**Projektmunka 2**

GKNB\_INTM005

**Szobanövény locsoló rendszer**

**Benyó Ádám (XW5W82)**

**Bogdán Patrik (ZXAAE3)**

**Gábriel Balázs Gábor (UTDWSP)**

**Rácz Dávid Richárd (E7PWEK)**

**Mérnök Informatikus BSc szak**

**2021/2022/1**

**Csapatelosztás**

Szinte az összes fejlesztéssel és szereléssel kapcsolatos napot együtt töltöttük, heti rendszerességgel brainstormingok formájában. Abban az esetben, ha emberekre kellene bontani, akkor a következőképpen alakult az eloszlás:

|  |  |
| --- | --- |
| Benyó Ádám | * kapcsolattartás * dokumentáció, prezentáció * web design * koordinálás |
| Bogdán Patrik | * frontend * backend * folyamat elkészítése * meglévő rendszerek felmérése |
| Gábriel Balázs Gábor | * szerelés * eszközök beszerzése * eszközök tesztelése * prezentáció |
| Rácz Dávid Richárd | * backend * tesztelés * web design * folyamat elkészítése |

Az összeülések végén egyeztettük az elért eredményeket és lefektettük a következő alkalom elvárásait:

* meglévő rendszerek felmérése
* eszközök beszerzése
* eszközök tesztelése
* szerelés
* kódolás
* web kialakítása
* dokumentáció, prezentáció elkészítése

**Az ötlet**

Egy egyszerű rendszer elkészítése volt a célunk, ami egy lokális weblapon keresztül vezérelhető öntöző, amely alap funkciókként képes lekérdezni a nedvesség állapotát, egy fájlból képes beolvasni, hogy mikor volt az utolsó locsolás és gombnyomásra végrehajtani 1 öntözést. Ezt követően a felhasználótól kérdez egy mennyiséget, hogy mennyit töltött a víztartályba, ezzel pedig képes egy automata öntözést elindítani és megszakítani. A víz mennyiségének a lényegét az ahhoz tartozó részben szeretnénk kifejteni.

A választásunk azért erre esett, mivel sok embertől lehetett hallani, hogy az időhiány vagy a távollét miatt kiszáradásra vannak ítélve a házban fellelhető növények.

**Alkatrészek beszerzése**

A feladatunkkal a nyár második felében kezdtünk foglalkozni, az alkatrészek egy része már rendelkezésünkre állt (előkészített Raspberry, kábelek, ellenállások), így a hiányzó elemek beszerzésével kezdtünk. Hiányzó elemeink közé tartozott egy 5V-os, egy csatornás relé modul, ami időközben kiderült, hogy nem megfelelő számunkra, de szerencsére raktáron elérhető volt egy 3V-os is és ezt a hibát nagyon gyorsan tudtuk javítani. Ezen kívül egy 12V-os víz szivattyú, amely egyaránt használható merülő pumpaként is (IP68 szabványos), ehhez szükség volt két 8mm-es belső átmérőjű szilikon vízcsőre, hogyha úgy adódna, akkor vízen kívül is használható legyen a pumpa és egy talajnedvesség érzékelő szenzorra. Az első problémánk ez utóbbival adódott, mivel jelenleg is rendelés alatt áll. Az első rendelésünk nem érkezett meg, a második körben kaptunk egy olyat, melynek egyik PINje hiányzott, de szerencsére a harmadik üzletből érkező szenzor jól működött.

**Az összes felhasznált elem**

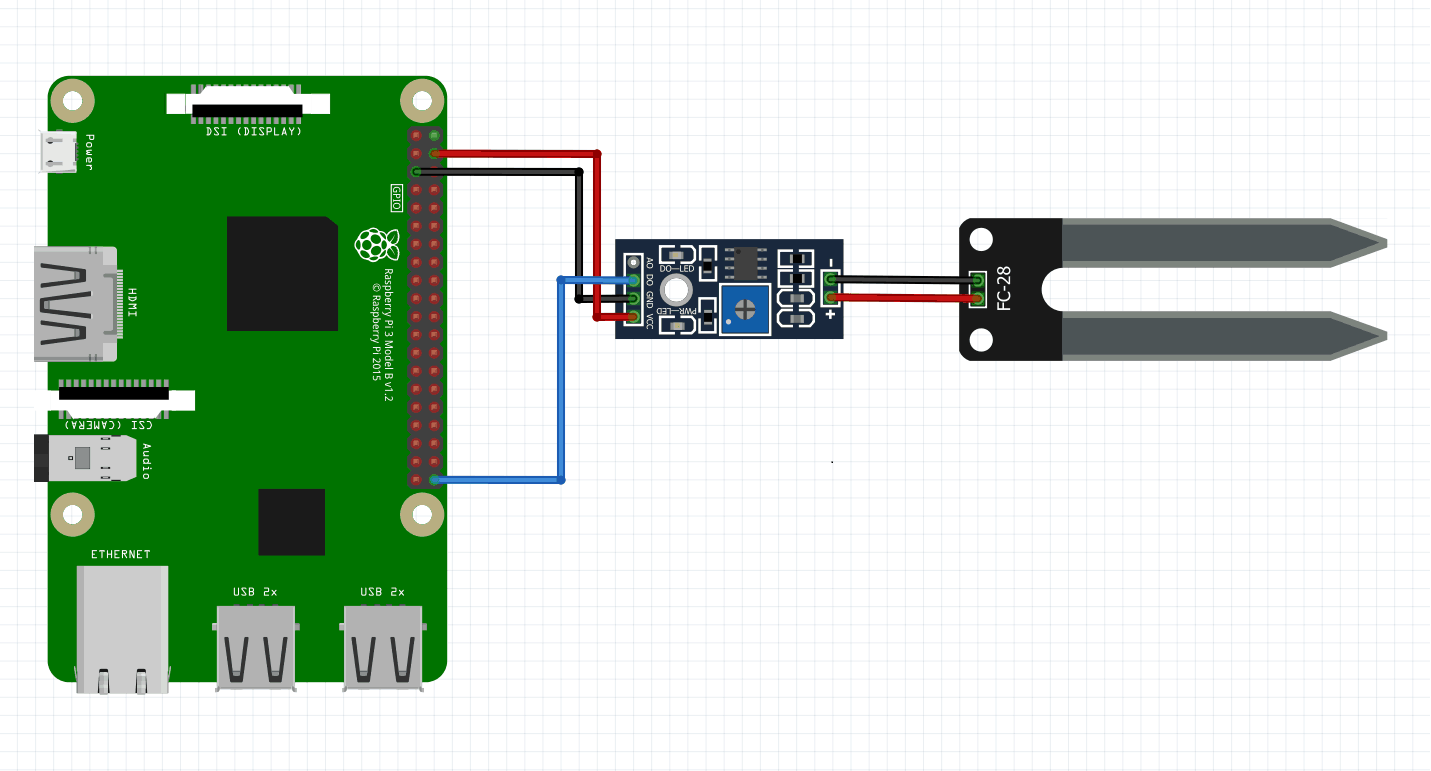
* Raspberry Pi 3 model B+ (SD kártya + hűtő)
* [Nedvesség szenzor SA-27](https://www.hestore.hu/prod_10035554.html)
* [3V-os, 1 csatornás SRD-DC03V-SL-C relé](https://www.hestore.hu/prod_10042172.html)
* [12V-os merülő szivattyú AD20P-12V-PUMP-C](https://www.hestore.hu/prod_10041524.html)
* tápellátás a szivattyúnak és a Raspberrynek
* [8mm-es belső átmérőjű szilikon cső](https://www.hestore.hu/prod_10042175.html)
* kábelek: Female to Male, Female to Female
* virág és edény a víznek

**Felhasznált fejlesztői eszközök**

* Python
* HTML

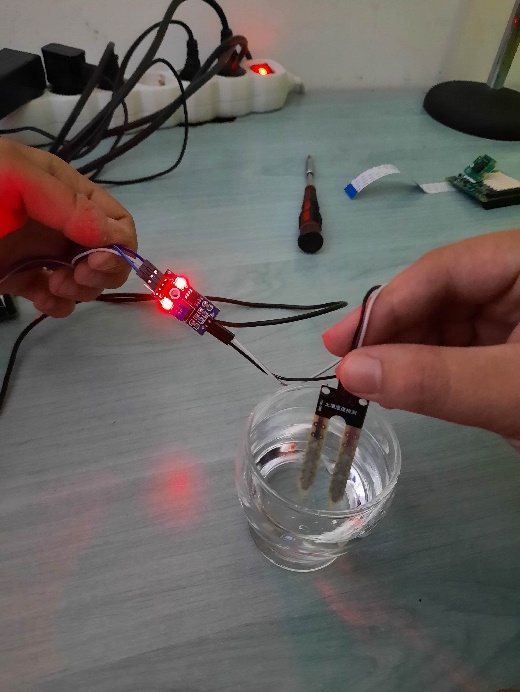
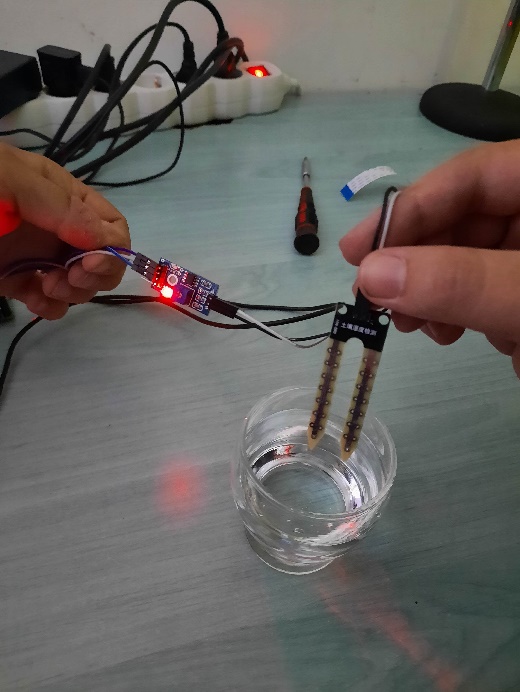
**A szerelés**

Első esetben le kellett ellenőriznünk működik-e minden, amit beszereztünk. A nedvesség szenzorral kezdtük. Ehhez egy interneten fellelhető bekötési rajzot és kódot használtunk, hogy megfigyeljük a szenzor érzékelőinek effektivitását. Meglepő módot a szenzor felülmúlta elvárásainkat és a legkisebb nedvesség változásra is jelzett. Ezek a sikerek megfigyelhetőek voltak a szenzorra csatlakoztatott átalakítón és a program futása után a kimeneten is. Mivel ez egy teszt volt így a kódban csak a kiírt szöveget és a bekötési útmutatón a PIN-ek használatát módosítottuk az általunk preferált bekötéshez: esetünkben ez csak annyi, hogy a jelhez a Raspberry egy másik csatornáját használtuk.



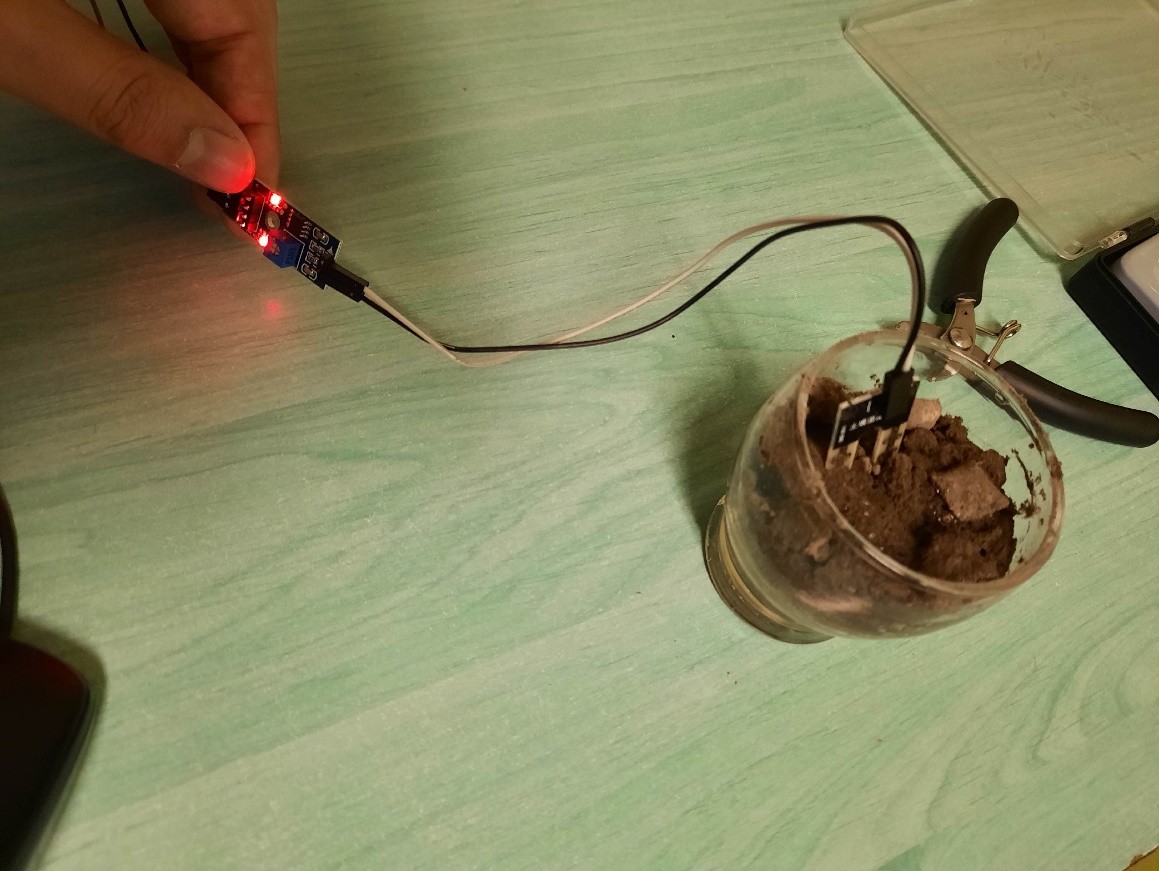
1. ábra: Szenzor teszt

A kimeneten a *Vizes/Nem vizes* üzenetet kaptuk vissza, az alábbi képeken pedig az átalakító visszajelzése látható. A szenzor képes analóg és digitális jel visszaadására is, de mi öntözés szempontjából megelégedtünk az analóg jellel, mivel csak arra vagyunk kíváncsiak, hogy nedves-e a talaj. A digitális jellel pontos értéket is kaphatnánk, de ez nem befolyásolna semmit, mivel egy bizonyos érték alatt automatikusan öntözne, és ahogy az érzékelőkhöz érne a víz, azonnal abbahagyná az öntözést. Alap esetben 1 LED világít, ami a bekapcsolt állapotot jelzi, nedvesség érzékelése esetén pedig felvilágít, az alap fénnyel ellentétes oldalon láthat LED is.



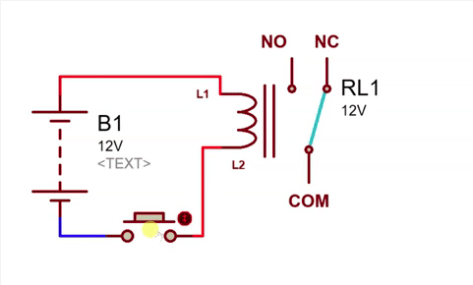
2. ábra: Szenzor teszt

Ezt követően élesben is leteszteltük, száraz földbe helyeztük a szenzort és mivel kis edényről volt szó így lassan adagoltuk a vizet. Itt adódott az első probléma, mivel a szenzor érzékelői nem csak az alján helyezkednek el, hanem teljes felületén, így ahogy a víz hozzáér bármely részéhez, az nedvesnek titulálja a földet és abbahagyja az öntözést. Arra a megoldásra jutottunk, hogy az öntözési pont az edényünk egyik felében, míg az érzékelő az edényünk másik felében fog elhelyezkedni. Amennyiben a cserép méretéhez képest, arányos nagyságú növény van a földben így ez a megoldás jól fog működni. A lényege annyi lenne, hogy mire a nedvesség elér a cserép másik feléhez, ahol a szenzorunk is található, addigra kellően vizes lesz a talaj a növényünknek.



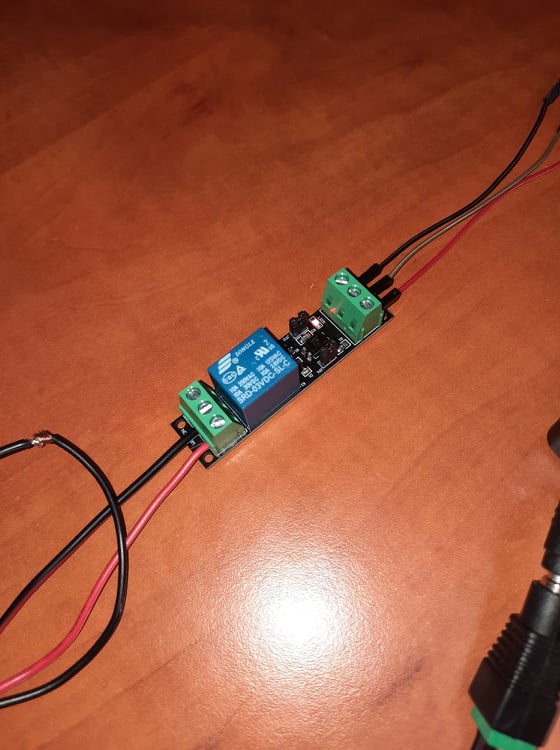
3. ábra: Valós teszt

Ezt követte a pumpa tesztelése már a véglegesnek szánt kódból. A pumpa működésére szolgáló részletet használtuk. A pumpa működtetéshez egy 12V-os áramforrást használtunk. Az áramforrás és a pumpa földeléseit egymással, a pozitív kábeleit pedig a reléhez csatlakoztattuk. Az áramforrás negatívja a relé általános csatlakozójába, a pumpa negatívja pedig a relé zárt csatlakozójába került. Ezzel elértük azt, hogy a pumpa alapesetben álljon és csak a szárazságot jelző jelre nyisson.



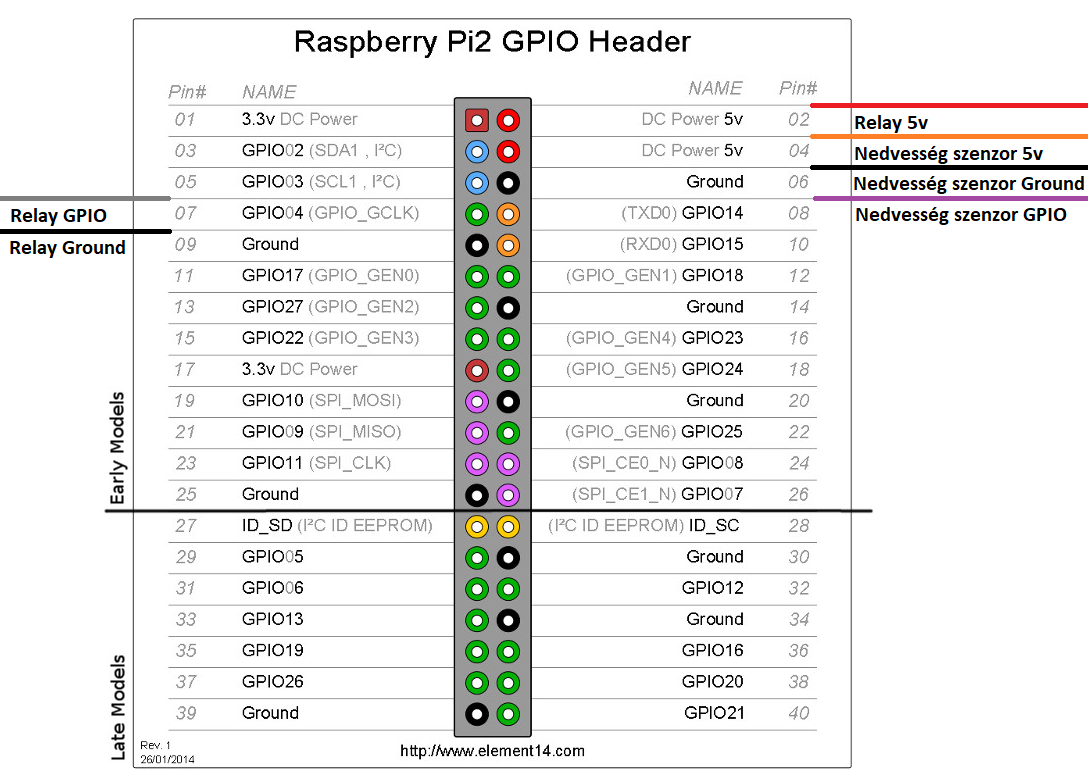
4. ábra: Relé kötési ábrája





5. ábra: Relé általi pumpa működtetés

Miután miden alkatrésszel rendelkeztünk és mindet leteszteltük el kellett döntenünk, miként fog alakulni a végleges PIN kiosztás. A nedvesség érzékelő szenzor a 8-as PIN-en keresztül kommunikált az eszközzel, a relé pedig a 7-esen keresztül. Emellett mindkettőt rákötöttük 1-1 földre és 1-1 3V-os PIN-re.

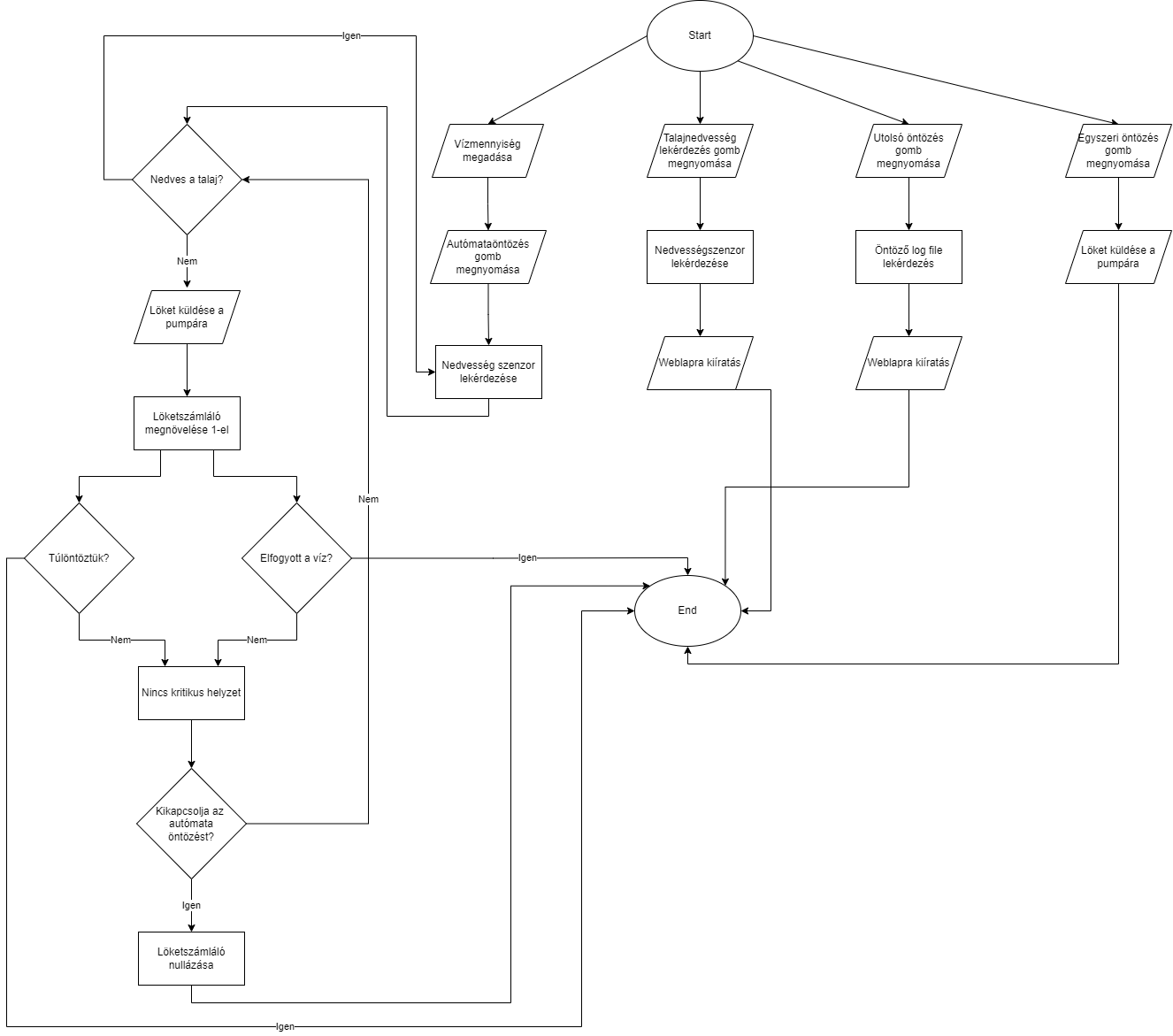


6. ábra: PIN kiosztás



7. ábra: felhasznált PIN-ek

**Folyamatábra**



8. ábra: Folyamatábra

**A kód**

Mindenek előtt a PIN-ek használata érdekében telepítésre került az RPI.GPIO könyvtár. Négy különböző részre bontottuk a megoldást:

* [water.py](https://github.com/patyaa/water/blob/master/water.py) – ebben a kódban került elhelyezésre az utolsó öntözés lekérdezése, a nedvesség ellenőrzése, a pumpa működtetése és az automata öntözés kifejtése
* [auto\_water.py](https://github.com/patyaa/water/blob/master/auto_water.py) – ezzel a résszel tudja a weboldal aktiválni az automata öntözést az előző kódból
* [web\_plants.py](https://github.com/patyaa/water/blob/master/web_plants.py) – a weboldal kódjának meghívásai itt történnek meg, mint folyamatok
* [main.html](https://github.com/patyaa/water/blob/master/templates/main.html) – a weboldal designjáért és megívásaiért felelős rész.

**water.py**

A lekérdező rész 2 funkcióra képes, amennyiben már volt öntözés és generálódott fájl ki tudja írni mikor volt az utolsó öntözés időpontja, amennyiben még nem volt öntözve egy üzenetet ad vissza, ami erre vonatkozik. A nedvesség lekérdezése a PIN-en keresztül egy 1-es vagy egy 0–ás jelet ad vissza az alapján, hogy éppen nedves vagy száraz a talaj. Az automata öntöző része a komplikáltabb folyamat. Itt van 2 biztonsági számítás elhelyezve. Ehhez először le kellett mérjük, hogy milyen mennyiségű vizet használ fel 1 löket. Ezzel az adattal egy egyenletet készítettünk, ami a szárazon futást akadályozza meg azzal, hogyha a felhasználó által megadott vízmennyiség 75% elfogyott, akkor leáll a rendszer, mivel az edény méretét nem tudjuk. Ez mellett egy túlöntözési tűréshatár is bekerült, ami pedig az egyszerre elfogyasztott mennyiséget nézi, tehát hogyha egyben elfogyaszt a pumpa 50% vizet, akkor megállunk, mivel ez túlöntözéshez vezethet. De amennyiben pl. 20%-nál nedveset jelez a szenzor és megáll, majd szárazság esetén újra indul, a rendszer nem a maradék százalékot veszi figyelembe, hanem újra figyelni kezdi az egyszerre öntözött víz mennyiségét. A löketek közé pedig egy 6 másodperces késleltetést állítottunk be, ez tűnt a legjobbnak, mivel kisebb értéknél nem volt ideje felszívódnia a víznek ezzel eláztattuk a föld felső részét és idő előtt nedvességet jelzett a szenzor. A pumpa löketére vonatkozó kód pedig még annyival több, hogy egy szöveg fájlba betölti az utolsó öntözés időpontját.

**auto\_water.py**

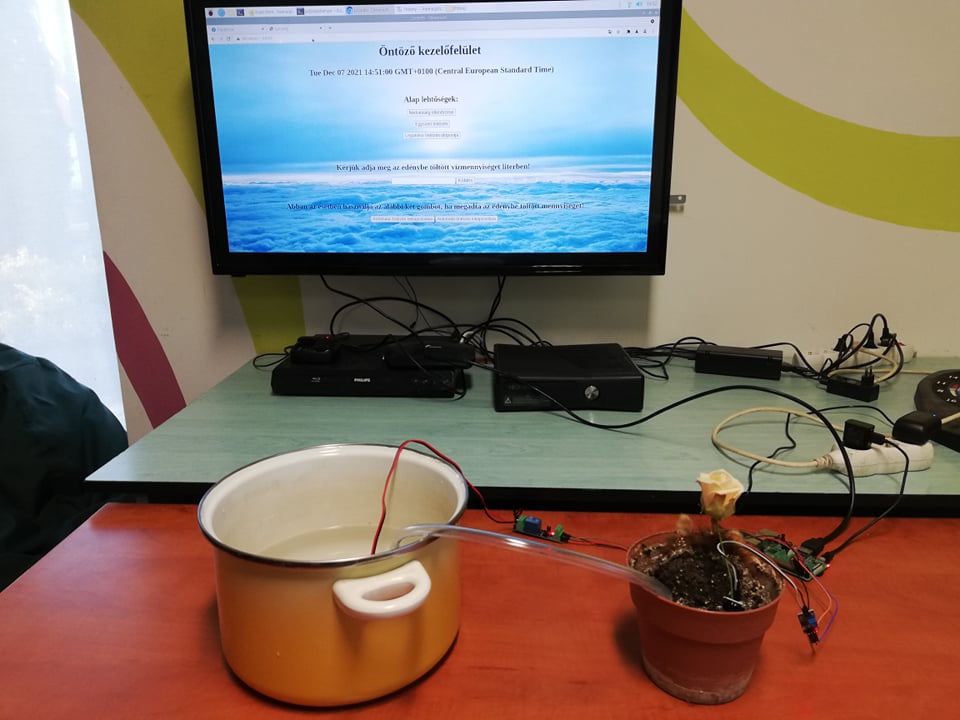
Importálja a *water.py*-t, valamint itt kerül feldolgozásra a felhasználó által megadott víz mennyiség.

**web\_plants.py és main.html**

A Python kód tartalmazza azokat a funkciókat, amit a webes felületen végre lehat hajtani. Elsőnek a nedvesség ellenőrzését hajtja végre és egy talaj száraz vagy talaj nedves üzenetet ír ki. Az egyszeri öntözés a pumpát kapcsolja egyszer és egy löket vizet a virágra juttat. A legutolsó öntözés a fájlba beírt időpontot adja vissza a felhasználónak. Amennyiben a felhasználó megadja a mennyiséget, amit az edénybe töltött, ezt is kiírja számára. Az automata kapcsolgatás pedig az automata öntözésre hivatkozik, és ez alapján dolgozik.



9. ábra: Weblap kinézete és a kiírás, jelen esetben a megadott mennyiségről



10. ábra: Az eszköz működés közben

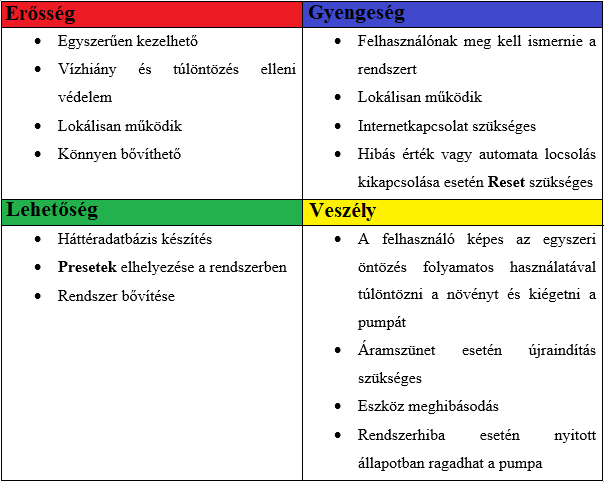
**Szükséglet**

Ahhoz, hogy a felhasználó az általunk elkészített rendszert tudja használni, kell, hogy biztos internt kapcsolattal rendelkezzen, amelyet csatlakoztat vagy **Wi-Fi**-n vagy kábelen keresztül az általunk elkészített kis számítógéphez. Amennyiben saját eszközről szeretné vezérelni a weblapot minden módosítás nélkül szükséges, hogy beszerezzen és összeépítsen mindent az alapján, ahogy mi is.

**Működés**

Az eszköz indítását követően a weblap is elindul és elérhetővé válik. A weblap állapotba áll és várja a felhasználó utasításait. A legutóbbi öntözés lekérdezése ebben a fázisban nem működik, mivel az indítás óta nem volt még öntözés, valamint az automata funkciók is működés képtelenek, mivel nem került megadásra mennyiség.

**SWOT**



11. ábra: SWOT analízis

**GPIO-hoz, web szerverhez és az automata induláshoz szükséges scriptek**

A fentebb is említett RPi.GPIO könyvtárhoz szükséges kód:

$> python3 –m pip install RPi.GPIO

Miután a szükséges könyvtárat beszereztük elkezdhető a kódok futtatása. Az alábbi parancsrészlettel futtatható a water.py:

$> python3

>>> import water

>>> water.get\_status ()

>>> water.pump\_on ()

Miután működik az öntözés a web szerver életre keltése a cél. Ahhoz, hogy minden jól működjön, szükséges, hogy a kódokat tartalmazó fájlok 1 mappában legyenek. Ezt követően szükség van két további könyvtárra:

$> python3 –m pip install flask

$> python3 –m pip install psutill

**Tesztelések, hibák kiküszöbölése**

Ahogy fentebb is említettük, tesztek folyamatosan zajlott az eszközök megérkezésétől kezdődően. A működéshez teszteltük úgy, hogy PIN-ként hivatkozunk és úgy is, hogy csatornánként. Ez utóbbi azért mellőztük, mivel alapesetben a PIN kiosztása az elterjedtebb és abban az esetben, ha valaki nincs tisztában azzal, mit használunk, és úgy próbál egy saját eszközt építeni sok megmagyarázhatatlan hibába tud ütközni.

Előszőr a relével is volt problémánk, mivel 5V-ossal rendelkeztünk, viszont a Raspberry csak 3V-ot volt képes adni, így nem hozta működésbe a pumpánkat egyszer sem.

A relénél merült fel a következő probléma is, ami több munkát igényelt. Esetünkben a pumpát a nyitott és a zárt részére tudtuk kötni. Logika azt diktálja, ha a pumpát a zártra kötjük, akkor addig nem indul el, amíg meg nem kapja az első impulzust. Ez fordítva működött, zárton alapból pumpálni kezdett és egy impulzus után leállt. Ezt követően már rendeltetésszerűen működött. Mai napig nem tudjuk ennek a pontos okát és mivel nem volt másik így tesztelni másikkal nem tudtuk, de jelenleg is működik szóval megoldódott a probléma.

A pumpa löketeivel is volt probléma, amit feljebb kifejtettem, hogy időzítés váltással korrigáltunk.

Először csak egy szárazon futás elleni védelem került bele, de rájöttünk, hogy nem csak a pumpa megóvása szükséges, mivel az egyik teszt során az edény nem ürült ki, de a virág és az asztal víz alá került, szóval ez miatt egy virág túlöntözés elleni védelem is bekerült.

**Összegzés**

Természetesen az eszközünk nem egy tökéletesen piacra dobható termék jelenleg, de vállalati gondolkodást követve, az eszközhasználat költséghatékony lett, inkább saját logikát építettünk bele az helyett, hogy minden külön szenzorokkal oldjunk meg. A csapat minden tagja szerint, egy körültekintő felhasználó kezei között biztonságban lenne az eszköz és a virágok is, ellenben egy majdnem tökéletes szoftver se nyújtana kellő védelmet egy hanyag, összeszedetlen felhasználóval szemben. Ha az eszközt forgalomba szeretnénk helyezni egy komplex tárolóra lenne még szükség és a kábelek hiánytalan burkolására, ettől eltekintve az eszköz több tesztet is abszolvált, amennyiben a kellő értelemmel kezelte a felhasználó.

**Források**

1. [Locsoló rendszer](https://medium.com/@albertolamana/water-plants-with-raspberry-pi-9a1127b57b99) 1
2. [Locsoló rendszer](https://shallowsky.com/blog/hardware/plant-watering-raspberry-pi.html) 2
3. [Locsoló rendszer 3](https://ozeki.hu/p_5958-plant-watering-using-soil-moisture-sensor.html)
4. [Locsoló rendszer 4](https://gist.github.com/rdeprey/6395b808c9b72213d8a3f298a63efaca)
5. [Szenzor teszt](https://www.piddlerintheroot.com/soil-moisture-sensor/) 1
6. [Szenzor teszt](https://www.techcoil.com/blog/how-to-read-soil-moisture-level-with-raspberry-pi-and-a-yl-69-fc-28-moisture-sensor/) 2
7. [Relé működtetése](https://kitflix.com/how-to-interface-relay-with-raspberrypi/)
8. [Pumpa leírása](https://grobotronics.com/mini-brushless-water-pump-12v-dc-240l-h-ad20p-1230a.html)
9. [Tryit editor](https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_input_type_number)
10. [HTML háttér elkészítése](https://www.javatpoint.com/oprweb/test.jsp?filename=how-to-add-background-image-in-html)
11. [HTML input bekérés](https://stackoverflow.com/questions/55456752/taking-user-input-from-html-form-as-a-variable-for-python-script)

**GitHub repository**

* [Saját GitHub repo](https://github.com/patyaa/water)