScriptben karakterkódot jelöl, és a robot a képpontok RGB kódja elé mindig tesz egy \ -t. Ugyanakkor szükséges az elválasztó jel, ezért *-ra cseréljük. A megformázott karaktersorozatot továbbküldjük a kamera() függvénynek, amelyik a HTML-ben található canvas elemet vezérli. A stringből kiolvassa a pixelek értékét, és összeállít egy tömböt, amelyik az összes képpontot tartalmazza, végül a canvas vászonra rárajzolja a tömböt. Ha minden jó, ezt látjuk:



Készítette: Kurdi Bogdán