Alkalmazásfejlesztés házi

Riasztó felépítése

Sziasztok! Ez a doksi tartalmazza a riasztó felépítését, amit megcsinálok itthon. A mikrokontroller egy arduino lesz, amihez hozzácsatlakoztatok egy potit, amivel szimulálni tudjuk egy akkomlátor feszültségét, amit tudunk mérni. Továbbá lesz rajta egy mágneses reed-szenzor, ami egy digitális 1 vagy 0 jelet ad ki annak függvényéven, hogy (pl. egy ajtó) nyitva van-e vagy csukva. Ezen kívül lesz rajta még egy PIR szenzor, ami szintén egy digitális 1 vagy 0 jelet állít elő. kimeneti portoknak egy RGB led-et gondoltam, amivel jelezhetjük, hogy be van e riasztva, ezen kívül egy kis buzzert gondoltam, ami riasztáskor megszólal.

Ha van még ötleletek, akkor az „ötletek” pontban írjátok le.

Röviden a ki és bemeneti egységek felsorolva:

* Inputok
  + potméter: az aksi szimulálása – egy 0-255-ig terjedő jelet fogunk mérni AD-val
  + reed szenzor: digitális 0 ha nyitva van, digitális 1 ha zárva
  + DHT11 hőmérséklet és páratartalom mérő, hogy tudjuk vektoros üzeneteket küldeni
* Outputok:
  + LED: piros élesített állapotban, zöld alapállapotban
  + Buzzer: Megszólal a riasztáskor

Kommunikáció QT-val: szerintem soros porti kommunikáció lenne előnyös, hiszen az arduinon van ilyen port.

Ötletek:

* ide írjátok az egyéb ötleteket

# 2. Követelmények

x megvan

x, még nincs meg, de készül

x nincs meg

Minimális feladatok (e nélkül a házi feladat értékelhetetlen és nem ér pontot, valamint az aláírásnak is előfeltétele):

● A kliens programnak C++-ban, Qt és QML-ben, qmake és gcc toolchainnel kell készülnie, GIT verziókövetéssel a tárgy keretében létrehozott classroom.github.com-os repositoryban.

● Grafikus felülettel kell rendelkeznie, ami parancsokat tud küldeni és állapotot tud fogadni a robottól vagy egy szimulátortól.

● A szimulátorral TCP socketen, soros porton vagy bluetoothon keresztül kell kommunikálnia.

● Egy parancs kiadására (pl. start) a visszakapott adatokból egyértelműen látszania kell, hogy a robot (igazi vagy szimulált) tényleg elindult. (Például a motoráram megnő, változik a pozíció stb.)

● A leadáskor a master branchen lévő verzió forduljon és fusson egy Windows 10 vagy Ubuntu virtuális gépen. Természetesen ha egy igazi robot jelenléte kell neki, akkor nem gond, ha nem működik minden funkció, de induljon el és ezt a tényt esztétikus formában jelezze. A helyes működés pedig a demó videón úgyis látszani fog.

Alap feladatok (max.40 pont):

● A robot állapot leírásában kell lennie skalár, vektor és szöveges (pl. log üzenetek) adatoknak is. (A vektor itt olyan értéket jelent, ami nem skalár, hanem pl. a vonal szenzorsor fényerő értékei egy adott pillanatban. Olyan érték, ami egy időpillanatban sok számot tartalmaz, így elküldeni egy vektor sorosításával praktikus.)

● Az aktuális állapot szövegesen és a vektor érték(ek) esetében grafikusan is jelenjen meg.

● A korábbi állapotok szövegesen és grafikusan jelenjenek meg (redundancia nem kell, minden úgy jelenjen meg, ahogy logikusabb, csak legyen szöveges és grafikus is).

● Az alkalmazás felhasználói felülete ne hasonlítson kísértetiesen a SimpleTelemetryVisualizer minta alkalmazáséra.

● Az alkalmazás használja a Qt signals and slots mechanizmusát legalább egy olyan helyen is, ami a SimpleTelemetryVisualizerben nem szerepel.

● A dokumentáció készüljön a forráskód alapján (kiegészítve egyéb fájlokkal) Doxygen segítségével. A generált HTML verziót a github “release” funkciójával kell letölthetővé tenni.

● A dokumentációban legyen legalább 1 UML class diagram és legalább 1 UML szekvencia diagram. (Itt nem a Doxygen által automatikusan generált UML diagramokra gondolok, hanem saját készítésű, áttekintő diagramokra, amin azok az osztályok szerepelnek, ami a magyarázathoz ott éppen indokolt.)

● A kliens program felhasználói felülete legyen esztétikus. Például az ablakot átméretezve ne essen szét az egész, hanem kövesse az átméretezést.

● A GIT repositoryban látszódjanak a fejlesztés során készült commitok, vagyis ne egyetlen commit rakja be a kész programot a repositoryba a munka legvégén**. A commitok ne szélsőségesen csak egy csapattagtól származzanak**. A commit megjegyzések legyenek kifejezőek.

Bónusz pontok:

● 6p: Alapos öntesztelő funkció a robot számára. A tesztet futtathatja a kliens program is, de a robot firmwareje is. A lényeg, hogy van öntesztelési funkció.

● 3p: A dokumentáció számos (5+) UML diagramot használ, a leírások kihasználják a Markdown lehetőségeit, valamint a dokumentációban vannak hivatkozások, @see, @warning stb. parancsok.

● 3p: QTest unit teszt, legalább 3 eltérő (nem triviális) test case.

● 5p: A fejlesztés során tapasztalt tanulságok részletes összefoglalása egy publikálható jegyzet (“snippet”) formájában, githubon pull request formájában leadva. Erre példát a snippet oldalon az előző félév hasonló összefoglalói között találhat: http://bmeaut.github.io/snippets/snippets/AlkFejlHfTanulsagok/alkfejlhf/ (Ennek határideje is a házi feladat leadási határideje.)