Mobility Okosparkoló

A beadandó során csapatunk célja egy okos parkoló kialakítása volt. Az alapelgondolás a be- és kimenő autók számlálása, és ezzel a parkoló foglaltságának megadása volt. Ezt a munka során e-mailküldő és tweetelő funkcióval bővítettük, hogy a parkolni vágyók folyamatosan értesítve legyenek a szabad helyekről.

1. Lépés – Eszközök, alkatrészek

A megvalósításhoz az első lépés megfelelő eszközök beszerzése volt, melyek a következők:

● Raspberry Pi 3

https://www.raspberrypi.org/products/raspberry-pi-3-model-b/

● Ultrahangos távolságmérő HC-SR04

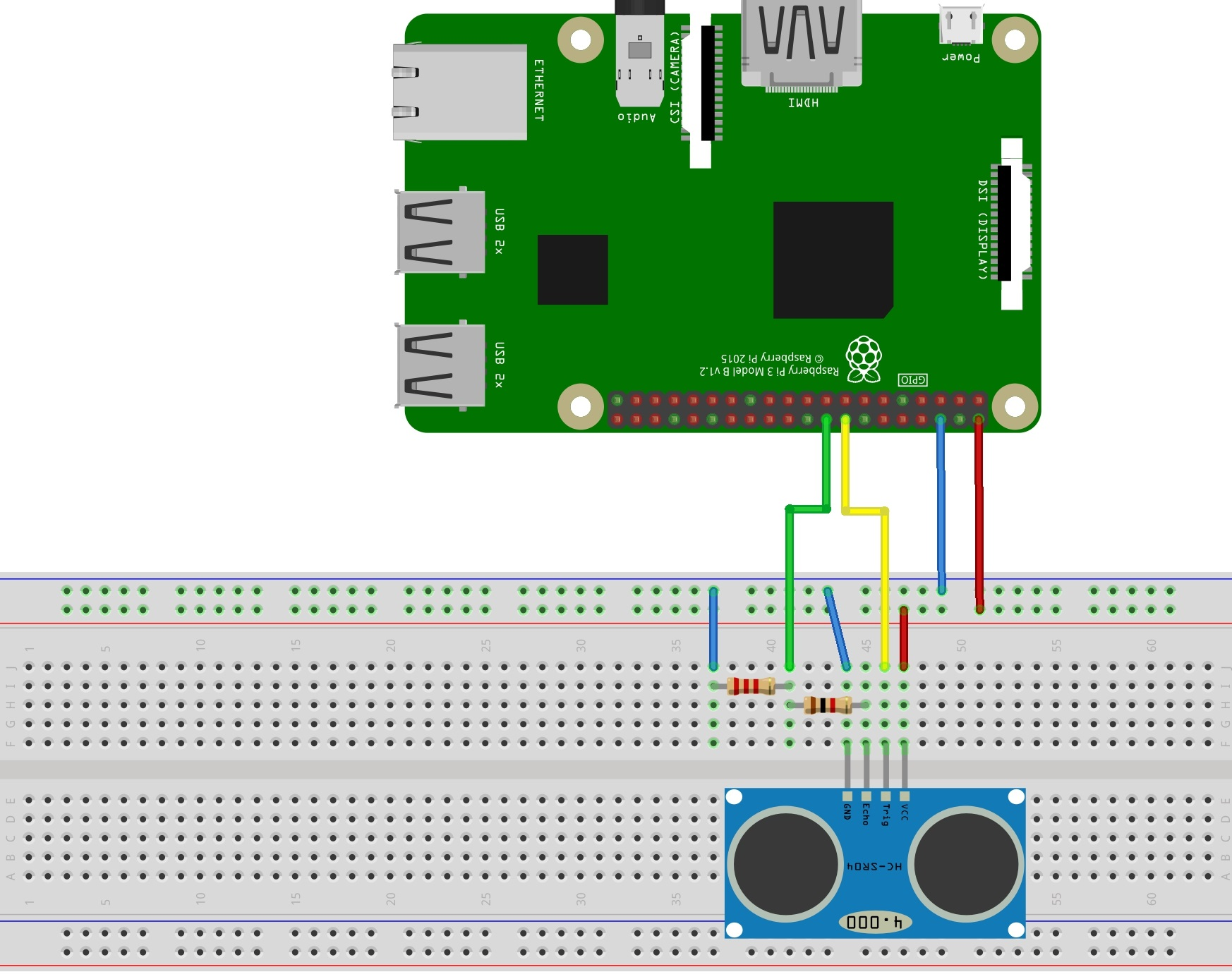
http://hobbielektronikabolt.hu/spd/HCSR04/Ultrahangos-tavolsagmero-HC-SR04

● szenzoroknak külön dashboard, illetve ahhoz szükséges kábelek, 1000 Ω-os ellenállások

● tápforrás – Powerbank

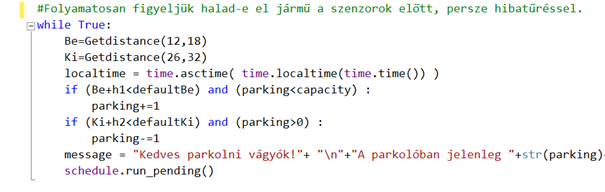
2. Lépés – Raspberry Pi és szenzorok

Második lépésként összeállítottuk a hardveres részt, vagyis bekötöttük a 2 ultrahang szenzort és feltelepítettük az operációs rendszert Raspberry Pi-re. Ezek után, hogy a működésüket is ellenőrizzük, pár sornyi kóddal teszteltük azt.



3. Lépés – Alapkód megírása

Következőleg az alap kódot írtuk meg. Az elgondolás alapja az objektumok (autók) érzékelése volt. Ezt úgy valósítottuk meg, hogy amennyiben egy a ki- vagy bemenő jármű érkezik a megfelelő szenzor előtt elhaladva az alaptávolság lecsökken. Attól függően, hogy a kimenő, vagy a bemenő szenzor érzékeli csökkenti, illetve növeli a bent tartózkodó autók számát.

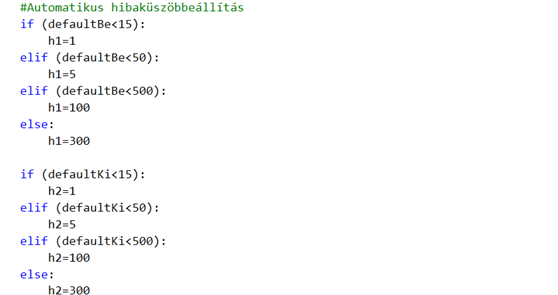


4. Lépés – Tesztelés

A munka során minden kódrész megírása után teszteltünk, hogy egy esetleges hibát azonnal javítani tudjunk, illetve, hogy könnyen kiszűrhessük, melyik részben történt hiba.

Az alapkód tesztelése során több paraméterváltoztatást kellett végeznünk. Ilyenek voltak az eredetileg megadott hibatűrés alapértékei, illetve a szenzorok mérései közötti időintervallumok.

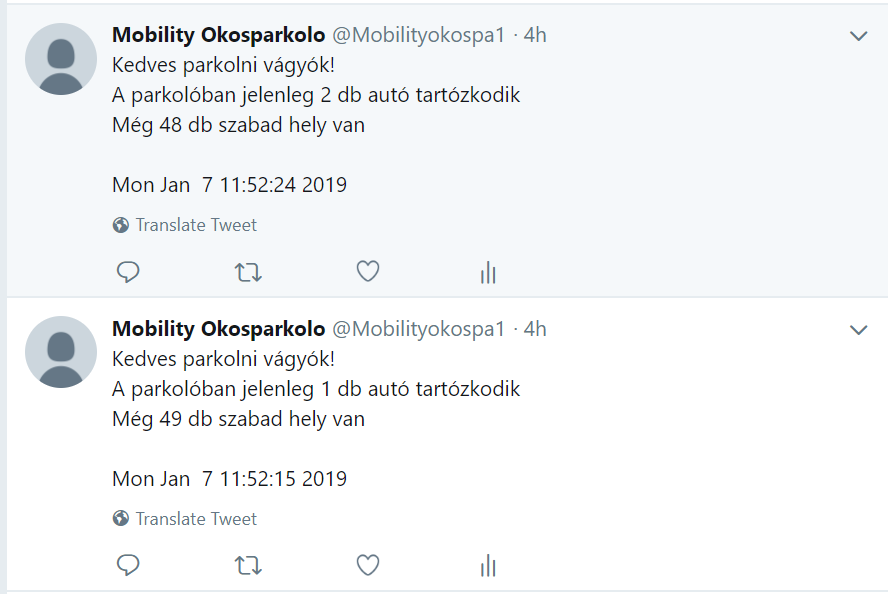
A hibatűrés értékeit eredetileg fixen állítottuk be, viszont tekintettel arra, hogy a rendszert csak felállítani kelljen egy esetleges helyváltoztatás esetén, az első tesztelés után if szekvenciát használtunk, több megadott értékkel.



5. Lépés – Extra funkciók

Az 5. lépésben egyrészt extra másrészt tájékoztató jelleggel további funkciókat vezettünk be. Ezek közül először egy tweetelő, majd egy e-mailküldő részt írtunk meg.

Mindkettő esetében 30 percenként érkezik az értesítés, ami a foglaltságról ad tájékoztatást.



6. Lépés – II. Tesztelés

A következő lépésként az újonnan hozzáadott elemek működését teszteltük.

Itt a twitter résznél fedeztünk fel egy olyan hibát, ami a kód működésének leállásához vezethet. A Twitter nem engedi ugyanis a duplikált posztokat, tehát ha 30 perce is 50 volt a foglaltság, és most is az, hibát jelez, és a program nem fut le. Ezt egy egyszerű időkiírás segítségével valósítottuk meg, ami a problémát meg is oldotta és a posztok hitelességét és segít javítani.

7. Lépés – Főpróba

Utolsó lépésként a teljes összeállított rendszert kellett tesztelnünk. Ezt a Mobilis parkolójában néhány önkéntessel végeztük el. Ez esetben is szükség volt egy kis paraméterezésre, amivel megfelelően tudtuk számolni az áthaladó járművek számát.

A tesztet 3 önkéntes segítségével végeztük, tehát a kocsik elhaladásával a foglaltság nőtt, és amikor azok a parkolót elhagyták, a foglaltság visszacsökkent az eredeti értékre. A teszt alapján megállapítottuk, hogy 1.5-ös set up time (sleep time) volt a legoptimálisabb arra, hogy az elhaladó kocsikat megfelelően érzékelje.

Az alábbi képen a tesztkörnyezet (Mobilis parkoló) látható



A következő képen pedig a rendszer felállítása



Ahhoz, hogy a munkánkat a vizsga során is be tudjuk mutatni, egy kartondobozból készített ,,pályát” hoztunk létre, ahol a szenzorok fixen rögzítésre kerültek.