Kompetenciafejlesztés IoT rendszerekkel

Korom Szilárd[1], Dr. Illés Zoltán[2]

[1]korom.szilard@gmail.com; [2]illes@inf.elte.hu
ELTE IK

Absztrakt. A középiskolai informatika/programozás oktatás ritkán helyezi az ismereteket kontextusba. A diákok ritkán látják a *Big Picture*-t, nehezen tudják megfogalmazni, hogy egy új témakör bevezetésére miért van szükség, azzal milyen problémát akarunk megoldani. A kutatás célja ennek azonosítása az elsőéves programozó-informatikus hallgatók körében, valamint egy lehetséges megoldás megfogalmazása az *IoT* (Internet of Things – A dolgok internete) *rendszereken* keresztül.

Kulcsszavak: programozás, informatikai kompetenciák, valós idejű rendszerek, beágyazott rendszer, raspberry pi, rendszerszintű gondolkodás

1. Bevezetés

A programozás oktatás során leginkább az *algoritmikus gondolkodás kompetencia* kerül előtérbe középiskolai szintén. Azonban az még jellemzően a felsőoktatásra is igaz, hogy ritkán hangsúlyozódnak a valós életbeli, komplex problémák, ahol a különböző technológiákat, paradigmákat integráltan kell alkalmazni. Ez azért probléma, mert a diák/hallgató az oktatás keretein belül csak elszórtan tud kilépni a konkrét tananyag egységből. Például az „adatbázisok" tananyag középiskolában is [1], de az nem kerül elő hogyan és mire használják a való életben.

Egy korábbi cikkünkben kiemeltünk 4 kompetenciát az informatikát érintők [2] közül, amikor a „milyen jellegű feladatokkal érdemes tanítani a programozást" [3] kérdésre kerestem a választ:

1. Algoritmikus gondolkodás
2. Adatmodellezés
3. A valós világ modellezése
4. Rendszerszintű gondolkodás

Jelen esetben azt kívánjuk bemutatni, hogy létezik olyan technológia, keret, melyben a diákokhoz mérten (életkor, érdeklődés, cél) kialakítható egy tananyag úgy, hogy ezt a 4 kompetenciát érintse (főleg), azaz kilépjen az *egyedi összetevők* szintjéről, a konkrét eszköz használatából, s *rendszer szintű* szemléleted adjon, kontextusba helyezze a megtanultakat.

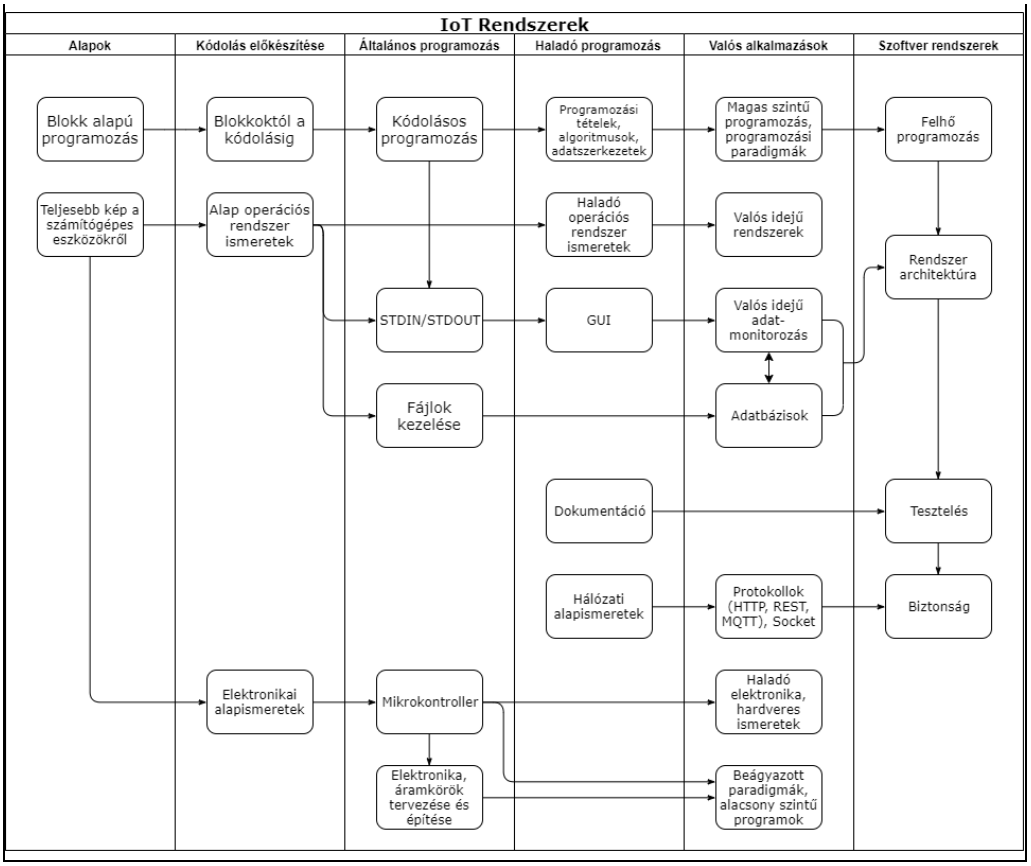
A cikkben az *IoT*[1] *rendszereket* hozzuk fel példának, megpróbáljuk pontosabban leírni mit szeretnénk/tudunk fejleszteni ezekkel az eszközökkel, majd egy kérdőív segítségével megmutatjuk, hogy a felvetett kompetenciák bizony valóban gondok okoznak a diákok számára.

2. Programozás oktatás moduljai IoT eszközökkel

Egy korábbi tanulmányunkban [4] a programozás oktatással foglalkozó néhány cikkek, valamint egyetemi tantárgyak leírásai [4, 5, 6] alapján az alábbi ábrában gyűjtöttük össze, milyen *modulokat*

[1] Internet of Things: A dolgok internete, vagy olyan apró céleszközök rendszere, melyek az interneten kommunikálnak egymással. Ezek jellemzően beágyazott, *valós idejű rendszerek* (de nem feltétlenül).

lehet oktatni *IoT* rendszerekkel. Azóta az ott bemutatott ábrát némileg módosítottuk, kiegészítettük, így kapjuk az 1. ábrát.



1. ábra: *IoT* rendszereken keresztül oktatható *modulok* és egymásra épülésük

Az 1. ábra azonban az *IoT* rendszerektől függetlenül is alkalmazható. Például GUI-t, adatbázist, vagy a tesztelési mintákat ettől teljesen függetlenül is lehet oktatni. Ezzel a szemmel az ábra az egész, és az egységek kapcsolatát tükrözi és további kérdéseket vet fel, ugyanis „a komponensek működése, a részproblémák megoldása nem okozza a rendszer működését" [3]. A cikk fő célja annak a hipotézisnek a bizonyítása, hogy még ha a diák ismer is egy-egy *modult*, nem feltétlenül tudja azt egy magasabb nézőpontból értelmezni, vagy indirekt módon alkalmazni, például, ha nem közvetlenül arra kérdezünk rá, hogy *mire jó az adatbázis*, hanem hogy azt hol és milyen körülmények között hasznos alkalmazni. Ha ezt sikerül bizonyítani, valamint a fenti ábrát elfogadjuk, mint alkalmas leírását az *IoT* eszközökkel való programozás oktatásnak, akkor egy alkalmas keretet kapunk, mely a skálázható, fókuszpontja mozgatható, s egyszerre képes *egyedi összetevők*- és *rendszer szinten* bemutatni egy problémát. Mit adnak tehát a begyázott, *valós idejű rendszerek*[2] a programozás oktatáshoz? Olyan eszköztárat adnak a tanár kezébe, mellyel a csoport életkorának, előismeretének, érdeklődési körének, s per-

[2] Itt elsősorban az olyan eszközökre gondoltunk, mely széles körben elterjedtek a programozás oktatásban, , mint pl.: micro:bit [5] Arduino, *Raspberry Pi* [6]

sze az oktató saját tudásának függvényében egy tananyagot tud felépíteni, mely látványos, bővíthető, s egy-egy probléma könnyen kilépteti a diákok az *egyedi összetevők* szintjén való gondolkodásból, azaz széleskörűbb tudást ad a tanulónak.

2.1. Egy példa

A fentebb leírtakat egy tananyag vázlatával (1. táblázat) szeretnénk szemléltetni, mely az ELTE-n MsC hallgatóknál egy *Embedded and Real Time Systems* nevű tárgy keretében lett kipróbálva. Itt a *Valós idejű rendszerek* elemen (az 1. ábrából kiindulva) volt a hangsúly, egészen egyszerűen azért, mert ez volt a tárggyal szemben a követelmény. A példa véleményünk szerint jól illusztrálja, hogy az *IoT* rendszereken keresztüli oktatás széles körű lehetőséget biztosítanak, rengetek technológia és programozási paradigma oktatását teszik könnyűvé úgy, hogy a diák kénytelen rendszer szinten gondolkodni, olyan megoldást adni, mely számos *modulból*, és azok együttes működésre bírásáról szólnak.

Az alapvető feltevés egyébként az volt, hogy *Raspberry Pi*[3] eszközökkel építsünk okos otthon projekteket, úgy, hogy a rendszer alkalmas legyen arra, hogy további eszközökkel bővítsük (pl.: egy új szoba felokosítása esetén), s az egészet monitorozni tudjuk egy grafikus alkalmazáson keresztül. Természetesen a fontos adatok az interneten keresztül elérhetőek legyenek (pl.: a monitorozás funkció), de építeni kellett egy lokális *IoT* átjárót (mert ez az iparban általában nagyon fontos).

Óra	Leírás
1.	**IoT bevezetés** Témakör bevezetése, elméleti háttér áttekintése. Egy minta program bemutatása, mely egy valós életbeli *IoT* projektet demonstrál **Hardverelemek összeépítése** A hardverek összeépítése, összekötése, élőben kipróbálása mintakódokkal. • Hogyan kommunikáljunk *Sense Hat*[4] *modullal* (szenzor adatok begyűjtése, LED mátrixra kirajzolás) • Hogyan vezéreljük a kamera modult[5]? *tkinter* Python GUI csomag alap funkcióinak kipróbálása
2.	**MQTT protokoll** MQTT broker installálása, konfigurálása és helyi hálózati kommunikációra használása. A *publish-subscribe* filozófia megismerése, események és eljárások használata az előző órai példák felhasználásával.
3.	**Felhő szolgáltatások** Firebase nem-relációs *valós idejű* adatbázis konfigurálása és integrálása egy okos otthon szimulációs projektbe.
4-6.	**IoT Lab** Önálló, valóság közeli rendszerek tervezése és implementálása 2 fős csapatokban.

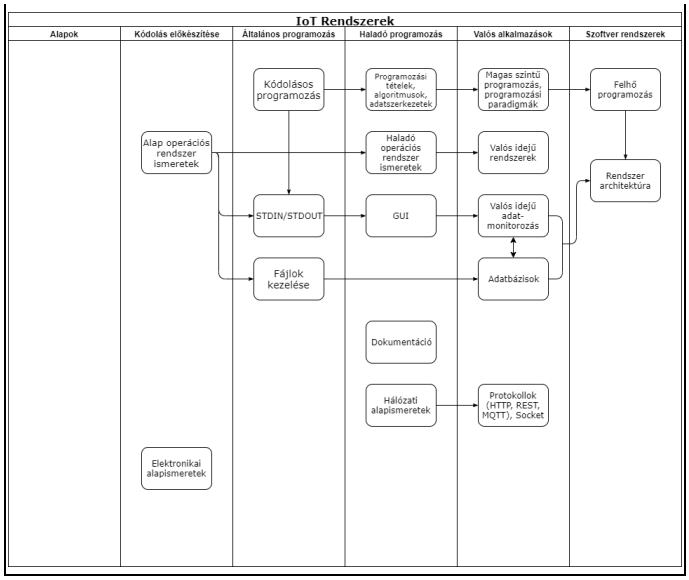
1. táblázat: Egy *IoT* tananyag címszavakban

[3] https://www.raspberrypi.org

[4] https://www.raspberrypi.org/products/sense-hat/

[5] https://www.raspberrypi.org/products/camera-module-v2/

Az 1. ábrából kiindulva szemléltetni kívánjuk mi mindent érintettünk ez alatt a 6*1,5 óra alatt. Látható a 2. ábrán, hogy viszonylag sokat (legalább is középiskolai szinthez mérve, így kijelenthető, hogy rövid idő alatt egy komplex rendszert kellett és tudtak alkotni a hallgatók.



2. ábra: Tananyag által érintett elemek

A skálázhatóság hangsúlyozása nagyon fontos. Nyilván ugyan ez nem tehető meg középiskolai körülmények között, de nem is ez a cél.

Lehetne ugyan ezt a projektet középiskolásoknak is tanítani? Véleményünk szerint igen, ha például nem kell interneten *valós idejű* adatbázison keresztül kommunikálni, vagy ha nincsen monitorozó alkalmazás, vagy lokális MQTT broker.

Lehetne ugyan ezt a projektet tovább vinni, még magasabb szinten oktatni? Meglátásunk szerint igen. Például, ha követelmény, hogy automatikus tesztek ellenőrizzék a rendszer helyes működését, ha nagyobb hangsúly van a biztonságon, vagy nem a kész *Sense Hat modult* használjuk, hanem magunk rakunk össze egy bonyolultabb áramkört.

3. A kérdőív

A kérdőívet elsőéves programozó informatikus hallgatók töltötték ki, pontosan 205-en. A kérdések megfogalmazásánál a következő szempontok domináltak:

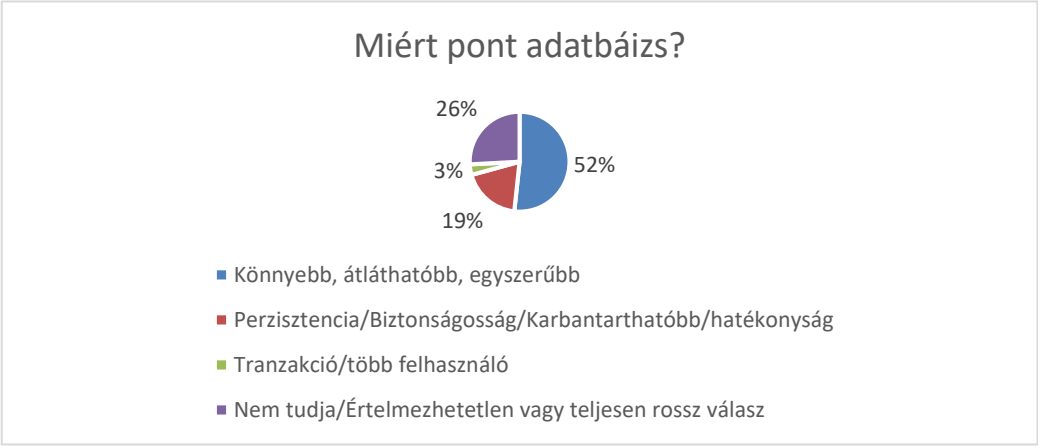
- középiskolai, esetleg aktuálisan tanult tananyaghoz kapcsoló kérdések legyenek
- attól, hogy a hallgató hibás választ ad, még elképzelhető legyen, hogy tudja a tananyagot, vagy teljesítse az adott tantárgyat (akár hibátlanul)
- a kérdések rendszerszintű kompetenciára kérdezzenek rá, vagyis a válaszok ismerete elengedhetetlenek legyenek a konkrét eszköz/technológia egy nagyobb projektben való alkalmazása során, de szükségtelenek az eszköz megtanulása során

Hogy az egyes kérdéseknél miért teljesülnek ezek a feltételek, arra egyesével ki fogunk térni. A kifejtős kérdéseknél a válaszokat próbáltuk kategorizálni. Ez sokszor nagyon nehéz volt, mert helyenként eléggé vegyes választ adtak a hallgatók. Az általunk megfogalmazott kategóriák a diagram-

moknál lesz láthatók, de igyekszünk majd példákat is adni, miket soroltunk az adott kategóriába. A továbbiakban felsoroljuk ezeket, s diagrammokon ábrázoljuk.

3.1. Miért tárolunk adatokat adatbázisban és nem fájlokban?

Az adatbázis-kezelés már a középszintű érettségiben is követelmény. Attól, hogy a diák tud táblákat létrehozni, azok tartalmát módosítani, hozzájuk lekérdezéseket írni a konkrét keretrendszerben (pl.: Access), még nem biztos, hogy látja az egyáltalán miért került képbe. A középszintű érettségin, valamint az egyetem első évében is jellemzően fájlokban történik az adattárolás, így nem biztos, hogy világos az adatbázis egyáltalán hogyan jött a képbe. Később azonban egy valós alkalmazásnál elengedhetetlen, hogy a diák tudja és értse a perzisztencia, a tranzakciók lényegét és értelmét.



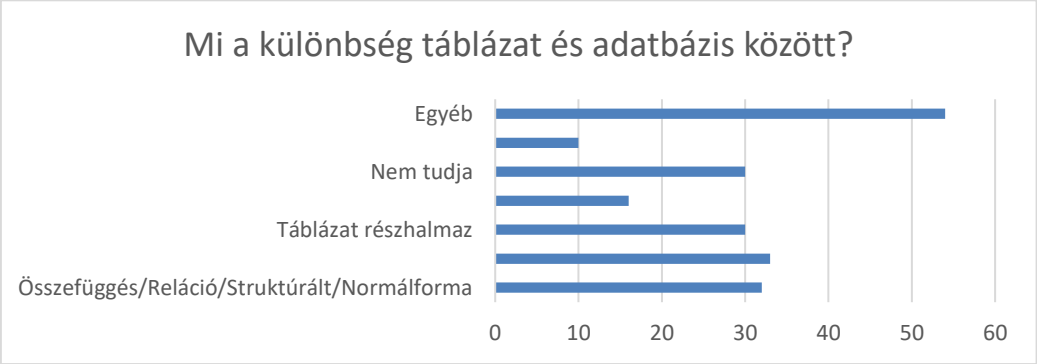
3. ábra: A kiértékelt, kategorizált válaszok az első kérdésre.

A *perzisztenciát*, *tranzakciót* konkrétan senki nem említette meg név szerint. Az ezeknek megfelelő kategóriákba azokat soroltuk, ahol bármiféle utalást találtunk ezekre a fogalmakra. Az utolsó kategóriába olyanok kerültek, amit nem igazán tudtunk hova tenni, például „Tömörítés".

Véleményünk szerint az eredmény eléggé lesújtó, bár a hipotézist igazolja: a diákok nem értik, az adatbázis miért és hol hasznos, egyáltalán mikor és mire használjuk.

3.2. Mi a különbség egy táblázat és egy adatbázis között?

Itt hasonlóan az előző kérdéshez, az adatbázis tágabb értelemben vett hasznára szerettem volna rákérdezni. A diákok a középiskolában gyakran nem értik, miért kell adatbázis-kezeléssel foglalkozni, amikor a táblázatkezelős feladatok nagyon hasonlóak, sőt, az előbbiben előkerülő feladatok nagy hányada meg is oldható az utóbbiban. Könnyen lehet tehát, hogy a hallgató ismeri és tudja használni az Excel és az Accesst anélkül, hogy tudná kívülről nézve mi a különbség a kettő között.



4. ábra: A kiértékelt, kategorizált válaszok a második kérdésre.

Ennél a kérdésnél a válaszok hihetetlenül vegyesek és érdekesek voltak. Nagyon nehéz volt bekategorizálni. Próbáltunk jóindulatúan következtetni arra, mire gondolhatott a hallgató. Természetesen nem konkrétan ezeket a válaszokat fogalmazták meg, pl.:

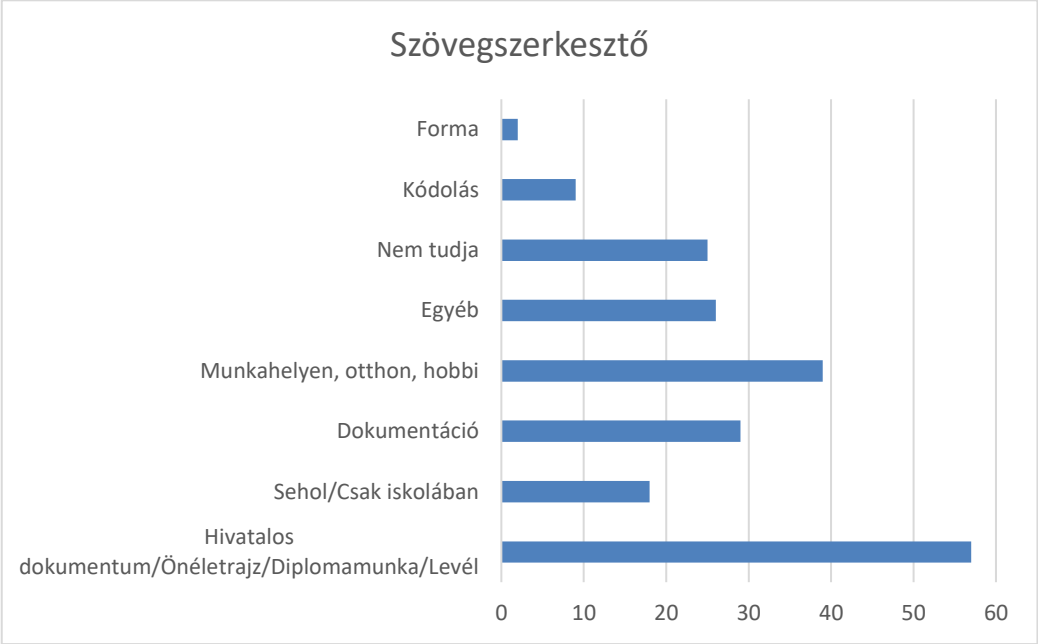
- Összefüggés/Reláció/Struktúrált/Normálforma: Az adatbázis táblái között is végezhetünk műveleteket.
- Egyéb: Adatbázisban több a funkció; Egy adatbázis sokféleképpen variálható, míg egy táblázat bizonyos szinten kötöttebb; Adatbázis olyan táblázat, amiben vannak adatok; Több dolog fér az adatbázisba
- Táblázat részhalmaz: Adatbázis táblázatok gyűjteménye; Több tábla

A kategorizálásnál tehát erősen érvényesült, hogy a diák gondolatmenetét igyekeztünk kategorizálni, s nem a konkrét választ. Még ezzel a jóindulatú módszerrel is nagyon érdekes eredmény jött ki.

Az eredmény azért nagyon meglepő (4. ábra), mert táblázatkezelést és adatbázis kezelést is biztosan tanultak a középiskolában, de ezek szerint egyáltalán nem értik a különbséget.

3.3. Fogalmazd meg, szerinted hol és milyen formában fogod használni a szövegszerkesztőben megtanultakat?

A szövegszerkesztés szintén követelmény már a középszintű érettségin is [1]. Fontossága pedig nem csak a programozók számára hangsúlyos. Van azonban a szövegszerkesztésnek általános szabályai, eszközei, mely szinte minden szövegszerkesztésre alkalmas környezetben előfordul (akár egy e-mail kliens esetében, akár fórumokon stb.). Például, ha ipari környezetben dokumentációt kell majd írnia a hallgatónak, akkor szinte bizonyosan muszáj lesz használnia ezeket az ismereteket, hiszen az valószínűleg nem Wordben fog történni. Hogy ténylegesen tudatában van-e a karakter-, bekezdés-, szakaszszintű formázásokkal (vagy ezen kategóriák puszta meglétéről) az más kérdés, de itt a felvetés arra irányul valójában érti-e, hogy amit megtanult az hol köszön majd vissza.



5. ábra: A kiértékelt, kategorizált válaszok a harmadik kérdésre.

A kategóriák meghatározásánál külön figyeltünk, hogy valaki válaszolt-e a *milyen formában* kérdésre. Itt elsősorban valamilyen általános szövegszerkesztő elemre gondoltunk, például, hogy a karakterformázások más programokban is hasonló logikát követnek, mint pl. Wordben. Ahogy látható erre 2 helyen volt összesen bármilyen utalás. Pár példa:

- Egyéb: Amikor szöveget akarok szerkeszteni
- Sehol: Ha kibukok és titkárnő leszek
- Forma: Formai szerkesztésnél

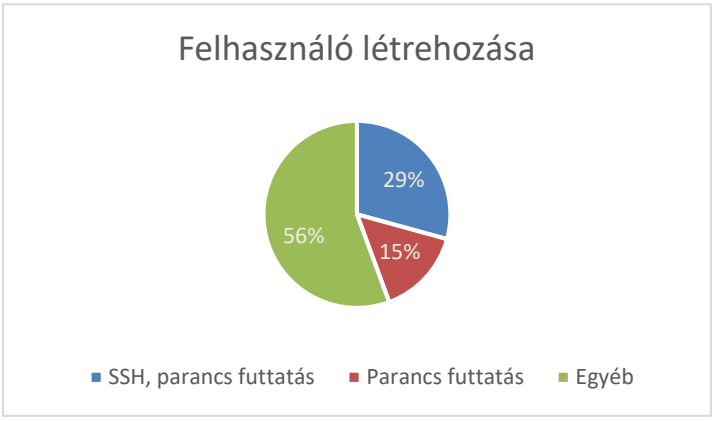
Szerintünk egyértelműen jelzik az adatok (5. ábra), hogy nem sikerült a szövegszerkesztőt általánosan értelmezni. Nagyon sok volt egy konkrét dokumentum megfogalmazása (utolsó pont), s nagyon kevés olyan, ami bármiféle módszerre, vagy logikára utalna. Az külön elkeserítő, hogyan sokan úgy gondolják, hogy semmire sem hasznos.

3.4. Tegyük fel az a feladatod, hogy létrehozz egy új felhasználót egy távoli Unix szerveren. Írd le általános milyen lépésekkel érnéd ezt el?

A kérdést azért mertük feltenni, mert aktuálisan a shell-scriptek tananyagot képeznek. Válaszként olyasmi várunk, hogy:

- SSH kapcsolat a távoli szerverrel admin jogosultságú felhasználóval
- Megfelelő parancs kiadása megfelelő paraméterezéssel

Mivel a felhasználók létrehozása konkrétan nem tananyag, a kérdés arra irányul, hogy érti-e mire fogja használni a shell környezetet, érti-e annak általános felépítését, hogy milyen jellegű problémákat tud majd ezen keresztül megoldani. Attól, hogy erre nem tud válaszolni, a tárgyat még hibátlanul teljesítheti, hiszen a zh keretében csak konkrétan megmutatott parancsokat kell használni, amit ettől függetlenül tudhat.



6. ábra: A kiértékelt, kategorizált válaszok a negyedik kérdésre.

Amikor a válaszokat kategorizáltuk, különösen igaz volt, hogy teljesen távoli válaszokat is a fenti kategóriákba soroltunk. Például:

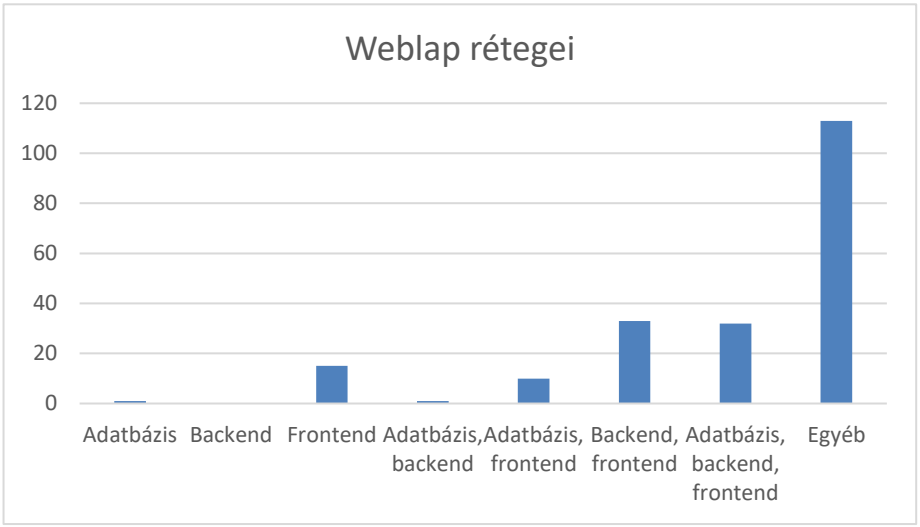
- SSH kapcsolat: Csatlakozás a szerverhez, felhasználó létrehozása
- Parancs futtatás: Google

Igyekezetem arra figyelni mire gondolhatott a hallgató. Ha az volt a válasza például, hogy „Google", szerintem arra gondolt, hogy megkeresi a megfelelő parancsot.

Ami viszont biztos, hogy még így is szembetűnő (a 6. ábra alapján), hogy ha felmerül egy probléma, azt nem tudják munkafolyamatként értékelni, hanem a konkrét megoldást várják, amit persze nem tudnak, hiszen nem tanulták a shell szkriptes tárgy keretein belül. Persze ezzel nem arra akarok utalni, hogy ha valóban kellene ne tudnák megoldani, inkább arra, hogy az oktatás keretein belül az általános, rendszerszintű gondolkodást nem sikerült átadni. Erre később lesznek rákényszerítve.

3.5. Tegyük fel készítened kellene egy webshopot. Sorold fel milyen rétegei vannak az alkalmazásnak!

Az adatbázis-backend-frontend hármasra vártunk, mint megfelelő válasz. Ezekből már középiskolában találkozott az adatbázissal, valamint a frontenddel. A backend réteggel még elképzelhető, hogy nem találkozott, mert egyetemen sem tananyag az első félévben. Ettől függetlenül az alkalmazás rétegelése elvárható a diáktól attól teljesen függetlenül, hogy például tud-e lekérdezéseket írni. A kutatás hipotézise persze az, hogy bár elvárható lenne, mégsem tudja megfogalmazni a rétegeket annak ellenére, hogy külön-külön ismeri azokat.



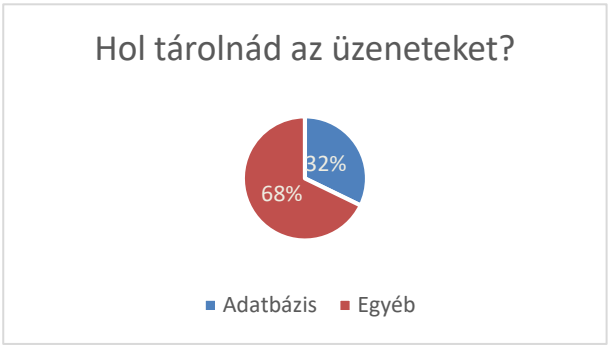
7. ábra: A kiértékelt, kategorizált válaszok az ötödik kérdésre.

Az utolsó adatsor a 7. ábrán jelzi azokat a válaszokat, ahol egyáltalán nem volt a rétegelésre utaló gondolat, vagy teljesen máshogyan osztotta fel (pl.: a weblap funkciói szerint). Mivel a többi adat összesen nem éri el az *Egyéb* kategóriába eső válaszok számát, ezért a felvetésünk helyes volt. Néhány példa a válaszokra:

- Adatbázis, backend,frontend: Design, szerverek, raktár az áruknak; Kinézet, háttérben futó scriptek, adatbázisok
- Adatbázis, frontend: Fizetési szoftver, képek, árak és termékek adatbázisa, adatrögzítő adatbázis..

3.6. Ha írnod kellene egy valós-idejű csevegő alkalmazást, hogyan tárolnád, menedzselnéd az üzeneteket?

A kérdést kifejezetten azért fogalmaztuk meg, hogy egy olyan példát hozzunk, amikor nem a technológia tanulását követi a feladat, hanem a valós problémához kell kötni a technológiát. Válaszként az adatbázisra való bármilyen utalást várunk. A kérdés egyébként azért is szerencsés a véleményünk szerint, mert kifejezetten egy programozói gondolkodás kell ahhoz, hogy a hallgató meg tudja válaszolni. Pusztán annak a felismerése, hogy egy informatikai technológia a válasz, már önmagában értékesnek mondható.

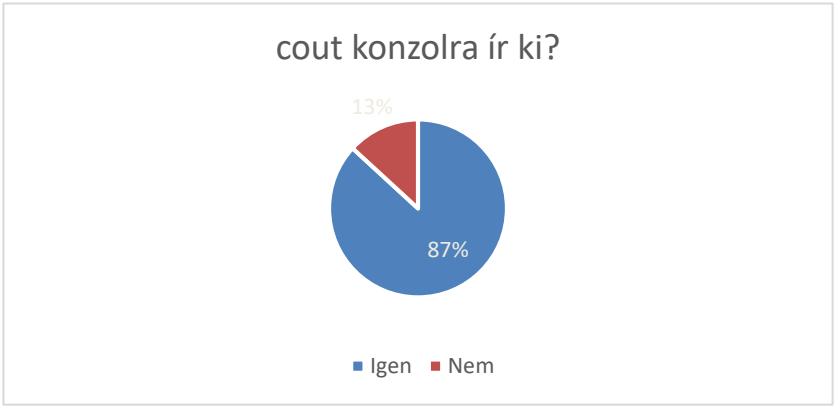


8. ábra: A kiértékelt, kategorizált válaszok a hatodik kérdésre.

Őszintén szólva ennél a kérdésnél arra számítottunk, hogy nem fogja a hipotézisünket igazolni, már csak azért sem, mert az első kérdés utalt rá, hogy adatbázisban érdemes információkat tárolni. Ennek ellenére nagyon szépen megmutatja a 8. ábra, hogy a probléma felől megközelítve egy tanult technológiát még nem képesek felismerni.

3.7. Igaz vagy hamis? C++ nyelven a cout a konzolra írja ki a megadott stringet.

A hallgatók az első félévben C++ nyelven (is) tanulnak programozni. Az eljárás tehát minden bizonnyal ismeretes a számukra. A válasz persze *hamis*, hiszen a standard kimenetre kerül a szöveg. Ha a C++ nyelvet például programozási tételek, alapvető algoritmusok, adatszerkezetek implementálására használják, a standard kimenet fontosságának megértése nem szükséges, elegendő, ha a konzolos alkalmazásban lát valamiféle eredményt. A későbbiekben azonban ez nagyon fontosság válik, hiszen, ha például a háttérben futó szervízeket kell majd írnia ipari környezetben, akkor a kimenetet minden bizonnyal át kell majd irányítania naplófájlokba. A kérdés számunkra azért is különösen érdekes, mert a shell szkript írást is most tanulják, ahol az átirányítások nagyon hangsúlyosak.



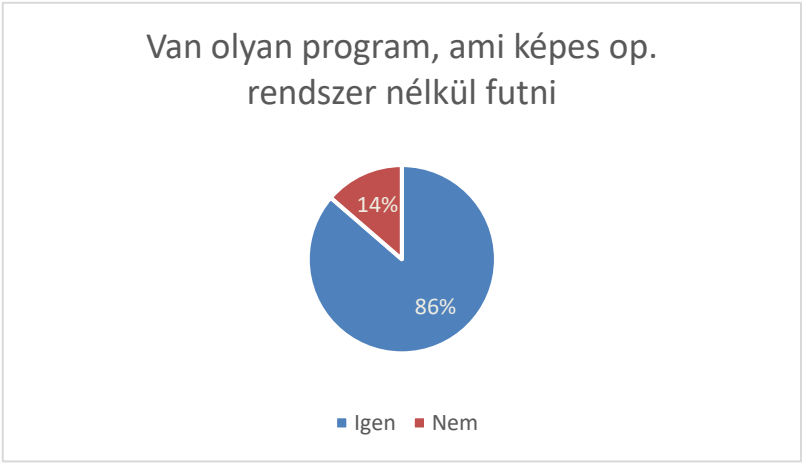
9. ábra: A hetedik (Igaz/Hamis) kérdés válaszai összesítve

Ennél a kérdésnél meg voltak adva az opciók, így nem kellett kiértékelni, csupán összesíteni. Az eredmény (9. ábra) szembetűnő: a hallgatók emlékeznek a parancsra, tudják használni, de a konkrét tapasztalatukra támaszkodnak, mely konkrét feladatokra épül, de nem értik általánosan az hogyan működik. Mivel tanulták már, hogy a sztandard kimenetet át lehet irányítani, ezért két összefüggő,

de apró képesség birtokában vannak, de az eredmények alapján ezt nem tudják összefűzni rendszerré.

3.8. Igaz vagy hamis? Van olyan program, ami képes operációs rendszer nélkül futni.

Itt valójában arra voltunk kíváncsiak, vajon értik-e mi történik, amikor futtatnak egy programot, el tudják-e különíteni, hogy a fordított állományuk éppen mivel kommunikál. Mivel tanulnak shell szkriptet írni, találkoztak olyannak, amikor egy másik program futtatja a kódot, de az imperatív programozás tárgyban C++-ban programoznak, ami a gépikódra fordul (amellett, hogy persze kommunikál az operációs rendszerrel, de nem feltétlenül van rá szüksége).

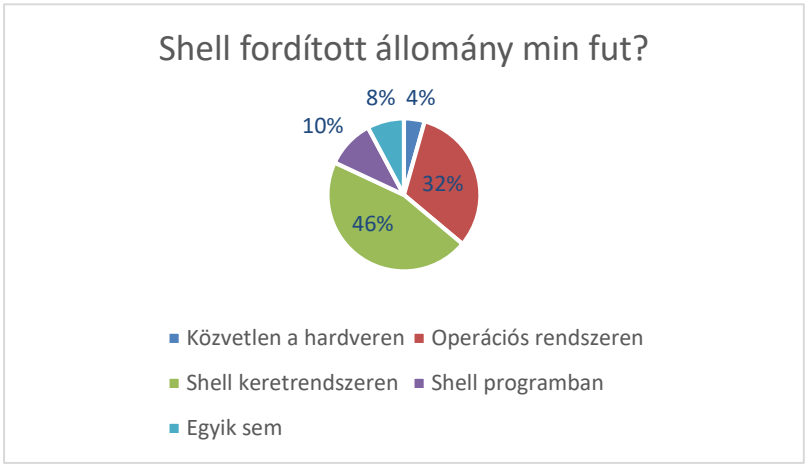


10. ábra: A nyolcadik (Igaz/Hamis) kérdés válaszai összesítve

Ez az eredmény (10. ábra) nem igazolja a hipotézist. A válaszok arra utalnak, hogy hallgatók értik mi történik a fordítás ideje alatt, sajnos azonban a következő kérdésre adott válaszok alapján egyáltalán nem vagyunk abban biztosak, hogy ez tényleg így van.

3.9. Ha egy shell fordított állományt (compilerrel) elindítasz, min fog futni? (közvetlen a hardveren, op rendszeren, shell keretrendszeren, shell programban, egyik sem)

Ez talán a legbeugratósabb mind közül. Maga a kérdés is értelmetlen, hiszen a shell szkriptet nem lehet fordítani, mert a shell program fogja értelmezni *valós időben* (ez a szkript definíciója). Amennyiben erre a hallgatók nem tudják a választ, az véleményünk szerint megvilágító erejű, hiszen ezt pont most tanulják (sőt, nagy részük ezen tárgy előadásán töltik ki a kérdőívet). Jelen állás szerint a hallgatók nagy része tud shell szkriptet futtatni, a tárgyat valószínűleg sikeresen fogják elvégezni, de az állományok futtatása elképzelhető, hogy fekete doboz maradt a számukra.



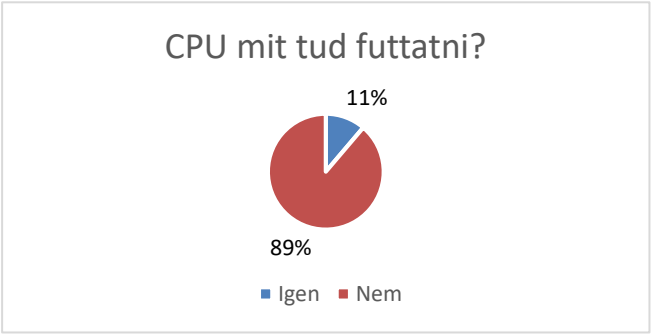
11. ábra: A kilencedik (feleletválasztós) kérdés válaszai összesítve

Az eredmények (11. ábra) szerintünk megdöbbentők. Igaz, a kérdés beugratós volt, de ha a hallgató még el is hiszi (beugrik a becsapásunknak), hogy létezhet shell fordítás, csak ő nem tud róla, akkor sem a shell keretrendszert (ahogyan a 11. ábra grafikonján látható) kellett volna választania (utalva az előző bekezdésre). Még a shell programban választ is meg lehetne magyarázni, hiszen itt előfordulhat, hogy a hallgató nem emlékszik mi a különbség szkript és program között, de arra igen, hogy a shell program értelmezi a kódját. A félév során a „Számítógépes rendszerek" előadást az egyik szerző tartotta (Illés Zoltán), a másik (Korom Szilárd) pedig bent volt az összesen (ahol tanulják ezt a témakört), így biztosan tudjuk, hogy az előadó többször is hangsúlyozta a megfelelő választ, azaz biztosan nem hallottak olyanról (egyetem keretein belül), hogy shell keretrendszer.

A gyakorlatokról tudjuk, hogy a többség tud shell szkriptet írni, de akkor mégis hogyan lehetséges, hogy nem tudják mi történik, amikor megfuttatják azt? Az eredményekből arra következtetünk, hogy azért, mert ha a *modul* szintjén maradnak (pl.: írj meg egy konkrét shell szkriptet; válaszold meg, mi az a keretrendszer), képesek megoldani a feladatot, mert ezt kérik tőlük számon. Egységesen az egészet azonban nem látják, nem tudják rendszerben értelmezni a megtanultakat.

3.10. Tud-e más kódot futtatni a CPU mint gépi kódot?

Itt a hardver és a szoftver kapcsolatára próbáltunk rákérdezni. Gyakran találkozunk azzal a kérdéssel középiskolások között, hogy a szövegszerkesztőben megírt program és az elektromos, áramkörökkel, bitekkel dolgozó hardver között mi a kapcsolat? Persze ha a hallgató a magas szintű nyelveknél marad, akkor ezzel a problémával talán soha nem találkozik. Teljesen vígan lehet kódot írni, tesztelni, futtatni, dokumentálni e nélkül s mégis, véleményünk szerint ez egy nagyon elemi kérdés, vagyis nem a rendszerre kérdez rá.



12. ábra: A tizedik (Igaz/hamis) kérdés válaszai összesítve

A 12. ábrán látható, hogy közel 90%-os a helyes válaszarány, vagyis ha olyan kérdést kap a hallgató, melyet tanult, s egyetlen egységre kell fókuszálnia (vagyis nem rendszer szinten, hanem az *egyedi összetevők* szintjén kell gondolkodnia), akkor azt meg tudják válaszolni.

4. Összegzés

A hipotézist, miszerint, ha diák ismer is egy-egy *modult*, nem tudja azt rendszer szinten igazolni, véleményünk szerint sikerült belátni. Az utolsó kérdés ilyen *modul* szintű volt, amire 90%-os aránnyal válaszoltak (12. ábra), míg a többiben (1 kivétellel) a nem helyes választ adták meg. A válaszok megítélésünk szerint további kérdéseket is felvetnek (például, hogy miért hiszik, hogy van „shell keretrendszer, ahogyan a 11. ábrán látható), hiszek meghökkentő arányok is jöttek ki.

Mivel a nem kifejtős kérdések esetén még kategorizálni lehetett a válaszokat, nem tartjuk kizártnak, hogy más szempontból is érdemes lehetne megvizsgálni az eredményeket. Például az érdekesség kedvéért a kérdőívben szerepelt a programozói tapasztalatukra is kérdés, ám a cikk keretében úgy éreztük nem fért bele a válaszok vizsgálata ennek tükrében.

A tanulmány elején igyekeztünk egy keretet, technológiát és módszert definiálni, mely alkalmas lehet a *modulok* összefűzésére, a rendszer szintű gondolkodás fejlesztésére. Az *IoT* rendszerek ugyanis olyan problémák megoldására, implementálására irányítják a diákokat, mely során muszáj több *modult* összeépíteniük, azokat egyidejűleg használniuk. Egy lehetséges tananyag pedig jól skálázható, fókuszpontja a csoporthoz és tanárhoz mérten beállítható. Annak igazolása, hogy ez valóban hatékony még nem történt meg, ám a jövőben erre is sor kerülhet.

Köszönetnyilvánítás

EFOP-3.6.3-VEKOP-16-2017-00001: Tehetséggondozás és kutatói utánpótlás fejlesztése autonóm járműirányítási technológiák területén – A projekt a Magyar Állam és az Európai Unió támogatásával, az Európai Szociális Alap társfinanszírozásával valósul meg.

Irodalom

1. *Informatika kerettanterv:* http://kerettanterv.ofi.hu (utoljára megtekintve: 2019.11.14)
2. Szlávi Péter, Zsakó László: *Informatikai kompetenciák – A valós világ modellezése.* In: Szlávi Péter, Zsakó László (szerk.) INFODIDACT 2013 ISBN:978-963-08-8387-0

3. Korom Szilárd: *Architektúrális gondolkodás fejlesztése valós idejű rendszerekkel.* In: Szlávi Péter, Zsakó László (eds.) INFODIDACT 2018 ISBN: 978-615-80608-2-0

4 Korom Szilárd: *IOT Systems in Education*. In: Marek, Smid; René, Bílik; Veronika, Stoffová (szerk.) XXXII. Didmattech 2019, Trnava, Szlovákia : Trnava University in Trnava Faculty of Education, (2019) p. 19

2. Universitat Pompeu Fabra Barcelon: *Subject syllabus – The Internet of Things.* https://www.upf.edu/pra/en/3376/22580.pdf (utoljára megtekintve: 2019.11.14)

3. University of Virginia – Angela Orebaugh, PhD: *Securing the Internet of Things.* https://collab.its.virginia.edu/syllabi/public/bcca51cc-9823-4bd4-8045-55b7b8f098cf (utoljára megtekintve: 2019.11.14)

4. Uppsala Universitet: *Syllabus for Internet of Things.* http://www.uu.se/en/admissions/master/selma/kursplan/?kKod=1DT094 (utoljára megtekintve: 2019.11.14)

5 Dr. Abonyi-Tóth Andor: *A micro:bitek felhasználási lehetőségei az oktatásban*. In: Szlávi Péter, Zsakó László (eds.) INFODIDACT 2018 ISBN: 978-615-80608-2-0

6 Módi József (2015): *A Raspberry Pi számítógép a gyakorlati oktatásban.*

7. Heizlerné Bakonyi Viktória, Illés Zoltán: *Valós idejű oktatási rendszerek*. In: Szlávi Péter, Zsakó László (eds.) INFODIDACT 2018 ISBN: 978-615-80608-2-0

8 Németh Tamás, Tornai Henrietta: *Scratch-től JavaScript-ig.* In: Szlávi Péter, Zsakó László (eds.) INFODIDACT 2018 ISBN: 978-615-80608-2-0

9 Bernát Péter: *Feladattípusorientált játékfejlesztés a Scratchben.* In: Szlávi Péter, Zsakó László (eds.) INFODIDACT 2017 ISBN: 978-615-80608-1-3