Okostelefonnal, WiFi-n keresztül vezérelt modellvasút 2.

Kovács Tamás okl. villamos üzemmérnök, ERICSSON Hungary

Az igazi előnye azonban abban mutatkozik meg, hogy ezeket a protokollokat a WiFi rádiós egységgel együtt minden mobil eszközben megtalálhatjuk. Az ESP8266 szintén implementálja mindezeket, ezért a kommunikáció a mobil eszköz és az ESP8266 modul között magas szinten, rendkívül egyszerűen megoldható, minden kiegészítő hardver elem nélkül is. Továbbá az általunk használt NodeMCU firmware tartalmaz egy komplett web szervert megvalósító szoftver blokkot is, amely segítségével, a mobil eszköz web böngészőjével, HTTP protokollon keresztül létesíthetünk két irányú adatkapcsolatot, célszerűen egy erre tervezett weboldal révén. Az adatok küldésére szerverre, illetve fogadására a szervertől (a szerver esetünkben az ESP8266 modul) a HTTP protokoll GET parancsát fogjuk használni. A webböngészők szintén a GET parancsot alkalmazzák a weboldalak letöltésére a webszerverekről. A fentiek alapján most nézzünk egy példát, hogyan tudunk egy webszervert indítani az ESP8266 modulunkon, amely automatikusan felkapcsolódik egy WiFi router-re, és DHCP-n keresztül kap a routertől egy IP címet pl.: 192.168.0.1, majd szolgáltat egy egyszerű weboldalt, ami jelen esetben csak egy üdvözlő szöveg: *Hello, NodeMCU!* Ennek kipróbálásához a **9. ábrán** látható lua szkriptet kell feltöltenünk a modulba és futtatni. A kód két fő részre bontható. Az első részben a WiFi, rádiós kapcsolatot hozzuk létre, a másodikban pedig a web szervert, amely már a weboldalunk tartalmát is magába foglalja, mindezt összesen 50 sor kóddal! A programkód elemzése a következő: A lua ún. objektum orientált programnyelv. Ez nagyon leegyszerűsítve az jelenti, hogy benne, logikailag összetartozó elemeket, olyan halmazként definiálunk, amelyek tartalmaznak adatokat, és műveleteket is képesek végezni az adataikkal. Ezen felül kommunikációs lehetőséget és adatcserét is tudnak biztosítani másik ilyen halmazokkal. Ezeket a halmazokat nevezzük objektumoknak. Az előbbi példa kódjában a „wifi" objektum már létezik, amin egy művelet elvégzését kérjük, szaknyelven meghívjuk a wifi objektum egy metódusát, amivel beállítunk vele egy adatot, – szaknyelven változót –, ami a wifi objektumot most arra állítja be, hogy STATIONAP, azaz router közbeiktatott hálózati STATION, illetve ún. direkt, Access Point AP, hibrid módban működjön. A hibrid mód itt azt jelenti, hogy mindkét mód egyszerre aktív, tehát az ESP modulhoz routeren keresztül, illetve router közbeiktatása nélkül, direkt módban is tudunk kapcsolódni a mobil eszközzel. Az objektum neve után ponttal elválasztva adjuk meg a metódus nevét, a zárójelben pedig a beállítani kívánt változó értéke van megadva: wifi.setmode(wifi.STATIONAP). A (8-16) sorokban az AP, direkt módhoz tartozó beállításokat adjuk, meg egy cfg nevű adatstruktúra változó segítségével. A következő két sorban (18-19) annak a WiFi routernek az SSID-ját, illetve jelszavát állítjuk be, amelyhez az ESP8266-os modullal csatlakozni szeretnénk. Úgy is mondhatjuk, a wifi objektum, az ESP8266-modul WiFi áramkörének szoftveres leképezése, szakszóval reprezentációja. A wifi objektumból egy darab lehet, és azt a rendszer automatikusan létrehozza, mi csak a tulajdonságait, paramétereit állíthatjuk be metódusokon keresztül. A második részben (38-50) sorok, arra látunk példát, mikor egy objektumból mi hozunk létre egy új objektumot, amely révén a webszervert valósítjuk meg. A net objektum, hasonlóan a wifi objektumhoz automatikusan létrejön. A net objektum createServer nevű metódusával hozhatunk létre egy Server objektumot. Ilyen szervert, elvileg többet is létrehozhatunk, ezért ezt, a most létrehozni kívánt szervert egyedi névvel kell ellátni, neve ebben az esetben srv. Úgy mondjuk, hogy az srv, a Server objektum egy példánya. Létrehozásához szükséges kód pedig a következő: srv = net.createServer (net.TCP). A zárójelben levő paraméter utal arra, hogy TCP/IP szervert szeretnénk létrehozni, a weboldal kiszolgáló részére. A további kódsorok a létrehozott, új szerver példányt aktiválják, a srv:listen (... ún. csoportos paraméter beállítások alkalmazásával, aminek lényege több metódushívás és paraméter beállítás összevonása a kódban. Eredmény képen egy olyan TCP/IP szervert indítottunk a rendszerben, (ESP8266 modulban), amely a 80-as TCP-porton (alpértelmezett HTTP port) figyeli (szószerinti fordítás szerint: listen / hallgatja) a bejövő kéréseket, és ha van ilyen, akkor egy ún. TCP/IP socket-et /sck/ (csatornát) épít fel a kliens felé és elküldi (client:send) neki a weboldal tartalmát HTTP protokoll szerint, amit a kliens, a mi esetünkben egy mobil eszköz, mobiltelefon vagy tablet, a webböngészőjé-

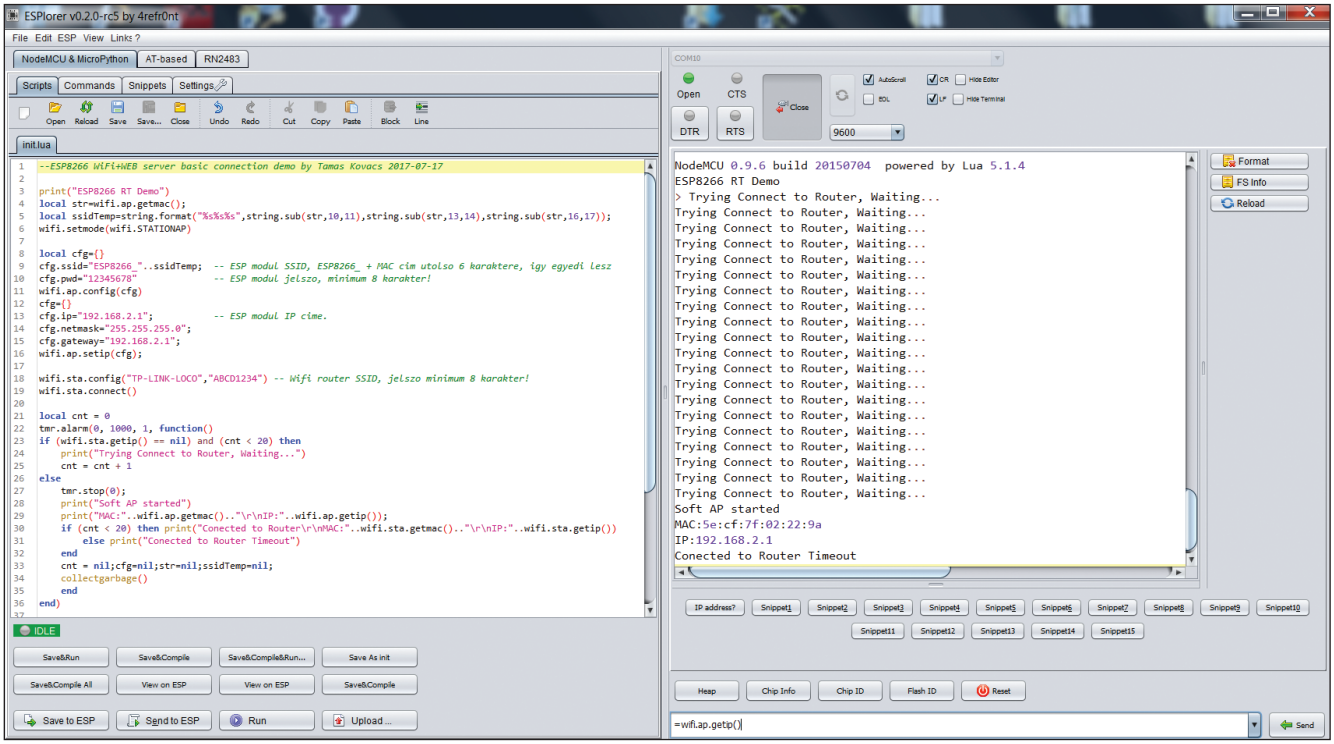
ben megjelenít a felhasználónak. Az srv server, indítása után a háttérben futó, ún. programszálként (thread) viselkedik, ami azt jelenti számunkra, hogy a programunk további részében számíthatunk annak szolgáltatásaira, amig az srv.close() paranccsal le nem állítjuk.

Nézzük most meg, hogyan tölthetjük fel, és próbálhatjuk ki ennek a lua szkriptnek a működését a gyakorlatban. A lua programok feltöltéséhez, illetve a teszteléséhez az ESPlorer nevű ún. integrált fejlesztői környezetre (IDE Integrated Development Environment) lesz szükségünk. Ez egy Java-ban írt program, a zip file-t kicsomagolva, az *esplorer.jar* file-t futtatva elindul, külön installálni nem kell. Az IDE futási képe a **10. ábrán** látható. Csatlakoztassuk a modul USB-TTL konverterrel a számítógéphez, Nyomógomb legyen kiengedett állapotban, GPIO0 láb H szinten.

Az *ESPlorer IDE* bal oldali ablakában egy text editor található, amiben a lua szkripteket lehet megnyitni, editálni, a jobb oldali ablak gyakorlatilag egy soros terminál, ami az ESP8266-modullal való kommunikációra szolgál. A COM port és adatsebesség 9600 b/s beállítás után az OPEN gombra nyomva lépjünk kapcsolatba az ESP8266 modullal. A text editorban nyissuk meg az *init.lua* file-t, ami a 9. ábrán is látható lua kódot tartalmazza. Írjuk át a kódban (18. sor) a WiFi routerünk SSID-jét, ill. jelszavát és mentsük el. *Fontos!:* Az ESP modul alapértelmezetten WPA2 titkosítást használ a WiFi rádiós hálózaton, tehát a routeren is ezt kapcsoljuk be és minimum 8 karakter legyen a jelszó hossza! Ezután a SAVEto-ESP gombbal töltsük fel az ESP modulra. Az ESP modul mindig az *init.lua* nevű file-t fogja először futtatni reset után. A IDE-n a Reset gomb megnyomása után a szkript futni kezd, majd megpróbál csatlakozni a WiFi routerhez, amennyiben ez sikerül kiírja: Connected to Rouer és az IP címet, amit a router kiosztott számára. A WiFi router menüjében megtaláljuk az ESP modult, a DHCP clients menüpontban, itt is megtudhatjuk milyen IP címet osztott ki a router az ESP modul számára. pl.: 192.168.100.1.

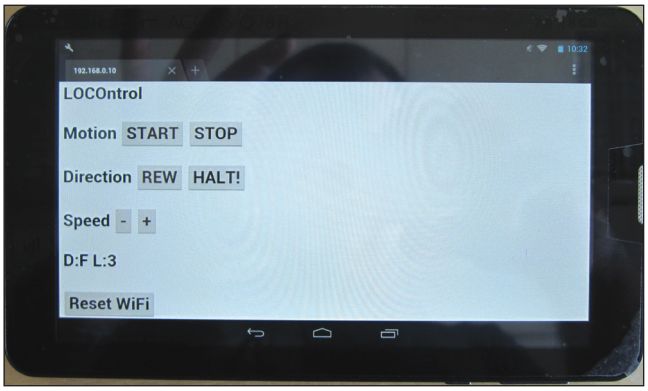
A szintén a rouerhez csatlakozó mobil eszközön nyissunk meg egy böngésző ablakot, majd ennek címsorába írjuk be az ESP modul IP címét. A böngésző megjeleníti az ESP modulban tárolt weboldalt. A modul IP címét az ESP modul terminálból is kiírathatjuk, a: print('IP: ',wifi.sta.getip()); parancs segítségével. A szkript is minden futtatáskor ki fogja írni a terminálra az aktuális IP címet. (WiFi router hiányában, az ESP modulhoz, direkt módban is csatlakozhatunk, jelen esetben a 192.168.2.1 IP címmel.)

A *mozdony teljes programkódja* négy lua file-ból áll: *loco.lua, setwifi.lua, wifi.lua, init.lua.* Ezeket nyissuk meg az IDE-ben, majd sorban töltsük fel ezeket az ESP modulra, ügyelve, hogy más file-ok ne legyenek rajta, ha mégis vannak, azokat először töröljük le. Futtassuk a programot, majd első lépésként direkt módban kapcsolódjunk a mozdonyhoz, a WiFi hálózat neve (SSID): LocoC, jelszó: 12345678. Az IP cím: 192.168.4.1, ezt írjuk be a böngésző címsorába. A böngészőben, egy konfigurációs weboldal jelenik meg **(11. ábra).** Itt



10. ábra

11. ábra



12. ábra

kell megadnunk a Wifi router SSID-jét és jelszavát, amit használni kívánunk a többi mozdonyunk ill. a terepasztal vezérléséhez. Az adatokat a program egy kétsoros szöveg file-ba menti, az ESP modulra, és ezen adatok felhasználásával a következő indulásnál már az itt beállított WiFi router-hez fog kapcsolódni a mozdony automatikusan. Természetesen ezen beállítások a fő oldalról is elérhetőek és szükség esetén megváltoztathatók. A már beállított adatokkal rendelkező mozdony, a Wifi router által kiosztott, IP címen, a **12. ábrán** is látható weboldalról lesz irányítható. A weboldalon a Start gombbal a mozdony elindítható, sebessége a +, – gombokkal növelhető, ill. csökkenthető. A REW gombbal a menetirány változtatható meg. Ilyen esetben a vonat lelassul, megáll majd az ellenkező irányba fokozatosan felgyorsul arra a sebességre, amin az előző irányban haladt. A HLT! gomb azonnali megállást vált ki. A weblapról az aktuális sebesség és irány is leolvasható. A Reset WiFi gombbal beléphetünk az előbb már ismertetett konfigurációs weboldalra. Láthatjuk, hogy amennyiben több ilyen WiFi-s mozdonyunk is van, a WiFi router mindegyiknek más IP címet fog kijelölni. A WiFi routereknek van egy olyan szolgáltatása, hogy egy MAC addresshez mindig ugyanazt az IP címet rendelik hozzá. A MAC cím minden ESP modulnak egyedi, ezért érdemes ezt a szolgáltatást igénybe venni. Az adott mozdony MAC címét a konfigurációs weboldalról is leolvashatjuk. Így elérhetjük, hogy egy mozdony mindig ugyanazon az IP címen legyen elérhető. Ez kényelmesebb kezelést biztosít számunkra ugyan, de így is annyi böngészőablakot kell megnyitnunk, ahány mozdonyt kívánunk egyszerre irányítani, persze több mobilról is irányíthatjuk a mozdonyokat egyszerre. Itt mutatkozik meg a WiFi és az IP technológia előnye, mert ezt hagyományos rádiótávirányítás esetén csak többcsatornás eszközök összehangolt rendszerében tudnánk megvalósítani. A webes mozdony vezérlés motorja a *loco.lua* fileban található kód. A kódot a **13. ábrán** találhatjuk, sok ismerős elemet találunk benne pl. a web szerver létrehozását, weboldal felépítést stb. Egy új elemet érdemes kiemelni, ami a HTTP szerver része, ez pedig a HTTP protokoll GET parancsának implementálása. A weboldalon keresztül ezzel a parancsal tudunk adatokat elküldeni a web szervernek, ill. lekérni onnan. A programunk vezérlési algoritmusának kulcs eleme a GET parancs. A HTTP gyakorlatilag szöveg alapú protokoll. Amikor a weboldalon egy vezérlőgombot megnyomunk, a böngésző egy karakter sort, szöveget küld a webszerver, a mi esetünkben az ESP modult tartalmazó mozdony felé. A szerver oldalon a html szövegből ún. parse-olással (szövegmintára való szűréssel) megkeressük a weboldalon lenyomott gombhoz tartozó transfer változó(pin) értékét (6-17 sorok), majd a végrehajtást „if” elágazás feltételekkel átadjuk a megfelelő kezelő funkciók számára (24-43 sorok).

Példaként nézzük meg egy vezérlési folyamat algoritmusát, a többi is teljesen hasonlóan működik. A START gomb megnyomásakor a vezérlési változó pin=„START” lesz. Ilyenkor a szerveroldalon a 24. sorban levő szűrőfeltétel, „igaz” esetben, a start_handler() nevű függvényt hívja meg (78. sor). A start_handler funkció működése: A „level” nevű globális változó a mozdony sebesség adatát tárolja, az egész START folyamat csak akkor végrehajtható, ha ez a változó 0 értékű, vagyis a mozdony áll (80. sor). A „direction” nevű szintén globális változó a mozdony irány adatát tárolja 0 értékű előremenet, 1 értékű hátramenet esetén. Az aktuális menetirány vizsgálata alapján (84. sor), ha 0 az érték beállítjuk az előremenethez tartozó PWM objektum paramétereket (87-90 sorok). Majd vezéreljük a menetfény LED-eket a menetiránynak megfelelően (92-93 sorok). Amennyiben a menetirány ellentétes, a (100-106 sorokban) található hasonló kód fut majd le. Végül a deltaspeed_handler_up() (110. sor) funkció kerül meghívásra, ami a PWM jel kitöltési tényezőjét folyamatosan változtatva (156-167 sorok) emeli a kívánt fordulatszámra a meghajtómotor fordulatszámát, ezzel gyorsítva fel a mozdonyt a megfelelő sebességre.

Az alkamazás során szükségünk van arra, hogy a mozdonyból az aktuális sebesség, ill. irány adatokat le tudjuk kérdezni. Például akkor, ha böngészőt bezárjuk és az adatkapcsolat megszakad, majd új böngészőt nyitunk meg és újra kapcsolódunk az egyik mozdonyhoz. Ezért a weboldalba injektáljuk a következő ún. xml struktúrát (61. sor): <tr at="le">D:"..directionp.." L:"..levelp.."</tr> Ennek az adat beágyazásnak az az előnye, hogy a weboldalon minden egyéb formázás nélkül kiírja a benne foglalt adatokat, jelen esetben pl. **D:F L:3** 12. ábra. Ez esetben az xml adatstruktúra így néz ki:

<tr at="le">D:F L:3</tr>

A mobilalkalmazásunkban pedig szintén előnyös lesz, mert az ilyen, xml formátumú adatstruktúra rendkívül egyszerűen feldolgozható, ezért ebben az esetben is hatékonyan tudjuk alkalmazni web alapú adatlekérések esetében.

A mobilapplikáció az Adobe AIR elnevezésü fejlesztői környezetben lett kivitelezve. Ennek egyik nagy előnye, hogy itt fejlesztett alkalmazás fut Google Android, Apple iOS, ill WebOS operációs rendszerű mobileszközökön egyaránt. További előnye, hogy a programozási technikája lényegesen egyszerűbb, mint mondjuk a natív Android fejlesztőkörnyezetben használt Java alapú kódolás. Szerencsére magyar nyelvű szakirodalom is rendelkezésre áll, *Fehér Krisztián: Androidos szoftverfejlesztés alapfokon*. Ez a könyv részletesen foglalkozik az Adobe AIR mobil programozási témával. A mobilapphoz tartozó forráskód és egyéb hasznos infók azzal kapcsolatban a LOCONTROL_MOBILEAPP nevű könyvtárban találhatóak. Az Adobe AIR hivatalos fejlesztőkörnyezete az Adobe FlashBuilder nevű IDE, ez azonban fizetős szoftver, ugyan egyhónapos próbaverziót le lehet tölteni, mégis az ingyenesen is használható, majdnem egyenértékü FlashDevelop nevű IDE-t javaslom azoknak, akik szeretnének effajta mobilapp fejlesztéssel foglalkozni. A mobilapp Androidra lefordított, installálásra kész változata a LOCONTROL_ANDROID könyvtárban található. A *LOCOnew001.apk* nevű file-t másoljuk a mobileszközre, file manager segítségével, majd telepítsük. A telepítésnél az Android figyelmeztet, hogy az Adobe AIR környezet is szükséges a programunk futtatásához és azt is telepítenie kell, ezt engedélyezzük neki! Az mobilapplikáció „LOCONTROL”, SQLite adatbázisban tárolja a mozdonyokhoz tartozó adatokat, pl. az IP címet is. Ezeket a beállításokat a mobilapp adatbázisában menti, és a rendszer újraindításakor automatikusan betölti. Az első indítás alkalmával hozzunk létre egy adatbázist. A kezdőképernyőn nyomjuk meg a „+” ikont, majd a következő képernyőn a „ceruza” ikont, majd ezt követően megjelenő képernyőn a CREATE DB gombot, majd az OK-t. Az alkalmazásból lépjünk ki, majd indítsuk újra. A mozdonyokhoz képeket is hozzárendelhetünk. A kép file-okat file managerrel a belső tárolón létrehozott Locontrolpics könyvtárba mentsük, majd a mozdony adatlapján „Edit model” képernyőn a Picture file path: hoz adjuk meg az útvonalat: */Locontrolpics/mozdonykep1.jpg* például, majd Save gombbal mentsük el. A modell sebességét a mobilapplikációból nem csak a -/+ gomb nyomogatásával, hanem egy tolópotméterhez hasonló csúszkával is szabályozhatjuk. A vezérlő menüből a pipa ikon megnyomásával lekérdezhetjük az adott mozdony aktuális sebességét és haladási irányát, ezzel aktualizálva a kijelzett értékeket a képernyőn.

Kérem a kedves olvasót, hogy az itt közreadott alkalmazást ne tekintse professzionális, kereskedelmi célú projektnek. A cél elsősorban a téma iránt való érdeklődés felkeltése, ill. ismeretterjesztés volt. A kidolgozásnál és tesztelésnél ügyeltem arra hogy minden működőképes legyen, azonban bőven lenne mit még fejleszteni, tökéletesíteni rajta. Ezzel kapcsolatban minden érdeklődőt bíztatnék a továbbfejlesztésre, illetve más projektek indítására a témával kapcsolatban. Felmerülő kérdéseket, visszajelzéseket szívesen fogadok a következő e-mail címen: *kovtam112@gmail.com*

A Rádiótechnika HAM-bazár egységcsomag-kínálata

LP1 LED-pakk (30 db 3 mm-es kfl. színű, gründolt LED)	**200 Ft**
LP2 LED-pakk (15 db 3 mm-es + 15 db 5 mm-es vörös LED)	**200 Ft**
LP3 LED-pakk (25 db kfl. színű, extra-forma, gründolt LED)	**200 Ft**
VP1 varikap-pakk (6 db 2V104D, 6 db 2V110V, 1 db B B112, 4 db BB329, 8 db BB521)	**700 Ft**
ZP1 zener-pakk (70 db klf. Z-dióda, 2 V ... 120 V közötti értékek)	**700 Ft**
FP1 FET-pakk (14 db 2SK168D, 8 db 2N3820, 8 db J202)	**1500 Ft**
MP1 MOSFET-pakk (6 db BF961, 6 db BF964, 18 db BF982)	**1200 Ft**
TP4 tranzisztor-pakk (60 db klf. npn, pnp, Si és Ge kistelj.)	**500 Ft**
TP5 tranzisztor-pakk (140 db klf. npn, pnp, Si és Ge kistelj.)	**1000 Ft**
TP6 tranzisztor-pakk (25 db klf. npn, pnp, Si és Ge nagytelj.)	**1000 Ft**
TP7 tranzisztor-pakk (60 db klf. npn, pnp, Si és Ge nagytelj.)	**2000 Ft**
DAP1 darlington-pakk (2 db BDX33C, 2 db BDX34C, 6 db BC516, 6 db BC517)	**700 Ft**

Bpest. XIII., Dagály u. 11. I. em. H-P. 09-14 ó, Cs. 09-17 ó vagy postai utánvéttel.
Tel.: 239-4932/36, 239-4933/36 1374 Bp., Pf. 603 hambazar@radiovilag.hu pk1