Szakdolgozat

Pintér László

Műszaki informatika, hálózati technológiák

Gépipari és Automatizálási Műszaki Főiskolai Kar

Kecskemét

2007

A Millikan-mérés számítógépes szimulációja

Tartalomjegyzék

1. Bevezetés

Számomra a fizika egy nagyon érdekes tudomány, sok lenyűgöző elmélet, és kísérlet látott már napvilágot a keretein belül. Mikor meghallom ezt a szót, azonnal eszembe jut az általános iskolai fizikatanárom kijelentése: *"Bárhova néztek, bármit csináltok, az mind fizika"* . Ezen akkoriban nem tűnődtem el, de mostani ésszel nagyon is értem mire gondolt, és hogy mennyire igaza van. Némely fizikai kísérlet roppant egyszerű, de vannak órákat, napokat felölelők és körülményesek is. Továbbá az előkészületek, és a költségek se elhanyagolandók. Ezen kísérletek közé tartozik Robert Andrews Millikan úgynevezett olajcsepp-kísérlete. Millikan Egyesült Államok-beli fizikus volt, aki a 19. és 20. század fordulóján élt, és tevékenykedett. Róla természetesen a későbbiekben még szót ejtek.

Arra gondoltam tehát, hogy informatikus, és kezdő programozó lévén, egy nagyszerű feladat lesz ezt a bonyolult kísérletet az érdeklődők számára számítógépre vinni, egy szimuláció formájában. Magához a program megírásához a Java programnyelvet választottam. Két oka is van a választásomnak, egyrészt jobban elsajátíthatom ezt a manapság nagyon népszerű nyelvet, másrészt pedig, ha Java applet formájában készítem el, akkor a futtatáshoz elegendő lesz egy megfelelő böngésző program is (például Mozilla Firefox, Internet Explorer, stb…). Továbbá van még egy harmadik előnye is a Java-nak, ami nem más, mint a programnyelv platformfüggetlensége, vagyis bármilyen számítógép konfiguráción megállja a helyét az elkészített program.

2. A Millikan-kísérlet fizikatörténeti jelentősége

2.1. Robert Andrews Millikan élete és munkássága

Robert Andrews Millikan 1868. március 22-én született az USA-ban az Illinois állambeli Morrisonban. Apja tiszteletes, nagyszülei bevándorlók voltak. Egy ideig bírósági jegyzőkönyvvezető volt. Később az Oberlin College-ban szerzett bakkalaureátus fokozatot 1891-ben klasszikus humán szakon. Fizikai doktorátusát 1895-ben szerezte a Columbia Egyetemen. Ugyanebben az évben Németországba utazott, ahol a Berlini és a Göttingeni Egyetemen töltött egy évet. Henry Gordon Gale szerzőtársával írtak egy nagyon sikeres fizika tankönyv sorozatot is. 1896-ban a Chicago-i Egyetemre kerül laboratóriumi asszisztensként, és fizikát is tanít. 1910-ben már az egyetem professzoraként publikálta olajcsepp-kísérletének első eredményeit. Egy kis idővel ezután Felix Ehrenhaft fizikus azt állította, hogy ő a hasonló kísérleteivel kisebb töltést mért. Erre Millikan egy újabb kísérletsorozatot végzett, majd 1913-as publikációjában megerősítette a korábban mért eredménye helyességét. 1916-tól a fotoeffektus kutatásával foglalkozott, kutatásaival a Planck-állandó numerikus értékének első közvetlen fotoelektromos meghatározását adta. 1917-től egy kaliforniai kutatóintézetben, a Throop College of Technology-ban volt kutató. Az intézmény később a California Institute of Technology - röviden Caltech - nevet vette fel, amely végrehajtó tanácsának Millikan lett az elnöke 1921-ben. Egészen 1945-ös nyugdíjazásáig ezen a poszton maradt. Az intézet főleg a kozmikus sugárzás kutatásával foglalkozott, és ért el fontos sikereket. 1923-ban fizikai Nobel-díjat kapott az "elektron elemi töltésének meghatározásáért, és a fényelektromos hatásra vonatkozó kutatásaiért". 1927. környékén Freidrich Hunddal közösen megalkotta a kvantumviselkedéssel kapcsolatos Millikan-Hund-elméletet. Millikan nős ember volt, és három fia született. Sokszor említik a teniszjáték iránti szenvedélyét is. 1953. december 19-én halt meg Kaliforniában, San Marinó-i otthonában. Glendale-ben helyezték örök nyugalomra.

2.2. A kísérlet jelentősége

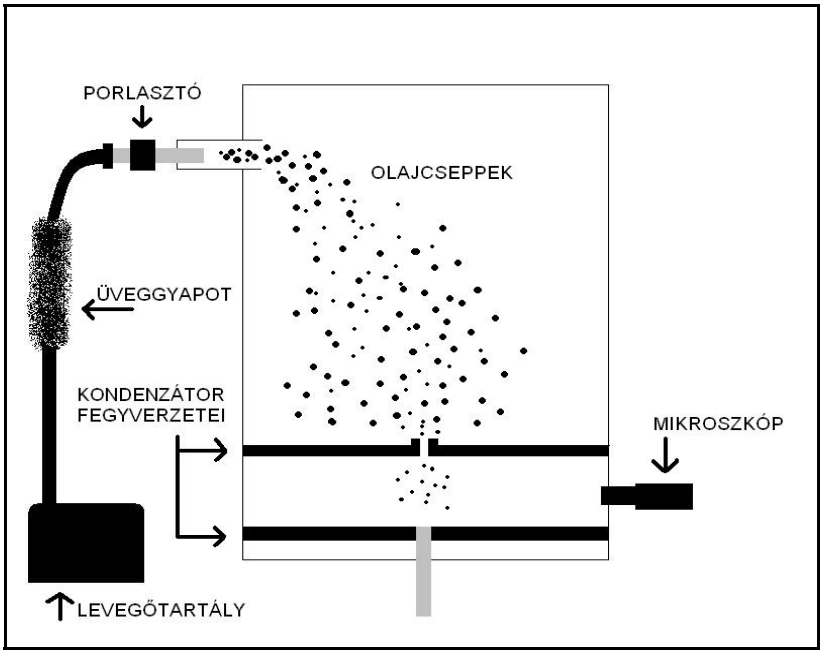
Millikan előtt már sokan próbálkoztak az elektron elemi töltésének meghatározásával, bár még maga az "elektron" szó is csak 1891-ben vált használatossá. Már az ókori görögök is feltételeztek valamilyen "láthatatlan erőt" mikor azt tapasztalták, hogy a megdörzsölt borostyánkő a könnyebb tárgyakat magához vonzza. Ez volt az emberiség első találkozása az elektromossággal. A 18. században ezt Benjamin Franklin azzal magyarázta, hogy minden anyagban bizonyos mennyiségű elektromos folyadék található, melynek többlete pozitív, hiánya pedig negatív töltést eredményez. Ő már feltételezte azt is, hogy létezik olyan elektromos részecske, amely elég kicsi ahhoz, hogy behatoljon az anyagba. Franklin eme két elmélete a 19. század végéig az elektromos jelenségek magyarázata maradt. H.S. Wilson Townsend volt az első a 19. század végén aki megkísérelte megmérni egyetlen ion töltését. Eredményei 10^{-19} C nagyságrendbe estek. Pár évvel később, 1899-ben, Joseph John Thomson, angol Nobel-díjjas fizikus, hasonló kísérletével $2x10^{-19}$C értéket kapott. 1909-ben Townsend és Thomson kísérleti elképzeléseit Millikan alapjaiban javította és fejlesztette tovább, így jutva a későbbi helyes eredményre.

Ez a méréssorozat volt az olajcsepp-kísérlet, amivel rájött arra, hogy van egy elemi töltésadag, aminél kisebb töltés nem létezik, és minden töltés ennek az adagnak a többszöröse. Ezt követően katódsugárzásos módszerrel kimutatta, hogy az elektron töltésének a nagysága nagyjából megegyezik az elemi töltéssel. Majd ezekkel az ismeretekkel visszatérve az olajcsepp-kísérletéhez, pontosította és továbbfejlesztette azt. Millikan végeredménye $-1{,}592x10^{-19}$C lett az elemi töltés értékére, amelyben csekély az eltérés a mai modern értékhez ($-1{,}602x10^{-19}$C) képest. Az olajcsepp-kísérlet volt tehát az első közvetlen és mindent eldöntő mérés egyetlen elektron elektromos töltésének mérésére. Továbbá ezen kísérletek révén vált lehetségessé az elektron tömegének a meghatározása is, a korábban már kimért fajlagos töltés felhasználásával. Ezután kijelenthetjük, hogy ez a több évet felölelő méréssorozat bizonyult a fizikatörténet egyik legjelentősebb eredményének, méltán értékelték Millikan teljesítményét Nobel-díjjal.

3. A Millikan-mérés részletes leírása

3.1. A berendezés

Millikan első olajcsepp-kísérletében egy igen bonyolult technikai berendezést használt. Mivel rendkívül pontos megfigyelést igényel a mérés, ezért nehézkes volt a megvalósítás, főleg az akkori technikákkal. Feljegyzései szerint a 60 napos méréssorozat alatt mindössze 58 csepp töltését sikerült meghatároznia. A berendezés, amivel dolgozott egy nagy olajtermoszban volt, annak érdekében, hogy a hőmérséklet állandó legyen a kísérlet alatt. Különleges vákuumolajat használt, hogy csökkentse a párolgást. A termoszban voltak elhelyezve egy kondenzátor fegyverzetei. A termoszba nagynyomású tartályból érkezett a levegő, amit üveggyapoton keresztül juttatott egy porlasztóba, párátlanítva azt. A porlasztóban a levegő keveredett az olajjal, és miután az olajcseppek bejutottak a termoszba, egy kis zárható nyíláson tudtak beesni a kondenzátor fegyverzetei közé. Itt történt a megfigyelés is mikroszkóppal egy lyukon keresztül, ami a termosz oldalfalában volt. Később a katódsugárzásos kísérlete után, Millikan megoldotta azt is, hogy a cseppek töltése a vizsgálat alatt is változtatható legyen. Ehhez röntgensugárral ionizálta a fegyverzetek közti légteret, és az ebbe belehulló cseppek a levegőtől töltéseket vettek fel. A berendezés vázlatrajza az 1. ábrán látható.



1. ábra. A berendezés vázlatrajza

3.2. A mérési folyamat és a mérés kiértékelése

A kísérletet az olajcseppek befecskendezésével kezdjük. Ahogy a cseppek esnek, néhány közülük átesik egy nyíláson a kondenzátor fegyverzetei közé. Ha a kondenzátor nincs töltve, nincs elektromos mező, és a cseppek esnek tovább az eredeti sebességükkel. A cseppekre ilyenkor három erő hat, a nehézségi erő (F_g) lefelé, a közegellenállási erő (F_s) és a felhajtóerő (F_f) felfelé. Mivel a cseppek tömege nagyon kicsi, ezért hamar elérik a végsebességüket, és csaknem egyenletesen mozognak lefelé. A mozgás azért nem lesz teljesen egyenletes, mert a kis tömeg miatt a cseppek végeznek némi Brown-mozgást is. A mikroszkópon át, amelynek lencséjén egy fokokra osztott skála található, megmérhetjük a kiválasztott csepp esési sebességét ($v_{esési}$) úgy, hogy megfigyeljük a csepp mennyi idő alatt jut egyik beosztástól a másikig. Ezután a kondenzátorra feszültséget (U) adunk, és a cseppek elkezdenek emelkedni az elektromos tér hatására. A feszültséget pedig úgy állítjuk be, hogy a kiszemelt csepp egyenletesen emelkedjen. Ezután az emelkedési sebességét ($v_{emelkedési}$) szintén megmérhetjük. Miután megvannak ezek az adatok, már csak a következő egyenleteket kell alkalmaznunk az adott csepp töltésének (q) meghatározásához:

Tudnunk kell a csepp sugarát (r), amit az

$$r = \sqrt{\frac{9\eta\, v_{esési}}{2\rho g}} \tag{1}$$

képletből kiszámíthatunk, ahol η a levegő viszkozitása ($\eta = 1{,}81 \times 10^{-5}\ \frac{Ns}{m^2}$), ρ az olaj sűrűsége ($\rho = 886\ \frac{kg}{m^3}$) és g a nehézségi gyorsulás ($g = 9{,}806\ \frac{m}{s^2}$).

Ezután meghatározzuk a csepp tömegét (m) az

$$m = \frac{4}{3}\pi \rho r^3 \tag{2}$$

egyenletből. Ezek után már csak a csepp töltésének meghatározása van hátra a

$$q = \frac{mg\left(1 + \frac{v_{emelkedési}}{v_{esési}}\right)}{E} \tag{3}$$

képlet alapján.

Így tehát megkapjuk egyetlen cseppecskének a töltését. Ahhoz viszont hogy kiszámoljuk az elemi töltést, több ilyen cseppecskének a töltésére van szükségünk, majd az adatok alapján a közös többszöröst keresve meglelhetjük az elemi töltés értékét.

4. A Millikan-mérés, mint hallgatói labormérés

A természettudományi és műszaki felsőoktatásban a Millikan-mérés igen széleskörűen megjelenik a hallgató labormérésekben, kihangsúlyozva annak mind elvi jelentőségét, mind gyakorlati vonatkozásait. Például a Kecskeméti Főiskola GAMF Kara is rendelkezik Millikan mérőberendezéssel (PASCO AP-8210), de a hallgatói labormérések közt nem szerepel, éppen az alábbiakban megfogalmazott problémák miatt.

A Millikan-mérés kivitelezése ugyanis sok potenciális hibaforrással terhelt és roppant időigényes. A használt olaj minősége és állapota, a porlasztó tisztasága és beállítása, az ionizáló forrás stabilitása, a levegő páratartalma, a berendezés rezgései, a leolvasási pontatlanságok külön-külön alig észrevehető, de együttes hatásukban már jelentős hibaforrást jelentenek. Ezért a valós kísérletben igen nagy számú mérést kell végezni (sok cseppecskével és egy cseppen is lehetőleg többször megismételve), a statisztikus hibák csökkentése érdekében. Sajnos még így is maradhatnak szisztematikus hibák, amelyek a teljes mérési sorozatot negligálhatják. Tapasztalatok szerint egy elfogadható eredményt produkáló mérés-sorozat átlagosan 6-10 munkaórát igényel. Ennyi időt a legtöbb esetben nem szánhatunk egy hallgatói mérésre.

Ugyanakkor a kísérlet óriási tudománytörténeti jelentősége - minden idők legszebb és legfontosabb fizikai kísérleteinek listáján, melyet amerikai fizikusok véleményei alapján készítettek, bronzérmes helyet foglal el (lásd pl. http://www.inczedy.hu/~szikszai/nuke1/fizika/kis/index.htm) - miatt fontos lenne, hogy a hallgatók mégis találkozzanak, és érdemben foglalkozzanak vele. A számítógépes szimuláció tartalmazza a mérés minden lényegi mozzanatát, de "steril" mérési körülményeket biztosít, így a mérés időtartama jelentősen lerövidül, bonyolultsága eltűnik.

5. A Millikan-mérés Java appletes megvalósítása

5.1. A fejlesztői környezet

A programot Java applet formában fogom elkészíteni. Ennek legfőbb oka az, hogy bárki egyszerűen használhatja majd otthoni számítógépén, mindössze internet kapcsolatra, és egy Java-t futtatni képes böngészőre van szüksége. Maga a programizási nyelv pedig lehetővé teszi, hogy bármilyen számítógép konfigurációval és operációs rendszerrel megjeleníthető legyen az alkalmazás. Az appletek olyan mini alkalmazások, amelyek a világhálón keresztül egy weblapba ágyazódva töltődnek le, amit azután a böngészővel futtathat a felhasználó. Az applet futtatását a Java interpreter végzi. Ezt használják megjelenítésre a Java-t támogató böngészők is. Az interpreter ellenőrzi a programot, mielőtt lefuttatná a felhasználó gépén. Ennek az az oka, hogy általában nem tudjuk, hogy ki készítette az alkalmazást, és az illetőnek lehettek ártó szándékai is. Erre az esetre szigorú előírások vonatkoznak, amelyek tartalmazzák hogy a Java program milyen hozzáférési engedélyekkel rendelkezhet a felhasználó számítógépén. Ezeknek az előírásoknak a betartatását is az interpreter végzi. Egy Java applet forráskód létrehozása után szükséges egy hozzá tartozó HTML állomány létrehozása is, amelybe beágyazhatjuk a programunkat. Erre azért van szükség, mert a böngészők csak így képesek futtatni az appleteket. Ezen felül követelmény a Java-ban, hogy minden appletnek kötelezően ki kell terjesztenie az Applet osztályt, és a forrást tartalmazó fájl nevének meg kell egyeznie a benne leírt osztály nevével. Jelen esetben Millikan.java a forrásállomány neve, és a benne leírt osztály pedig a Millikan.class.

Az alkalmazást JCreator LE 4.00.026 verziószámú, ingyenesen használható fejlesztői programmal készítem el. Ezen felül J2SE Development Kit (JDK) 5.0 Update 8 és J2SE Runtime Environment (JRE) 5.0 Update 10 van a számítógépemre telepítve. A fejlesztés alatt és után a teszteléseket Mozilla Firefox 2.0 verziószámú böngészővel fogom végezni.

5.2. A program specifikációja

Egy olyan programot szeretnék írni, amely bárki számára egyszerűen kezelhető, nem túlbonyolított. Az egyszerűségre törekvés, viszont ne menjen a pontosság rovására, továbbá ne fedjen el fontos információkat a felhasználó elől. Ez az utóbbi azért fontos, mert egy szimulációs programról lévén szó, a lehető legtöbb adatot kell szolgáltatni a futás során. A program feladata lesz különböző méretű pontokat megjeleníteni és mozgatni, amelyek az olajcseppecskéket szimbolizálják. A cseppecskéknek viszont nem egyforma sebességgel kell majd esniük, és az animáció futása közben a programnak adatokat kell majd szolgáltatni a felhasználó felé. A program tartalmazni fog grafikus elemeket, amik animálva lesznek a látványosság érdekében. Ez a grafikus rész a kondenzátor fegyverzetei közötti tér egy részletét szimbolizálja. Továbbá kell egy felület, amin keresztül a program informálja a felhasználót az aktuális történésekről. A programmal való kommunikáció érdekében gombokat kell alkalmazni, hogy a felhasználó irányíthassa a szimulációt. Tehát pontosítva az előző gondolatokat a Millikan-féle olajcsepp-kísérlet ismeretében, a következőkre lesz szükség:

- egy terület amelyen az animáció fut valós időben,
- kijelzők, és információ-kiírására szolgáló mezők,
- gombok amikkel utasítható a program.

Ezt a három csoportot három osztályban fogom megvalósítani. Az érdemi számításokat, és a kalkulációkat is az animációt megjelenítő osztály fogja végezni. A felhasználó tájékoztatására egy üzenőfalnak megfelelő rész lesz a grafikus felület mellett. A gombok pedig a grafikus és az üzenőrész alatti sávon fognak elhelyezkedni, amit a harmadik osztály fog kezelni. A program végcélja pedig az lesz, hogy különböző csepp-töltés értékeket, és esetleges rész-információkat szolgáltasson a felhasználó számára, amelyből aztán kiszámíthatja az elemi töltés értékét.

5.3. A program működési elve

A program más elven fog működni, mint a valódi kísérlet. A valódi kísérletben sok olyan tényezőt figyelembe kell venni, amit a programban elhanyagolhatunk. Ilyenek például a használt olaj minősége és állapota, a porlasztó tisztasága és beállítása, az ionizáló forrás stabilitása, a levegő páratartalma, a berendezés rezgései, a leolvasási pontatlanságok. Ezen okok miatt például egy cseppecskén több mérést is el kell végezni a minél pontosabb eredmény érdekében. A programból ezt a lehetőséget kihagyom. Vagyis hiába mérnénk meg egy cseppecske paramétereit újra és újra, ugyanazt az eredményt kapnánk, a "steril" körülmények miatt.

A program működésének vázlata a felhasznált képletek alapján tehát a következő: betöltődés után egy gombbal cseppecskéket "injektálunk" a rajzolófelületre. Ekkor lefut több véletlen szám generáló függvény a háttérben. Először a megjelenítendő cseppek számát generálja le, ami 50 és 120 közötti cseppszámot eredményez. A

$$q = N * e \tag{4}$$

összefüggés alapján megkapjuk a töltésértékeket. A N változó véletlenszerű értékeket kap minden egyes csepphez 0 és 50 között, majd a q töltésértékek egy tömbben indexhelyesen eltárolódnak. A e változó konstans, az elemi töltés értékét szimbolizálja. A cseppecskék sugarát (r) is ekkor generálja le a program 0,1 és 1 μm közötti tartományban, amit méterben tárol indexhelyesen egy tömbben. Továbbá ezekből a cseppsugár-értékekből kiszámítja a megjelenítéshez szükséges arányos méretet, mivel ilyen pici értékeket nem lehet megjeleníteni, és szabad szemmel sem lennének láthatóak. Ezeket a méreteket is egy tömb tárolja indexhelyesen. Végül minden csepphez generálódik véletlenszerűen, adott határok között, egy x és egy y koordináta, amely a megjelenésük helyét adja a képernyőn. Ezek is eltárolódnak egy-egy tömbben indexhelyesen.

Ezután még nem kapcsoljuk be az elektromos teret, így a cseppecskékre három erő hat, a gravitációs erő, a felhajtóerő és a közegellenállási erő. Ezek közül a felhajtóerőt a

$$F_f = 6\pi r\eta v_1 \tag{5}$$

képlet adja, a közegellenállási erőt pedig a következő összefüggés:

$$F_k = \frac{4\pi}{3} r^3 (\rho_0 - \rho_e). \tag{6}$$

Az (5) és (6) egyenleteket kombinálva kapjuk a

$$6\pi r\eta v_1 - \frac{4\pi}{3} r^3 (\rho_0 - \rho_e) = 0 \tag{7}$$

egyenletet, amiből következik, hogy

$$v_1 = \frac{r^2}{C_1}. \tag{8}$$

Ebből a képletből a program kiszámolja minden egyes csepp esési sebességét. Láthatjuk, hogy az esési sebesség a cseppek sugarától függ. Az eredményeket indexhelyesen eltárolja egy tömbben, és az így tárolt értékek alapján végzi a cseppek esésének szimulálását is. A képletben szereplő C_1 egy konstans, ami tartalmazza a harmadik cseppre ható erőt, a gravitációs erőt. Értéke a következőképpen alakul ki:

$$C_1 = \frac{9\eta}{2g(\rho_0 - \rho_e)}, \tag{9}$$

ahol ρ_0 az olaj sűrűsége, ρ_e pedig a levegő sűrűsége ($\rho_e = 1{,}3\frac{kg}{m^3}$). Ezt a konstanst a program induláskor kiszámolja, és a futás végéig változatlanul tárolja.

Ezután a cseppek elkezdenek esni a képernyőn, mindegyik a maga sebességével. Ezek a megjelenített esési sebességek természetesen nem a valóságosak, hanem arányosan növeltek a valós értékhez képest. Képzeljünk el például egy 0,1 $\frac{mm}{s}$-al haladó pontot a képernyőn. Egyértelmű, hogy a program használhatatlan lenne ilyen működési paraméterekkel. A lefelé tartó cseppek közül ezután kiválaszthatunk egyet, amin elkezdődik a mérés. Amint átért a kiválasztott cseppecske a mérési területen, a program kiírja az üzenőterületre az esési sebességét. Ezután bekapcsolhatjuk a feszültséget, és addig növeljük, amíg megfelelő nagyságú nem lesz az elektromos tér, és el nem indul fölfelé a csepp. Az üzenőterületen folyamatosan követhetjük a csepp mozgási irányát, ha eltűnne a látómezőből, és a

feszültség növelésével, vagy csökkentésével változtathatunk azon. A feszültség bekapcsolásakor már más erő is hat a cseppre, a Coulomb erő, a mi példánkban:

$$F_C = N * e\frac{U}{d} \tag{10}$$

Egy új konstanst is létrehozhatunk a további számítások egyszerűsítéséhez:

$$C_2 = \frac{e}{6\pi\eta d} \tag{11}$$

A konstanst a program az induláskor szintén létrehozza, és azt a továbbiakban változatlanul tárolja. Az emelkedési sebességünk egyenlete a (8), (10) és (11) képletek kombinálása után így alakul:

$$v_2 = C_2 U\frac{N}{r} - v_1 \tag{12}$$

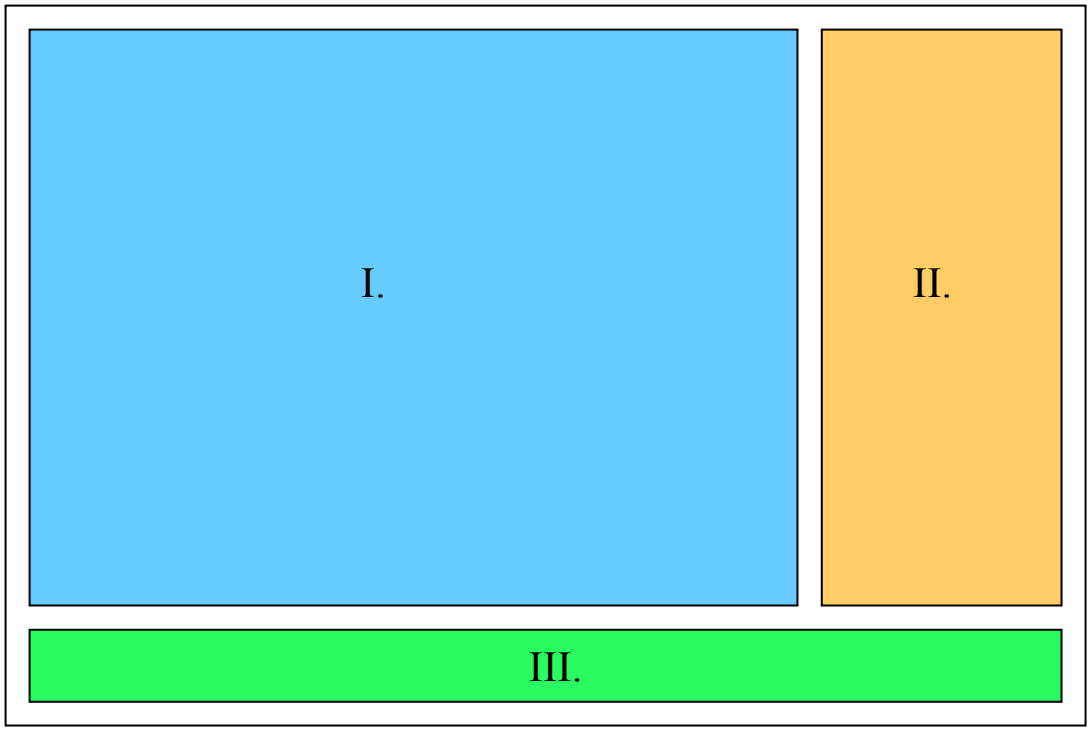
Mivel itt már csak egy cseppünk van, ezért nincs szükség tömbre az érték tárolásához. Amennyiben a cseppre nem rakódott töltés, vagyis a N értéke nulla, a v_2 sebesség soha nem lesz pozitív, és így nem is fog elhaladni a mérési területen. Ekkor a program mérés nélkül kiírja az emelkedési sebességére hogy "negatív". Ha pozitívvá válik a csepp emelkedési sebessége, bizonyos feszültségértékre, akkor elindul felfelé. Ha áthaladt a mérési területen a program automatikusan kiírja az emelkedési sebességet, a feszültségértékkel együtt, továbbá természetesen a cseppecske töltését is. Ezzel be is fejeződött egy mérési ciklus, az indító gomb megnyomásával kezdhetjük a következő mérést. A program tartalmaz egy számlálót is, ami méri az esési és emelkedési időt milliszekundumban, amíg a csepp a mérési területen tartózkodik. Ezt az időértéket minden alkalommal automatikusan ki is jelzi.

5.4. A felhasználói felület

A felhasználói felület tervezésekor a következő szempontok alapján jártam el:

- Egyszerűség, hogy a számítástechnikában kevésbé jártas emberek is használhassák a programot.
- Egyértelmű kezelőfelület, hogy ne lehessen félreértelmezni a feliratokat és az üzeneteket amit a program küld.
- Jól elhatárolható elemek, logikus elhelyezéssel.
- A lehető legtöbb információ közlése futás előtt és közben, de mégse történjen "információ-túlterhelés".

Ezek alapján a következő formára jutottam:



2. ábra. A felhasználói terület vázlatrajza

2. ábra értelmezése:

A I-el jelölt terület lesz az a terület, ahol az animáció megjelenik, a cseppecskék mozgását követhetjük itt nyomon. A továbbiakban rajzterületként említem.
A II-el jelölt területen lesznek láthatók információk, amiket a program futás közben közöl, és ide írja az eredményeket is. Ezután üzenőterületként nevezem meg.
A III-al jelölt területen helyezkednek majd el a gombok, a feszültségkijelző, és a számláló. A terület elnevezése irányítóterület lesz.
Az előző gondolatmenetet részletezve, és a 2. ábrát tovább gondolva eljutottam a felhasználói felület végleges formájához, amit a 3. ábrán már Java-ban elkészítve láthatunk:

3. ábra. A felhasználói felület Java-ban elkészítve

A 3. ábra részletezése, a felhasználói felület elemei:

I. A rajzterület a kondenzátor lapjai közötti rész középső részletét szimbolizálja. Ezen belül a zölddel színezett rész a mérési terület, ahol a mérések történnek. A cseppek injektálás után a felső világoskék területen jelennek meg, és elkezdenek esni különböző sebességekkel. A cseppeknek különböző a nagyságuk, attól függően, hogy mekkora a sugaruk. A cseppek töltése viszont nem függ a nagyságuktól, tehát a feszültség bekapcsolásakor nem tudhatjuk, hogy viselkedik majd a kiválasztott cseppecskénk. A program a cseppkiválasztást csak a felső világoskék területen engedélyezi a

mérés helyessége érdekében. Ha nem ott történik meg a cseppkiválasztás a cseppek automatikusan újrainjektálódnak.

II. Az üzenőterületre írja ki a program az információkat és a mérési eredményeket. Kezdéskor, ahogy betöltődik az applet, rögtön egy üdvözlőszöveggel fogadja a felhasználót, és tájékoztatja hogy mit is csinál a program. Ha a felhasználó elindítja a programot, és még nem választott cseppet, akkor a "Válassz cseppet" felirat látható az üzenőfalon. Ezt a felhasználó a méréssorozat alatt többször nem láthatja, mivel nyilvánvalóan a továbbiakban már tudni fogja tájékoztatás nélkül is ezt a feladatát. Miután választott cseppet, az üzenőfal első sorában átveszi a helyet egy iránymutató szolgáltatás, amely a későbbiekben folyamatosan jelzi a cseppecske haladási irányát. Tehát amint a kiválasztott cseppecske átért a mérőterületen, és bekapcsoljuk a feszültséget, automatikusan kiíródik az esési sebessége, majd elindul felfelé. Ha nem indulna el felfelé, azon okból kifolyólag, hogy nincs töltése, akkor az üzenőterületen az emelkedési sebességre a "negatív" feliratot kapjuk, és kiíródik a töltésére hogy "0 C". Viszont ha bizonyos feszültségértékre már elindul a csepp felfelé, és áthaladt a mérési területen, akkor szintén automatikusan kiíródik az emelkedési sebessége, az aktuális feszültségérték és a töltése. Ezzel el is készült az első mérés, kezdődhet a következő, amelynek az eredményei hozzáfűződnek az előző végéhez, természetesen jól elkülönítve.

III. Az irányítóterületen négy gomb (Injektálás, Feszültség +, Feszültség -, Info), 2 kijelző (feszültségkijelző, időtartam-kijelző) és a hozzájuk tartozó címkék ("Feszültség:", "Számláló:") helyezkednek el. Indításkor a program két lehetőséget kínál fel, amit kiír az üzenőterületen. Megnyomhatjuk rögtön az "Injektálás" gombot, amivel elindul a mérés, vagy ha még nem vagyunk tájékozottak a program működését illetően, akkor választhatjuk az "Info" gombot. Ennek a gombnak a megnyomásával a felhasználó a program kezelésével kapcsolatos információkhoz jut, továbbá megtudhatja a készítő e-mail címét, és nevét. Ha az "Injektálás" gombot választotta, elindul a mérés, és ha a kiválasztott cseppecskénk beér a mérési területre, elindul a számláló. A számláló automatikusan indul, és automatikusan megáll mikor

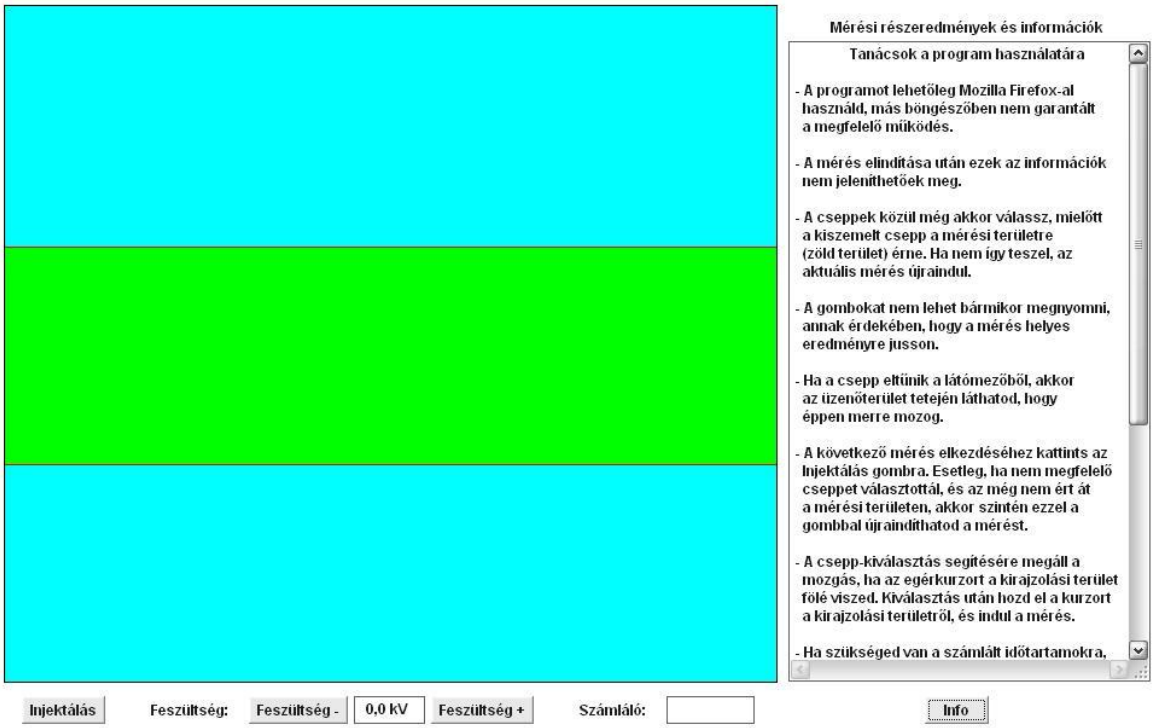
a csepp kiér a mérési területről. Akár lefelé, akár felelé halad a csepp, amikor a mérési területen tartózkodik a számláló mér. Az "Injektálás" gomb szolgál továbbá a következő mérés elindítására, vagy ha szükséges, az aktuális mérés újraindítására is. Ha a csepp kiválasztás után átért a mérési területen, bekapcsolhatjuk a feszültséget a "Feszültség +" gombbal. Ezután már szabadon használhatjuk a "Feszültség -" gombot is addig, amíg nulla nem lesz a feszültség, mert a negatív feszültség érték nem engedélyezett a programban. Továbbá az utóbbi két gomb tiltva van akkor is, amikor a cseppünk a mérési területen halad. A két gomb között található a feszültségkijelző, ami két tizedesjegyig és kV-ban jelzi a feszültséget. A gombokkal egy kattintással 0,05 kV-tal lehet növelni vagy csökkenteni a feszültséget.

Működés közben a felhasználói felület állapotai:

1. Betöltődési nézet, indítás előtt:

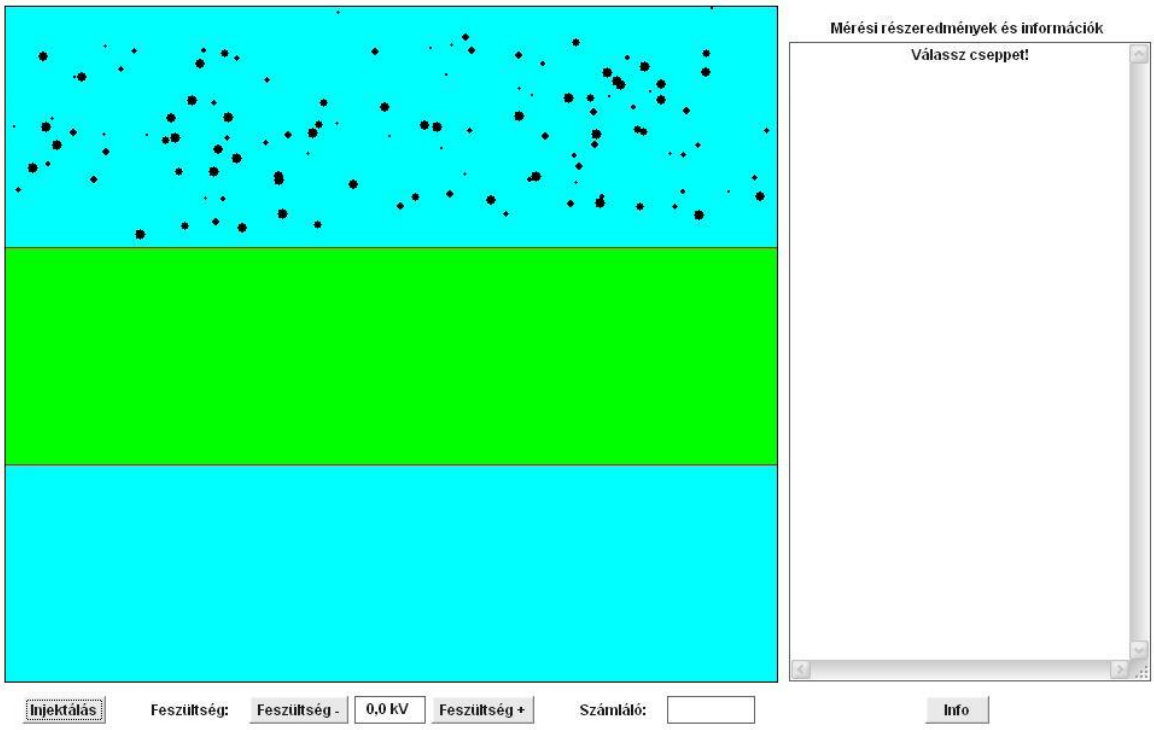
4. ábra. Betöltődési nézet

2. Indítás előtt, az "Info" gomb lenyomását követően:



5. ábra. Az "Info" gomb lenyomása utáni nézet

3. Az "Injektálás" gomb lenyomása utáni nézet, a program elindult:

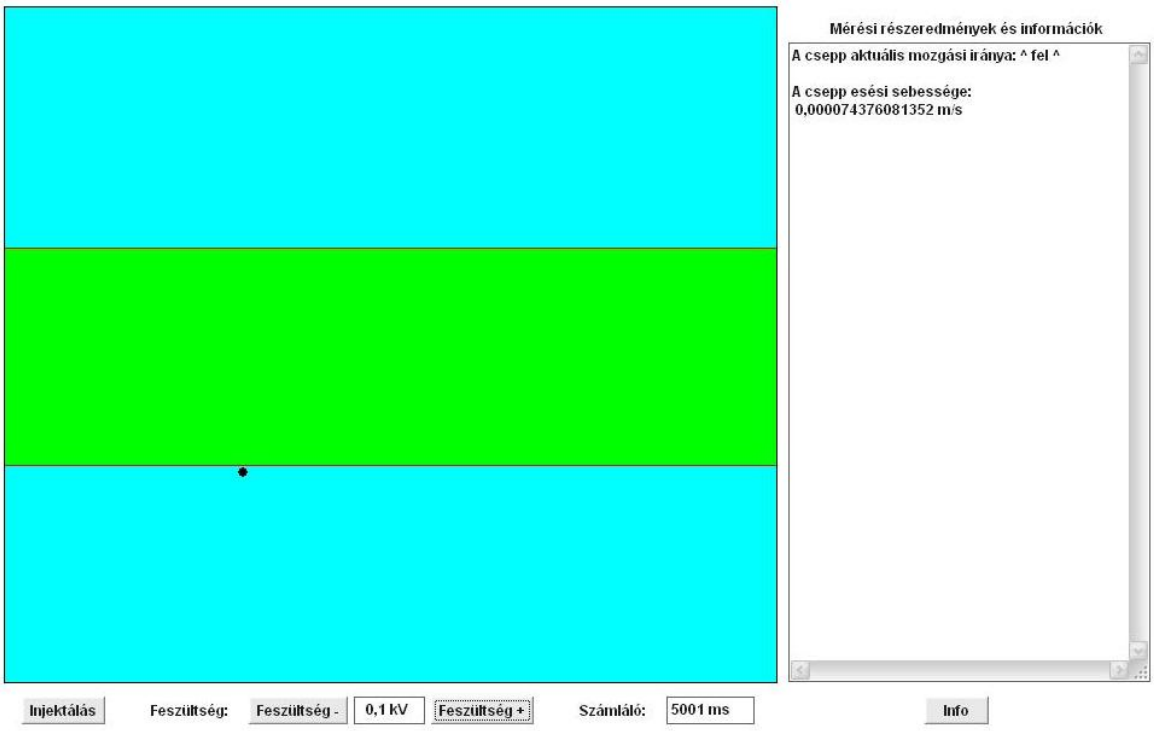


6. ábra. Az "Injektálás" gomb lenyomása utáni nézet

4. Csepp-kiválasztás utáni nézet:

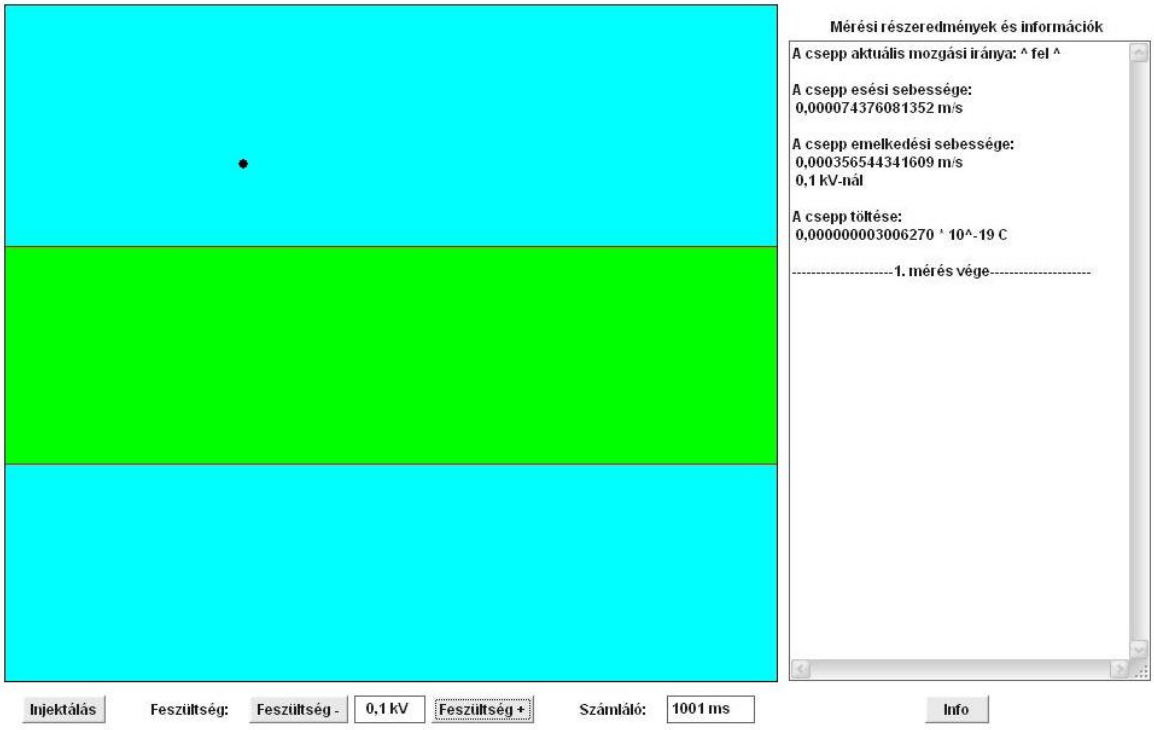
7. ábra. Csepp-kiválasztás utáni nézet

5. A csepp esési sebességének kiírása, miután átért a mérési területen:



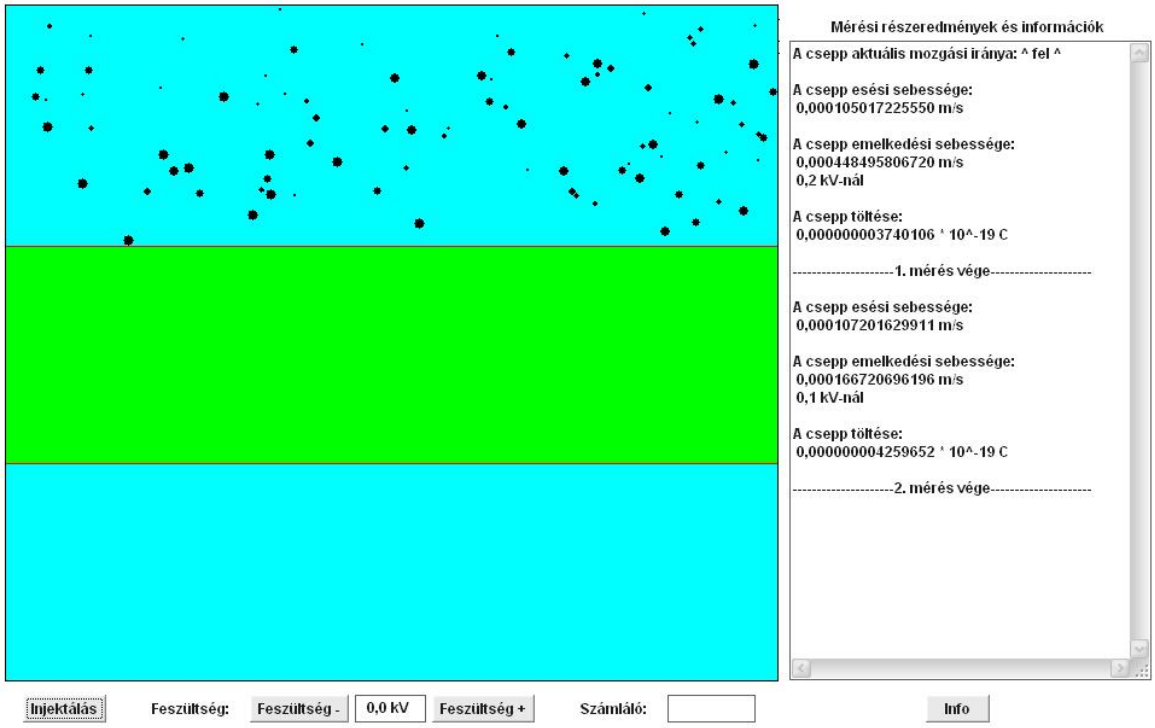
8. ábra. A csepp esési sebességének kiíratása

6. A csepp további paramétereinek kiíratása:



9. ábra. Az első mérés vége utáni nézet

7. Több mérés utáni nézet:



10. ábra. Több mérés utáni nézet

5.5. Osztályok és metódusok

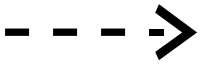
A szakdolgozatom ezen fejezetében részletesen ismertetném, hogy a programban helyet kapott osztályok, metódusok és változók milyen szerepet töltenek be, miért is szükségesek, és hogy milyen feladatot végeznek. Ezután röviden leírnám a programozási tchnikát is. Az osztályok és metódusok leírása a forráskód elejéről a végére tart, a begépelés sorrendjében, nem pedig a program futását végigkövetve. Előbb azonban, hogy összefogóbb képet kapjunk a program működéséről és struktúrájáról, két diagramon szemléltetném ezeket.
A program működésének követésére a használati eset diagram:

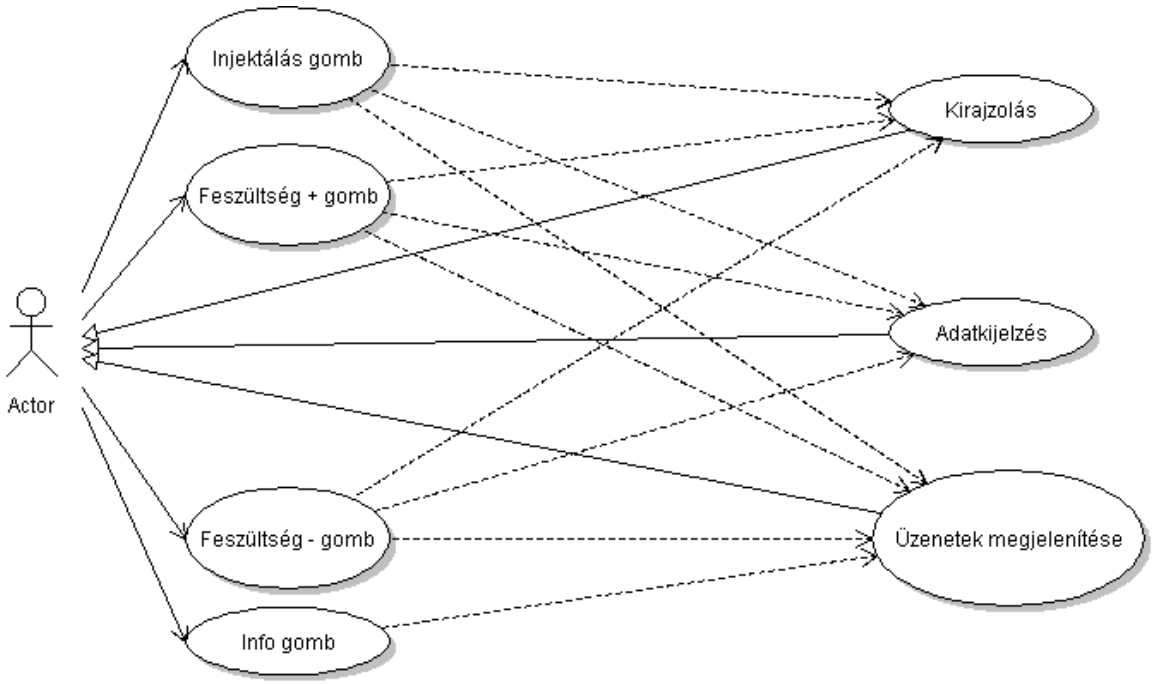
Jelölések:

- Actor, más néven a felhasználó, aki utasítja a programok
- A felhasználó beavatkozását jelölő nyíl:

- A program felhasználó felé továbbított jelzéseit szimbolizáló nyíl:

- A gombok programra való hatását jelölő nyíl:

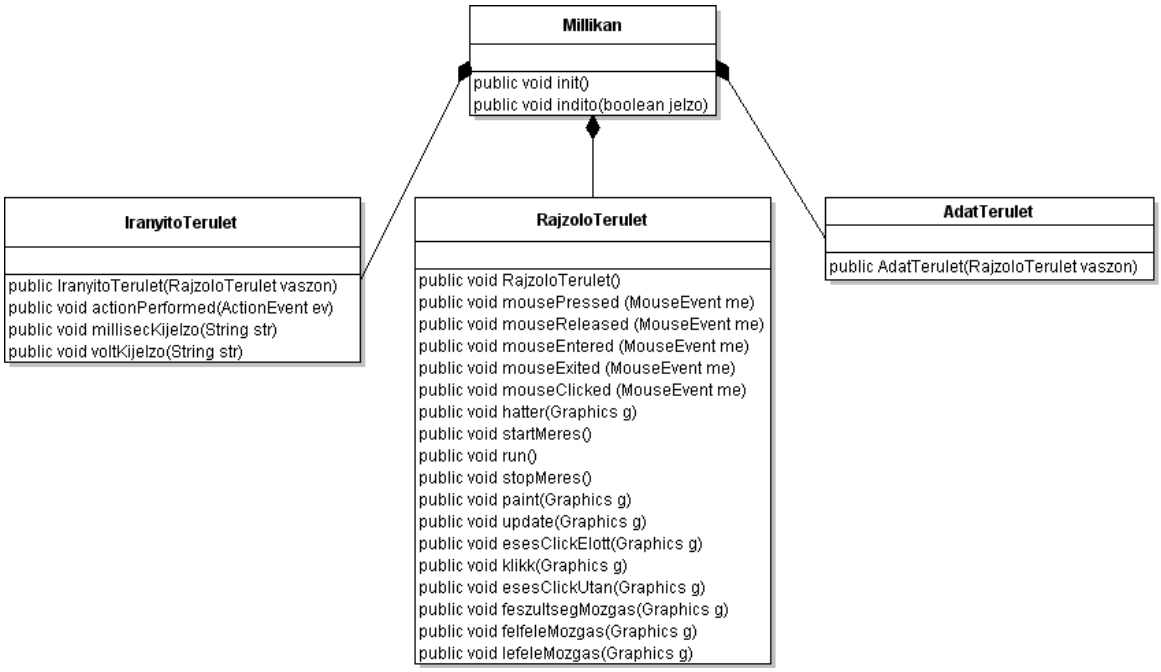




11. ábra. Használati eset diagram

A 11. ábra magyarázatát egy példán keresztül ejteném meg:
Ha az Actor megnyomja például az "Injektálás" gombot, akkor az 3 irányba vált ki hatást. Először történik egy változás az animáció megjelenítésében, megjelennek az újonnan generált cseppecskék a rajzterületen. Továbbá az üzenőterületen megjelenik a "Válassz cseppet" felirat, és végül, ha már volt egy mérés ezelőtt, akkor a kijelzők az irányítóterületen lenullázódnak.

Az osztálydiagram a program struktúrájának megjelenítéséhez:
A nyilak minden esetben erős aggregációt (tartalmazást) jelentenek, a Millikan osztály tartalmazza a másik három osztályt.



12. ábra. Osztálydiagram

A program osztályai, metódusai és attribútumai:

1. A program működéséhez szükséges csomagok, importok:
   - *awt*, a grafikus komponensek működéséhez
   - *applet.Applet*, a program típusa miatt
   - *util*, a véletlenszám generáláshoz, és az időzítéshez
   - *math*, a matematikai műveletekhez

2. A Millikan osztály:
   public class Millikan extends java.applet.Applet {}
   Ez az oszály a szülőosztály, tartalmazza a további három osztályt. Kiterjeszti az Applet osztályt.

   Az osztály attribútumai:
   - *boolean jelzo = false;*
     Logikai érték, a program indítását figyeli.
   - *boolean restart = false;*
     Logikai érték, az újraindítási parancsot figyeli.
   - *boolean infotiltas = false;*
     Logikai érték, az "Info" gomb lenyomását figyeli.

- *boolean meresitiltas = false;*

  Logikai érték, azt figyeli folyik-e éppen mérési folyamat.
- *int meresszam = 0;*

  Egész típusú változó, a mérések számát tárolja.
- *int kurzorpoz = 0;*

  Egész típusú változó, a kiíráshoz tartozó kurzorpozíciót tárolja.

Az osztály metódusai:

- *public void init()*

  A szülőosztály, és egyben az applet *init()* metódusa, a három osztály példányainak elhelyezését végzi az applet területén belül.
- *public void indito(boolean jelzo)*

  Az "Injektálás" gomb megnyomásakor hívódik meg, meghív további metódusokat, és elindítja az animációt.

3. A RajzoloTerulet osztály:

class RajzoloTerulet extends Canvas implements Runnable, MouseListener {}

Az osztály kiterjeszti a *Canvas* osztályt, így lesz képes a kirajzolásra. Implementálja a *Runnable* és a *Mouselistener* interfészeket, hogy képes legyen az animáció futtatására, és az egér-események (például a kattintás) kezelésére.

Az osztály attribútumai:

- *int cseppNagysag;*

  Egész típusú változó, a cseppek megjelenítési nagyságát átmenetileg tárolja.
- *int cseppSzam;*

  Egész típusú változó, a cseppek számát tárolja.
- *double r;*

  Dupla pontosságú lebegőpontos változó, a cseppek sugarát átmenetileg tárolja.
- *int[] xKoordinatak;*

  Egész típusú tömb, a cseppek x koordinátáját tárolja.
- *int[] yKoordinatak;*

  Egész típusú tömb, a cseppek y koordinátáját tárolja.

- *int[] N;*

  Egész típusú tömb, a cseppek töltésének egész szorzóját tárolja.
- *int[] cseppNagysagok;*

  Egész típusú tömb, a cseppek megjelenítési nagyságát tárolja.
- *int[] koordinatakKeres;*

  Egész típusú tömb, az futás alatt a cseppek aktuális y koordinátáját tárolja.
- *double[] cseppSugarak;*

  Dupla pontosságú lebegőpontos tömb, a cseppek sugarát tárolja.
- *double[] cseppToltesek;*

  Dupla pontosságú lebegőpontos tömb, a cseppek töltését tárolja.
- *double[] vEses;*

  Dupla pontosságú lebegőpontos tömb, a cseppek esési sebességét tárolja.
- *int frame;*

  Egész típusú változó, a megjelenítésben van szerepe.
- *int delay;*

  Egész típusú változó, a megjelenítésben van szerepe.
- *final double e = 1.602*Math.pow(10,-19);*

  Véglegesített dupla pontosságú lebegőpontos érték, az elemi töltés értéke.
- *final double Ceses = (9*1.8*Math.pow(10,-5))/(2*9.82*(886-1.3));*

  Véglegesített dupla pontosságú lebegőpontos érték, az esési sebesség meghatározását segítő konstans értéke.

- *final double Cemelkedes = e/(6*Math.PI*1.8*Math.pow(10,-5)*5*Math.pow(10,-3));*

  Véglegesített dupla pontosságú lebegőpontos érték, az emelkedési sebesség meghatározását segítő konstans értéke.
- *int poz;*

  Egész típusú változó, a kiválasztott csepp adatainak tömbökbeli pozícióját tárolja.
- *int clickX;*

  Egész típusú változó, az egérkattintás pontjának az x koordinátáját tárolja.
- *int clickY;*

  Egész típusú változó, az egérkattintás pontjának az y koordinátáját tárolja.

- *double cseppY = 0;*

  Dupla pontosságú lebegőpontos változó, a kiválaszott csepp y koordinátáját tárolja ideiglenesen.
- *double cseppYAktNov = 0;*

  Dupla pontosságú lebegőpontos változó, a kiválaszott csepp y koordinátájának aktuális értékét tárolja a csepp felfelé mozgása alatt.
- *double cseppYAktCsok = 0;*

  Dupla pontosságú lebegőpontos változó, a kiválaszott csepp y koordinátájának aktuális értékét tárolja a csepp lefelé mozgása alatt.
- *double Y = 0;*

  Dupla pontosságú lebegőpontos változó, a feszültség bekapcsolása után a csepp kirajzolását segítő változó.
- *float U = 0;*

  Egyszeres pontosságú lebegőpontos változó, az aktuális feszültségértéket tárolja
- *double vEmelkedes;*

  Dupla pontosságú lebegőpontos változó, a kiválasztott csepp emelkedési sebességét tárolja.
- *double feszkoztes;*

  Dupla pontosságú lebegőpontos változó, a kiválasztott csepp megjelenítését segítő változó, a kiválaszott csepp emelkedési sebességéből származtatott érték.
- *long elteltle;*

  Hosszú egész típusú változó, az időszámlálást segíti a csepp esésekor.
- *long elteltfel;*

  Hosszú egész típusú változó, az időszámlálást segíti a csepp emelkedésekor.
- *long msec;*

  Hosszú egész típusú változó, a számlált időtartamokat ideiglenesen tárolja.
- *String ido;*

  Szöveg típusú változó, a számlált időtartam megjelenítését segíti.
- *String volt;*

  Szöveg típusú változó, a feszültség értékének megjelenítését segíti.
- *boolean click = false;*

  Logikai érték, azt jelzi, hogy történt-e egérkattintást.

- *boolean bennvolt = false;*

  Logikai érték, azt jelzi, hogy az egérkurzor a rajzterületen tartózkodik-e.
- *boolean reset = false;*

  Logikai érték, azt jelzi, hogy van-e újraindítás kérés.
- *boolean ketszaz = false;*

  Logikai érték, a kiválasztott csepp tartózkodási helyét jelzi.
- *boolean haromszazny = false;*

  Logikai érték, kiválasztott csepp tartózkodási helyét jelzi.
- *boolean UBe = false;*

  Logikai érték, azt jelzi, van-e feszültség bekapcsolva.
- *boolean clickvege = false;*

  Logikai érték, azt jelzi, hogy lefutott-e már a kereső algoritmus egérkattintás után.
- *boolean fel = false;*

  Logikai érték, azt jelzi, hogy emelkedik a kiválasztott csepp.
- *boolean le = false;*

  Logikai érték, azt jelzi, hogy esik a kiválasztott csepp.
- *boolean gombnyomas = false;*

  Logikai érték, azt jelzi, hogy történt-e gombnyomás.
- *boolean pluszgombnyomas = false;*

  Logikai érték, azt jelzi, hogy történt-e "Feszültség +" gombnyomás.
- *boolean minuszgombnyomas = false;*

  Logikai érték, azt jelzi, hogy történt-e "Feszültség -" gombnyomás.
- *boolean mehetafesz = false;*

  Logikai érték, azt jelzi, hogy lehet feszültségadást engedélyezni.
- *boolean idomeressiker = false;*

  Logikai érték, azt jelzi, hogy sikeres volt-e az időmérés.
- *boolean kiiras = false;*

  Logikai érték, azt jelzi, hogy történt-e kiírás.
- *boolean replacefel = false;*

  Logikai érték, az iránykiírás aktuális állapotát jelzi.
- *boolean replacele = false;*

  Logikai érték, az iránykiírás aktuális állapotát jelzi.

Az osztály metódusai:

- *public void RajzoloTerulet()*
  Az osztály konstruktora, itt történnek a véletlenszám generálások, és a tömbök feltöltése, a szülőosztály *indito(boolean jelzo)* metódusa hívja meg.
- *public void mousePressed(MouseEvent me)*
  Ez a metódus üres, viszont kötelező létrehozni az egéreseményekkezeléséhez, egyébként az egérgomb lenyomásához kapcsolódó eseményeket lehet benne kezelni.
- *public void mouseReleased(MouseEvent me)*
  Ez a metódus üres, viszont kötelező létrehozni az egéreseményekkezeléséhez, egyébként az egérgomb felengedéséhez kapcsolódó eseményeket lehet benne kezelni.
- *public void mouseEntered(MouseEvent me)*
  Ez a metódus az egérkurzor rajzterületre való belépésekor hívódik meg, ha bevisszük a kurzort a rajzterületre, akkor megáll a mozgás a cseppkiválasztás megkönnyítésére.
- *public void mouseExited(MouseEvent me)*
  Ez a metódus az egérkurzor rajzterületről való kilépésekor hívódik meg, hanem kattintottunk a rajzterületen az egérrel, hanem anélkül kihozzuk a kurzort, akkor folytatódik tovább a cseppek esése.
- *public void mouseClicked(MouseEvent me)*
  Ez a metódus az egérkattintáskor hívódik meg, itt kapja meg a program az egérkattintás helyének koordinátáit.
- *public void hatter(Graphics g)*
  Ebben a metódusban készül el a háttérként kirajzolandó kép, a *paint(Graphics g)* metódus hívja meg.
- *public void startMeres()*
  Az animáció futását indító metódus, itt történik a futtató szál létrehozása is a szülőosztály *indito(boolean jelzo)* metódusa hívja meg.
- *public void run()*
  Feladatokat és a programfutás módját definiáló metódus, itt történik az időmérés is, a szülőosztály *indito(boolean jelzo)* metódusa hívja meg.
- *public void stopMeres()*
  Az animációt leállító metódus, a programszál értékének null-ra állításával

végrehajtva.

- *public void paint(Graphics g)*
  A kirajzolást megvalósító metódus, *repaint()* metóduson keresztül a run metódus hívja meg. A *repaint()* metódus egy a Java által előre definiált metódus, ezért a forráskódban külön nem szükséges definiálni.
- *public void update(Graphics g)*
  Összeállítja a kirajzolandó aktuális képet a *paint(Graphics g)* metódus számára, és itt történik a legtöbb logikai érték kiértékelése is, a *paint(Graphics g)* metódus hívja meg.
- *public void esesClickElott(Graphics g)*
  Az "Injektálás" gomb lenyomása utáni, de egérkattintás előtti cseppmozgatás megvalósítása, az *update(Graphics g)* metódus hívja meg. Ez a metódus számolja ki a cseppek esési sebességeit is, és adja tovább a többi metódusnak.
- *public void klikk(Graphics g)*
  Az egérkattintáskor fut le, megkeresi a kattintás koordinátáihoz legközelebb eső cseppet, szintén az *update(Graphics g)* metódus hívja meg.
- *public void esesClickUtan(Graphics g)*
  Az egérkattintás utáni, de még a feszültség bekapcsolása előtti cseppmozgatást valósítja meg, szintén az *update(Graphics g)* metódus hívja meg. Ebben a metódusban válik engedélyezetté a feszültség bekapcsolása, amint a kiválasztott csepp áthaladt a mérési területen.
- *public void feszultsegMozgas(Graphics g)*
  A feszültség bekapcsolása utáni cseppmozgatás koordinálása a feladata. Itt történik meg a kiválasztott csepp emelkedési sebességének kiszámítása. A metódus attól függően végzi a további metódushívásokat, hogy az emelkedési sebesség pozitív, vagy negatív. Szintén az *update(Graphics g)* metódus hívja meg.
- *public void felfeleMozgas(Graphics g)*
  A *feszultsegMozgas(Graphics g)* metódus hívja meg, ha az emelkedési sebesség pozitív. Feladata a cseppmozgatás, továbbá a kiírandó üzenetek továbbítása az *AdatTerulet* osztály felé.
- *public void lefeleMozgas(Graphics g)*
  A *feszultsegMozgas(Graphics g)* metódus hívja meg, ha az emelkedési

sebesség negatív. Feladata a cseppmozgatás, továbbá a kiírandó üzenetek továbbítása az *AdatTerulet* osztály felé.

4. Az IranyitoTerulet osztály:

class IranyitoTerulet extends Panel implements ActionListener {}

Az osztály kiterjeszti a *Panel* osztályt annak érdekében, hogy gombokat és kijelzőket lehessen általa képernyőre vinni. Továbbá implementálja az *ActionListener* interfészt, hogy képes legyen a gombnyomás-események kezelésére. Ez az osztály végzi a feszültség kijelző és az időtartam kijelző kezelését is.

Az osztály attribútumai:

- *Button injekt;*
  Az "Injektálás" gomb inicializálása.
- *Button feszMinusz;*
  Az "Feszültség -" gomb inicializálása.
- *Button feszPlusz;*
  Az " Feszültség +" gomb inicializálása.
- *Button info;*
  Az " Info" gomb inicializálása.
- *TextField feszKijelzo;*
  A feszültség kijelző inicializálása.

- *TextField szamlaloKijelzo;*
  Az időtartam-kijelző inicializálása.
- *Label szamlaloLabel;*
  Az időtartam-kijelző címkéjének inicializálása.
- *Label feszLabel;*
  Az feszültség kijelző címkéjének inicializálása.
- *Label rejt;*
  Az elemek elhelyezését segítő rejtett címkék inicializálása.

Az osztály metódusai:

- *public IranyitoTerulet(RajzoloTerulet vaszon)*
  Az osztály konstruktora, az applet indulásakor fut le, feladata létrehozni a

gombokat, a kijelzőket és a címkéket, majd elhelyezni őket az irányítóterületen. Továbbá a gombokhoz megfeleltei az eseménykezelő metódust.

- *public void actionPerformed(ActionEvent ev)*
  Az eseménykezelő metódus, ha gombnyomás történik, akkor hívódik meg. Ellenőrzi hogy melyik gombot nyomtuk meg, és a hozzá megfeleltetett feladatokat végzi el. Ez a feladat zömében az, hogy tájékoztassa a többi osztályt a gombnyomás jellegéről.
- *public void millisecKijelzo(String str)*
  Az időtartam kijelzést megvalósító metódus, a *run()* metódus hívja meg, mikor a kiválasztott cseppecske a mérési területen tartózkodik. Ezalatt folyamatosan kapja paraméterként a kijelezni kívánt időt.
- *public void voltKijelzo(String str)*
  A feszültségkijelzést megvalósító metódus, szintén a *run()* metódus hívja meg, amikor feszültségváltozás történt, és paraméterként küldi neki a kijelezni kívánt feszültséget.

5. Az AdatTerulet osztály:
   class AdatTerulet extends Panel {}
   Az osztály kiterjeszti a *Panel* osztályt, így képes lesz egy üzenőterületen kiírni az adatokat és információkat, amire a program futása során szükségünk van.

   Az osztály attribútumai:
   - *TextArea ta;*
     Szövegterület inicializálása, ez a típus több soros és oszlopos szövegek kiírására szolgál.
   - *Label meresi;*
     Az üzenőterület címkéjének inicializálása.

   Az osztály metódusa:
   - *public AdatTerulet(RajzoloTerulet vaszon)*
     Ennek az osztálynak ez az egy metódusa van, a konstruktor. Itt végzi a szövegterület létrehozását, és elhelyezését.

A programozási technika:
A program, az "Injektálás" gomb megnyomása után, vagyis a mérés indítása után, folyamatosan futás alatt van. Az egész működése a *RajzoloTerulet* osztály *run()* metódusa köré épül. Ez a metódus egy folyamatos ciklusként viselkedik, miután a *startMeres()* metódus létrehozza számára az *animator* nevű szálat. Amíg ez a szál él, addig a program fut. Ha nem állítjuk az animator értékét null-ra a forráskódban külön utasítással, akkor azt a futtató applet osztály teszi meg az applet bezárásakor, egy *destroy()* metódus meghívásával. Ebben a programban ez a feladat az applet osztályra van bízva. Azért volt szükség ilyen kivitelezésre, mert folyamatos, valós időben animációt futtató programról van szó. A kirajzolási gyakoriság alapegysége a *frame*. Ez egy változó a programban, ami folyamatosan nő egyel amikor a program belép a run metódusba. Ezt a változót használom a cseppek kirajzolásakor is.

6. Eredmények demonstrációja

Ebben a fejezetben a program teszteléséről fogok írni. Mivel pontosan azért készítettem appletet, hogy felkerülhessen az internetre, ezért a világhálón keresztül teszteltem. Feltöltöttem az alkalmazást a *www.pinterlaci.fw.hu* honlapra, és elkezdtem a tesztelést. Megkértem az egyik, szintén végzős főiskolás ismerősöm, hogy próbálja ki és véleményezze a programot. Mivel eleinte nem ismerte pontosan a Millikan-féle olajcsepp-kísérletet, nem tudta mire is szolgál a program. Ezután én tájékoztattam hogy a program használatához elengedhetetlen, ha csak nagyvonalakban is, hogy ismerjük a kísérletet. Tehát miután megtudta mi is a kísérlet lényege, már tudta használni a programot.

Véleménye szerint egyszerű a használata, és véletlenül se lehet tévútra jutni benne, a megfelelő gomb-engedélyezések miatt.

Ezután elkezdtem én is a tesztelést. Mikor a weboldal betöltődött semmi rendelleneset nem tapasztaltam, az elvártnak megfelelően megjelent az alkalmazás, az üdvözlőszöveggel. Első alkalommal az "Info" gombot választottam. Megnyomása után megjelentek a tájékoztató információk. Az elolvasásuk után rátértem a mérésre. Megnyomtam az "Injektálás" gombot, rendben megjelentek a cseppek. A teszt kedvéért megnyomtam még párszor egymás után többször is, ahogy elvárható volt mindannyiszor újra generálódtak a cseppek. Továbbá a másik három gombot is megnyomtam rögtön az "Injektálás" után többször is, de ahogy kellett nem történt váratlan esemény. A cseppek közül kiválasztottam egyet, a kiválasztás jól működött. Elkezdődött tehát a mérés az első cseppen. Ahogy a csepp a mérőterületre ért, elindult a számláló, egy közepes méretű csepp esési ideje 7407 ms, azaz 7,407 másodperc volt. Esési sebessége 0,0000487374 $\frac{m}{s}$, azaz 0,0487374 $\frac{mm}{s}$, ami reális érték a mi esetünkben. Ugyanennek a cseppnek az emelkedési sebessége 0,0001333761 $\frac{m}{s}$, azaz 0,1333761 $\frac{mm}{s}$ volt 0,05 kV-os feszültség mellett. Láthatjuk, hogy az emelkedési sebesség is reális keretek között maradt. A mért idő a csepp felfelé haladása során 2813 ms, azaz 2,813 másodperc lett. Azonnal felfedezhetjük hogy az érték helyes, mivel az emelkedési sebesség a többszöröse volt az esésinek, tehát kevesebb időt kellett hogy töltsön a csepp a mérési területen. A csepp töltése 41,652 * 10^{-19} C lett, ami szintén egy reális érték. A mérés közben többször változtatva a feszültséget, ezáltal a cseppet váltakozó mozgási irányokba kényszerítve, figyelemmel kísértem a mozgási irány jelzését az üzenőterületen, és helyes működést tapasztaltam. Az "Injektálás" gomb újbóli megnyomására elindult rendben a következő mérés. Az eredmények a következők lettek:

- Esési sebesség: 0,000100973 $\frac{m}{s}$
- Emelkedési sebesség 0,1 kV mellett: 0,000142339 $\frac{m}{s}$
- A csepp töltése: 40,05 * 10^{-19} C

A további mérésekből már csak a töltés értékeket feljegyezve, elértem a program végcéljaként kitűzött eredményt:

1. mérés: 41,652 * 10^{-19} C
2. mérés: 40,05 * 10^{-19} C
3. mérés: 65,682 * 10^{-19} C
4. mérés: 65,681 * 10^{-19} C
5. mérés: 19,224 * 10^{-19} C
6. mérés: 54,468 * 10^{-19} C

Sikerült összegyűjteni csepptöltés-értékeket, amelyek az elemi töltés megállapításának célját szolgálják. Természetesen bármennyi töltésértéket össze lehet gyűjteni, egy méréssorozattal, nincs a programban erre vonatkozó korlát.

7. Összefoglalás

Így a tesztelés után, friss tapasztalatokkal elérkeztem az összefoglaláshoz. A tesztet sikeresnek találtam, a program a kitűzött célt teljes mértékben teljesíti, talán a többlet-információk kiírásával túl is lépi azt. Hiszen az eredeti célkitűzések között konkrétan nem szerepelt sem számláló, sem pedig a sebességek kiíratása. De mivel ez a lehetőség megvolt a programban, gondoltam kihasználom, és érdekesebbé, interaktívabbá teszem vele. Hiszen

valamilyen szintű megelégedettséggel tölti el a felhasználót az, hogy a gombnyomogatás közben azonnal láthatja az eredményeket, és nem csak a mérés végén kap egy kidobott számot.

További lehetőségeket és fejlesztési pontokat is felfedeztem a programban, de azok véghezviteléhez, újabb, komolyabb eszmefuttatások szükségesek. Például egy ilyen pont az, hogy a cseppecskék kinézetét finomítani lehetne, esetleg valódi cseppalakká fejleszteni. De akár még több információ kiírását is szem előtt lehet tartani, viszont azzal már a program végcélja, megváltozna. A bevezetőben kezdő programozóként utaltam magamra, úgy gondolom, ez egyelőre így is marad, de rengeteget tanultam a munka során a programozásról, és persze a Java nyelvről. Azonban további önképzéssel, szorgalommal úgy érzem programozóvá is válhatok. Sajnos ma már az appletek helyét átveszik a különböző webes fejlesztések és a Flash animációk, de a programozási alapokat elsajátítani tökéletesnek éreztem. Végszóként leginkább fizika oktatási célra ajánlanám a programot, szemléltetőnek, hogy hogyan is néz ki, mi is a lényege Millikan fizikatörténelmet író kísérletének.

8. Irodalomjegyzék

Nyékyné Gaizler Judit (szerkesztő): Java 2 útikalauz programozóknak I. A kávé,
ELTE TTK Hallgatói Alapítvány, Budapest, 2000.

Nyékyné Gaizler Judit (szerkesztő): Java 2 útikalauz programozóknak II. A hab,
ELTE TTK Hallgatói Alapítvány, Budapest, 2000.

Nyékyné Gaizler Judit (szerkesztő): Java 2 referencia programozóknak: 1.3 A csésze, ELTE TTK Hallgatói Alapítvány, Budapest, 2001.

Szabó Dániel (szerkesztő): A 95 legjobb Java-applet,
Panem Kiadó, Budapest, 1999.

http://hu.wikipedia.org/wiki/Robert_Millikan

http://surphy.fat.bme.hu/pub/Fizlabor_III/Millikan%2020070215.pdf

http://www.ibela.sulinet.hu/atomfizika/millikan/millikan.htm

http://hu.wikipedia.org/wiki/Olajcseppkísérlet

http://indy.poliod.hu/program/java/html31.htm

9. Mellékletek

1 db CD-ROM

Tartalma:
- Forráskód: Millikan.java
- Project állomány: MillikanProject
- Futtatható állomány: Millikan.html a hozzátartozó fájlokkal

Melléklet

A mérési eredmények elemzése

Ebben a mellékletben kifejteném, hogy hogyan is lehet elvégezni a program által szolgáltatott cseppértékekből az elemi töltés meghatározását. A szemléltetéshez a 6. fejezetben kapott töltés értékeket fogom felhasználni, amelyek a következők:

1. mérés: $41{,}652 * 10^{-19}$ C

2. mérés: $40{,}05 * 10^{-19}$ C

3. mérés: $65{,}682 * 10^{-19}$ C

4. mérés: $65{,}681 * 10^{-19}$ C

5. mérés: $19{,}224 * 10^{-19}$ C

6. mérés: $54{,}468 * 10^{-19}$ C

A megfelelő matematikai módszer az elemi töltés meghatározásához, a legkisebb osztó megkeresése. Kezdésként a kapott töltés értékekből kiválasztom a legkisebbet ($19{,}224 * 10^{-19}$) , feltételezvén, hogy akár ez is lehet az elemi töltés értéke. A továbbiakban lehagyom a 10^{-19} -es szorzót, mert az eredményeket ebben az esetben nem befolyásolja. Ezután az 1-es számú töltés értéket elosztom a legkisebbel, vagyis 41,652/19,224-el. Ha egész számot kapok, akkor erősödik a gyanú, hogy ez lesz az elemi töltés értéke. Az eredmény 2,1 lett, vagyis nem egész szám. Ezután felesleges 19,224-el tovább osztania többi töltésértéket is, mert már biztos, hogy nem ez lesz az elemi töltés értéke. A következő lépés az, hogy megpróbálom a 19,224 felével az osztást, azaz 9,612-vel. Szintén az 1-es számú töltés értéket osztom először. Az eredmény 4,3 lett, ami megint nem egész szám, vagyis a 9,612 sem az elemi töltés értéke. A továbbiakban ugyanezt ismételve az eredmények:

19,224/3=6,408 → 41,652/6,408=6,5 → nem egész, 6,408 nem elemi töltés

19,224/4=4,806 → 41,652/4,806=8,6 → nem egész, 4,806 nem elemi töltés

19,224/5=3,844 → 41,652/3,844=10,8 → nem egész, 3,844 nem elemi töltés

19,224/6=3,204 → 41,652/3,204=13 → egész, következő töltés érték osztása

→ 40,05/3,204=12,5 → nem egész, 3,204 nem elemi töltés

19,224/7=2,746 → 41,652/2,746=15,1 → nem egész, 2,746 nem elemi töltés

19,224/8=2,403 → 41,652/2,403=17,3 → nem egész, 2,403 nem elemi töltés

19,224/9=2,136 → 41,652/2,136=19,5 → nem egész, 2,136 nem elemi töltés

19,224/10=1,922 → 41,652/1,922=21,6 → nem egész, 1,922 nem elemi töltés

19,224/11=1,747 → 41,652/1,747=23,8 → nem egész, 1,747 nem elemi töltés

19,224/12=1,602 → 41,652/1,602=26 → egész, következő töltés érték osztása

→ 40,05/1,602=25 → egész, következő töltés érték osztása

→ 65,682/1,602=41 → egész, következő töltés érték osztása

→ 65,681/1,602=40,999 → 0,001-es hiba a kerekítések miatt megengedett, a szám egésznek tekinthető, következő töltés érték osztása

→ 19,224/1,602=12 → egész, következő töltés érték osztása

→ 54,468/1,602=34 → egész

Az 1,602 lett a töltésértékek legkisebb közös osztója, tehát megtaláltuk az elemi töltés értékét, ami 1,602 * 10^{-19}C. Mint láthatjuk, helyes értékre jutottunk, a kapott elemi töltés megegyezik az ismert értékkel.