

Tallinna Reaalkool

Atmosfääri mudeldamine ja katseandmete põhjal täpseima mudeli leidmine

Uurimistöö

Jarl Patrick Paide

11.a

Juhendaja: õp Mart Kuurme

Tallinn 2019

Sisukord

Sissejuhatus	4
1 Mudeli teoreetilised alused	5
1.1 Adiabaatiline protsess	6
1.2 Adiabaatiline temperatuurigradiant	8
1.3 Rõhu muutus kõrgusega	9
1.4 Veeauru kondenseerumine	9
1.5 UV kiirgus stratosfääris	10
2 Katse ülesehitus	11
2.1 Katse eesmärk	11
2.2 Katsemetoodika valik	11
2.3 Katse planeerimine	12
2.4 Katseseadmed	14
2.5 Katse läbiviimine	15
3 Katseandmete analüüs	17
3.1 Lennu asukohaline ülevaade	17
3.2 Temperatuuri muutus kõrgusega	19
3.3 Rõhu muutus kõrgusega	26
3.4 Mitteadiabaatilised vahemikud	28
3.5 Järeldus	29
Kokkuvõte	30
Kasutatud materjalid	31
Lisa 1 Andmete kogumise kood	32
Lisa 2 Sensori kood	38

Lisa 3 Andmete analüüs kood	42
Lisa 4 Logifail	44
Kinnitusleht	45
Resümee	46
Abstract	47

Sissejuhatus

Globaliseeruvas ja ülerahvastatud maailmas on Maa atmosfääri reostatus üks kõige olulisemaid probleeme. Atmosfääri reostusega kaasneb kasvuhooneefekt - kliima soojenemine, mis omakorda viib maailmamere tõusule. Atmosfääri mudeldamine aitab mõista atmosfääris toimuvaid protsesse ja leida lahendusi atmosfääri seisundi parandamiseks. Mudeli andmeid saab kasutada globaalse atmosfääri mudeli väljatöötamisel.

Uurimistöö eesmärk on leida vaatlusandmete alusel võimalikult täpne mudel, mis kirjeldaks atmosfääri temperatuuri ja rõhu seoseid vastavalt kõrgusele maapinnast. Uurimisküsimus on "Millised seosed on atmosfääris mõõdetavate parameetrite vahel - temperatuur, rõhk ja kõrgus maapinnast?".

Uurimistöö hüpotees on, et atmosfääri madalamates kihtides toimuvad termodünaamilised protsessid on adiabaatilised protsessid.

Uurimistöö alguses leitakse eeldusel, et atmosfääris toimuvad termodünaamilised protsessid on adiabaatilised protsessid, seosed temperatuuri, rõhu ja kõrguse vahel.

Praktilises osas tehakse mõõtmisi heeliumõhupalli külge kinnitatud mõõteriistaga, mis lennutatakse stratosfääriini ja pärast kontrollitakse katseandmete põhjal teoreetilises osas saadud seoste kehtivust.

Uurimistöö autor soovib tänada kõiki katse korraldamisel ja läbiviimisel aidanud inimseining uurimistöö juhendajat, kes andis head nõu uurimistöö teostamisel.

1 Mudeli teoreetilised alused

Käesolevas osas leitakse seosed, kuidas kirjeldada atmosfääris rõhu ja temperatuuri sõltuvust kõrgusest.

Atmosfääris tekib rõhk kõrgemal olevate õhukihtide raskusjõu poolt tekkinud jõust. Kuna kõrguse kasvades vähneb kõrgemal asuva õhu mass, siis väheneb ka rõhk kõrguse kasvades. Rõhu erinevus erinevatel kõrgustel toob kaasa rõhkude vahest tingitud ülespoole suunatud jõu, mida tasakaalustab gravitatsioonijõud. Rõhk muutub dp võrra kõrguse dz võrra kasvades, kui õhu tihedus kõrgusel z on ρ , järgnevalt (Wallace, Hobbs 2006: 67–68):

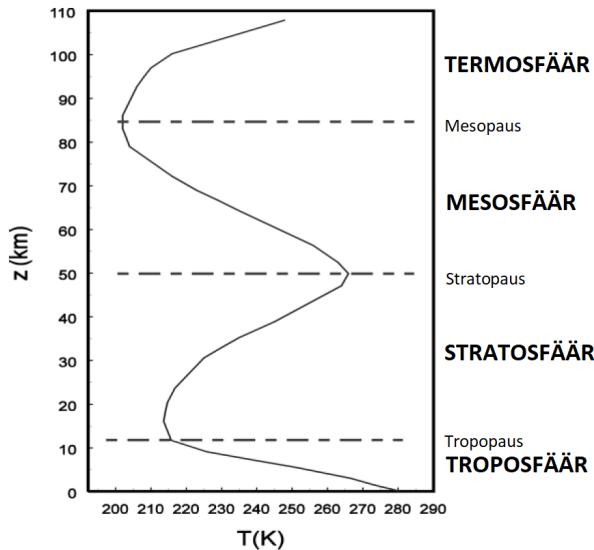
$$dp = -\rho g dz. \quad (1)$$

On olemas erinevaid gaasi mudeliteid. Nendest kõige tuntum on ideaalse gaasi olekuvõrrand. Ideaalne gaas erineb reaalsetest gaasist kahe eelduse poolest. Ideaalses gaasis ei arvestata osakestevahelisi vastastikjõude ja ideaalses gaasis lihtsustatakse, et osakestel on tühiselt väike suurus. Võib eeldada, et gaas on ideaalne siis, kui rõhk on väike vörreldes kriitilise rõhuga ja temperatuur on kõrge vörreldes kriitilise temperatuuriga. Kui rõhk on kõrge, siis on osakestevahelisi kokkupõrkeid palju ja osakeste suurus saab oluliseks. Kui temperatuur on madal, siis liiguvad osakesed aeglaselt ja osakestel on rohkem aega olla üksteise mõjuväljas ja saada mõjutatud. Kui vaadata atmosfääri, siis kõige kõrgem rõhk on Maa lächedal ja kõrguse suurenedes see väheneb, millega väheneb ka osakestevaheline vastasmõju. Atmosfääris olevad rõhud on väikesed, vörreldes õhu kriitilise rõhuga. Temperatuur võib langeda atmosfääris küll madalale, aga mitte piisavalt madalale, et see läheneks kriitilisele temperatuurile. Seega võib eeldada, et atmosfääris olevad gaaside käituvad kui ideaalsed gaaside. Ideaalse gaasi olekuvõrrand on:

$$pV = \nu RT, \quad (2)$$

kus p on gaasi rõhk, V on gaasi ruumala, $\nu = \frac{m}{\mu}$ on gaasi kogus moolides, kus m on gaasi mass ja μ on gaasi molaarmass, R on universaalne gaasikonstant ja T on gaasi temperatuur (Khan 2016).

Kuna selle uurimistöö käigus koguti katseandmeid troposfääris ja stratosfääris, uuritakse ainult neid piirkondi. Joonisel 1 on näha, kuidas temperatuur muutub kõrguse kasvades. Stratopausis on üks temperatuuri maksimume. Seal olev kõrge temperatuur on otse tingitud UV kiirguse neeldumisest osoonikihis. Kuigi osooni kih on kõige tihedam madalamal stratosfääris umbes 20 km ja 30 km vahel, on temperatuur ikkagi kõige kõrgem umbes 50 km juures. See on tingitud osooni kih läbipaistmatusest UV kiirguse jaoks, mistõttu suurem osa kiirgusest neeldub kõrgemal. Troposfääri ei ole mõjutatud Päikese kiirgusest, kuna see on läbipaistev. Erinevalt stratosfäärist ringleb õhk troposfääris palju. Troposfääri soojendab maapind, mida soojendab Päike. Soojalt maapinnalt tõustes hakkab rõhk vähenema ja ideaalse gaasi olekuvõrrandi kohaselt hakkab temperatuur langema (Marshall, Plumb 1959: 25–26).



Joonis 1. Temperatuuri sõltuvus kõrgusest

Allikas: Marshall, Plumb 1959

Tropsfääris on õhk pidevas liikumises. Ning kuna troposfääris soojeneb õhk ainult maapinna lähedal, siis võib oletada, et termodünaamilised protsessid, mis toimuvad õhu üles ja alla liikumisest, mille käigus muutub õhu rõhk, on adiabaatilised protsessid, ehk õhu vahel soojusvahetust ei toimu.

1.1 Adiabaatiline protsess

Termodünaamika esimene seadus on

$$dU = dQ - dA, \quad (3)$$

kus dU on gaasi siseenergia muutus, dQ on soojushulk ja dA on töö. Gaasi siseenergia U avaldub vabadusastmete i kaudu järgneva seose abil:

$$U = \frac{i}{2} \nu R T. \quad (4)$$

Konstantse ruumala puhul tööd ei tehta, seega kogu soojus läheb siseenergia suurendamiseks. Saame soojusmahutavuse, võttes siseenergia muudust tuletise temeratuuri järgi:

$$C_V = \frac{dQ}{dT} = \frac{i}{2} \nu R. \quad (5)$$

Molaarset soojusmahutavust saab avaldada valemiga $c_V = \frac{C_V}{\nu}$, saades molaarseks soojusmahutavuseks

$$c_V = \frac{i}{2} R. \quad (6)$$

Kui aga vaadata isobaarilist protsessi, siis olekuvõrrandist tuleneb: $pdV = \nu R dT$, ning gaas teeb tööd $dA = pdV = \nu R dT$. Avaldades need valemisse 3 saadakse:

$$dQ = dU + dA = \frac{i+2}{2} \nu R dT,$$

millest järeltub:

$$c_p = \frac{i+2}{2} R.$$

Samuti saab näidata c_V ja c_p vahelist seost:

$$c_p = \frac{i+2}{2} R = \frac{i}{2} R + R = c_V + R. \quad (7)$$

Adiabaatiline protsess on termodünaamiline protsess, mille käigus ei toimu soojusvahetust väliskeskkonnaga, seega valemis 3 $dQ = 0$. Gaasi poolt tehtud töö on $dA = pdV$ ja gaasi siseenergia muut on $dU = \nu c_V dT$. Sellest järeltub:

$$\nu c_V dT = -pdV. \quad (8)$$

Ideaalse gaasi olekuvõrrandist 2 saadakse tuletist võttes ja dT avaldades järgneva seose:

$$dT = \frac{pdV + Vdp}{\nu R}. \quad (9)$$

Asendades valemi 9 valemisse 8 saadakse uus seos:

$$pdV(c_V + R) + c_V V dp = 0. \quad (10)$$

Asendades siia sisse valemi 7 ja adiabaadi näitaja $\gamma \equiv \frac{c_p}{c_v}$ saadakse võrrand

$$\gamma \frac{dV}{V} + \frac{dp}{p} = 0.$$

Seda integreerides saadakse uus võrdus:

$$\int \gamma \frac{dV}{V} + \frac{dp}{p} = \gamma \ln(V) + \ln(p) = Const.$$

Sellest saab järellelada

$$pV^\gamma = Const.$$

Samuti, kasutades ideaalse gaasi olekuvõrrandit, saab tuletada järgneva seose:

$$p^{1-\gamma} T^\gamma = Const. \quad (11)$$

Seega, kui vaadata mingit kogust gaasi adiabaatilises protsessis, siis jääb antud võrduse väärustus gaasi parameetrite muutumisel samaks (Kalda 2014).

1.2 Adiabaatiline temperatuurigradient

Selles osas tuletatakse temperatuuri gradient adiabaatilise protsesi puhul. Valemist 11 saadakse

$$d \ln(p^{1-\gamma} T^\gamma) = 0,$$

millest saadakse

$$\frac{dp}{p} = \frac{\gamma}{\gamma - 1} \frac{dT}{T}. \quad (12)$$

Asendades seosed 1 ja 2 seosesse 12 saadakse temperatuurigradient:

$$\Gamma \equiv \frac{dT}{dz} = -\frac{\gamma - 1}{\gamma} \frac{\mu g}{R} = -\frac{R}{c_p} \frac{\mu g}{R} = -\frac{\mu g}{c_p}. \quad (13)$$

Valem 13 kehtib juhul, kui õhus pole veeauru. Kui õhus on küllastumata veeaur, toimuvad atmosfääris ikkagi adiabaatilised protsessid. Sellisel juhul kehtib valem

$$\Gamma = -\frac{g}{c_m},$$

kus c_m niiske õhu erisoojus. Erisoojus avaldub molaarsest erisoojusest kujul $c_m = \frac{c_p}{\mu}$. Kui asendada nüüd valemisse õhu erisoojuse osakaalud, saadakse lõplikuks valemis

$$\Gamma = -\frac{g}{(1 - \omega)c_o + \omega c_v}, \quad (14)$$

kus c_o on õhu erisoojus, c_v on vee auru erisoojus ja ω on vee massiosakaal (Boeker, Grondelle 2011: 36).

1.3 Rõhu muutus kõrgusega

Integreerides valemit 13 saadakse

$$\int_{T_0}^T dT = \int_0^z \Gamma dz$$

$$T - T_0 = \Gamma z,$$

kus T_0 on temperatuur algpunktis ja T on temperatuur kõrgusel z algpunktist. Avaldist teisendades saadakse:

$$T = T_0 \left(1 + \frac{\Gamma}{T_0} z\right).$$

Kasutades nüüd seost 11, saab eelmise valemi ümber kirjutada kujul:

$$p = p_0 \left(1 + \frac{\Gamma}{T_0} z\right)^{\frac{\gamma}{\gamma-1}}.$$

Astendaja saab asendada kujuga

$$\frac{\gamma}{\gamma-1} = \frac{c_p}{R} = -\frac{g\mu}{\Gamma R},$$

saades lõplikuks valemiks:

$$p = p_0 \left(1 + \frac{\Gamma}{T_0} z\right)^{-\frac{g\mu}{\Gamma R}}.$$

Seega, kui on teada algpunktis olev rõhk p_0 , temperatuur T_0 ja temperatuurigradiant Γ , on võimalik leida seda valemit rõhk kõrgusel z algpunktist (Fitzpatrick 2006).

1.4 Veeauru kondenseerumine

Kui niiske õhk tõuseb kõrgemale, siis langeb temperatuur ja rõhk. Kui õhk jahtub ja rõhk väheneb, siis väheneb ka õhu võime hoida vett auruna endas ja veeauru kondenseerub väga väikesteks tilkadeks. Veeaurul on lihtsam kondenseeruda, kui veeaur saab kondenseeruda osakese külge. Nendeks osakesteks on tavaliselt tolm või õietolm. Kui piisavalt palju veeauru kondenseerub väikeste osakeste külge, siis moodustub pilv (Wallace, Hobbs 2006).

Vee kondenseerumisel pilvedes eraldub soojus, seega pilvedes toimuv termodünaamiline protsess ei vasta adiabaatilisele protsessile. Kuid kuna väljaspool pilvi veeaur ei kondenseeru, siis pilvedest madalamal ja kõrgemal termodünaamilised protsessid on adiabaatilised protsessid.

Järgnevalt leitakse suhe õhust välja aurustunud vee ja õhu masside vahel. Temperatuuri gradient vahetult pilve all on Γ , temperatuur pilve all on T_0 , temperatuur pilvede kohal z võrra kõrgemal algtemperatuurist on T_2 , vaadeldava õhu mass on m_a ja sellest õhust kondenseerunud õhu vee mass on m_v . Kui pilvi ei oleks, siis muutuks temperatuur edasi vastavalt temperatuurigradiendile. Seega temperatuur oleks T_2 asemel

$$T_1 = T_0 + \Gamma z.$$

Vearu annab kondenseerumisel ära energia

$$\Delta U = Lm_v$$

kus L on aurustumissoojus. See energia läheb õhu siseenergia suurendamiseks. Gaasi siseenergiat saab arvutada valemiga 4, seega siseenergia muut on

$$\Delta U = \frac{i}{2} \frac{m_a}{\mu} RT_2 - \frac{i}{2} \frac{m_a}{\mu} RT_1 = \frac{i}{2} \frac{m_a}{\mu} R (T_2 - T_0 - \Gamma z).$$

Seega saab avaldada masside suhte järgnevalt:

$$\frac{m_v}{m_a} = \frac{i}{2} \frac{R}{\mu L} (T_2 - T_0 - \Gamma z).$$

1.5 UV kiirgus stratosfääris

Troposfäärist kõrgemal asub stratosfääri. Kuna UV kiirguse neeldumisel eralduv soojus antakse õhule juurde, siis pole tegu enam adiabaatilise protsessiga. Järgnevalt võrreldakse olukorda, kus UV kiirgus ei neeldu stratosfääris, tegeliku olukorraga, kus UV kiirgus neeldub stratosfääris. Arvutatakse gaasi siseenergiate suhe nendes erinevates olukordades. Temperatuur stratosfääri ja troposfääri vahelisel alal on T_0 ja temperatuurigradienit samal kõrgusel on Γ . Temperatuur z võrra kõrgemal stratosfääris on T_2 . Kui UV kiirgus ei neelduks, siis temperatuur langeks edasi temperatuurigradiendi järgi. Sellisel juhul oleks temperatuur z võrra kõrgemal algpunktist

$$T_1 = T_0 + \Gamma z.$$

Siseenergia avaldub kujul:

$$U = \frac{i}{2} \gamma RT.$$

Seega on siseenergia

$$\frac{U_2}{U_1} = \frac{\frac{i}{2} \gamma RT_2}{\frac{i}{2} \gamma R (T_0 + \Gamma z)} = \frac{T_2}{T_0 + \Gamma z}$$

korda suurem siis, kui UV kiirgus neeldub, võrreldes olukorraga, kui UV kiirgust ei neeldu.

2 Katse ülesehitus

Katse käigus mõõdeti atmosfääris, troposfääris ja stratosfääri madalamates kihtides rõhk, temperatuuri ja õhuniiskust. Selle jaoks kasutati heeliumõhupalli külge kinnitatud sondi, mis tegi mõõtmisi. Andmeid koguti heeliumiga täidetud õhupalliga kaasa saadetud sondiga, mis mõõtis erinevaid andmeid.

Põhilisteks andmeteks on kõrgus, asukoht, aeg, välistemperatuur ja rõhk. Sondi pardal oli Raspberry Pi arvuti. Õhupall kerkis atmosfääri kõrgematesse kihtidesse, sest üleslükkejõud ületab kerge gaasi ja sondi massi poolt tekitatud raskusjõu. Kõrgemale tõustes rõhk väheneb. Et õhupalli siserõhk oleks tasakaalus välisrõhuga, suureneb õhupalli ruumala, kuni õhupall lõhkeb ülepingest. Peale seda kukub sond alla ja leitakse GPS jälgija abiga üles.

2.1 Katse eesmärk

Katsel oli kaks eesmärki. Üks eesmärk oli käesoleva uurimistöö jaoks koguda atmosfäärist andmeid, et testida reaalsete katseandmete kokkulangemist teoreetiliselt tuletatud valemitega. Teiseks tekkis idee tuua midagi atmosfäärist kaasa ja see idee suunati edasi Reaalkooli põhikooli õpilastele, kes arendasid ideed edasi ja otsustasid stratosfäärist õhku läbi filtri juhtida ja sellega koguda stratosfäärist tolmu ja muid suuremaid osakesi. Tolmu kogumine ei käi selle uurimistöö juurde.

Esimeses peatükis koostatud mudeli kontrollimiseks oli vaja koguda atmosfääri erinevate kõrguste andmeid temperatuuri, rõhu ja õhuniiskuse kohta.

2.2 Katsemetoodika valik

Selles uurimistöös tehtud katsed on tehtud koostöös Eesti kosmosekoolide võrgustikuga (kosmosekoolid.ee), mille asutaja Väätja põhikool oli enne selle uurimistööga tehtud lendu teinud kaks lendu. Esimene lend, mis tehti 29. märtsil 2018, kestis umbes kaks tundi, kus kõrgeim punkt, milleni jõuti, oli 26 474 m. Sellele kõrgusele jõuti pooleteise tunniga.

Kokku tehti 220 mõõtmist. Mõõdeti kellaega, laiuskraadi, pikkuskraadi, kõrgust, kapsli sisetemperatuuri, välistemperatuuri ja rõhku. Teine lend tehti 11. mail 2018 ning seekord kestis lend peaaegu neli tundi. Kolme ja poole tunniga jõudis sond kõrgusele 32 608 m. Katse tegijad vähendasid teisel lennul õhupallis heeliumi kogust, mille tõttu oli tõusmise kiirus väiksem, aga õhupall lõhkes hiljem ja jõuti kõrgemale. Teisel lennul tehti 4300 mõõtmist ja mõõdeti samu parameetreid.

Katse korraldamiseks ja läbiviimiseks vajalikud oskused saadi Eesti kosmosekoolide võrgustiku poolt korraldatavalt koolituselt ja sealt on pärit ka vastav metoodika, kuidas katselendu läbi viia.

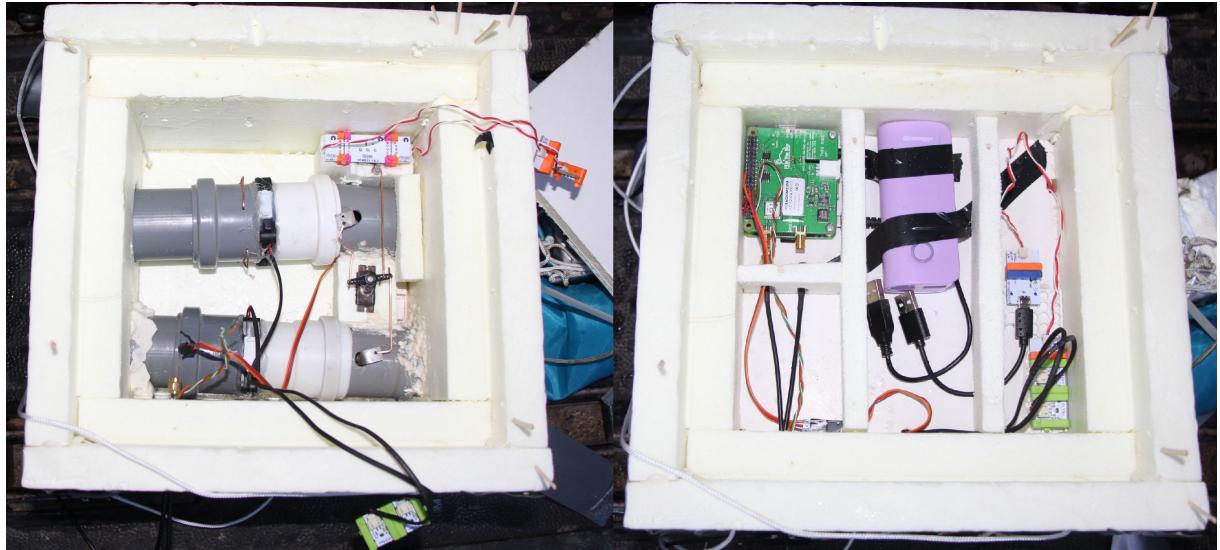
2.3 Katse planeerimine

Alguses oli plaanis heelumpall lendu lasta Väätsa staadionilt, aga kasutades Internetis olevat kalkulaatorit leheküljel <http://predict.habhub.org/>, oli ennustusest näha, et sond oleks kukkunud Jõhvi lächedale. Kuna eksimusruum võib olla suur ning Venemaa piir ja meri ei olnud kukkumiskohast kaugel, siis otsustati lennu start teha võimalikult edelast nii, et edelatuultega kalduks sond Kesk-Eestisse. Rihtides lõppsihtkohta Paide peale, otsustati start teha Varblast Pärnumaalt.

Lennu jaoks oli vajalik saada Lennuametilt kooskõlastus. Selle jaoks kirjutati nädal aega enne planeeritud lendu ja küsiti kooskõlastust asukoha järgi. Kuna viimasel hetkel stardi asukoht muutus, pidi uue kooskõlastuse küsimä, aga see ei olnud probleem. Vahetult enne lendu pidi telefoni teel saama viimase kinnituse stardiks.

Sondi ehitamise eest võttis vastutuse enda peale Tallinna Reaalkooli põhikooli õpilaste robootikameeskond Viirus. Sondi põhiehitusmaterjaliks oli penoplast ja puidust tikud. Sond pidi hoidma sisetemperatuuri madalate välistemperatuuride eest, et sisemuses olev elektroonika töötaks. Samuti pidi korpus vastu pidama kukkumisele ning sondi korpus pidi olema võimalikult kerge. Selle tõttu valiti korpuse materjaliks penoplast ja tikud. Sondi sees olev ruum oli jaotatud kaheks. Sondi sisemus on näidatud joonisel 2, kus vasakul on sondi alumine osa ja paremal on ülemine osa. Alumises olid kaks toru, mis olid paralleelsed ja läbisid sondi kere. Nende sees olid klapid, ventilaator ja filtripid. Ülemises osas oli kogu tehnika. Mõõtmisi tegi ja klappe avas Raspberry Pi arvuti. Voolu andis nii ventilaatoritele

kui ka Raspberry Pi'le akupank. Raspberry Pi külge oli kinnitatud raadioantenn, GPS-antenn, mootor klappide jaoks ja andur BME280, mis mõõtis õhurõhku, temperatuuri ja õhuniiskust. Sondi mass oli 1375 g. Sondi välimus on joonisel 3. Sondist tuli välja nii



Joonis 2. Pilt sondist

Allikas: http://pildid.real.edu.ee/main.php?g2_itemId=84935

raadio- kui ka GPS-antenn ning ka andur BME280. Sondi peale oli kinnitatud langevari. Eraldi korpuses asus GPS jälgija GL300, mida kasutati pärast sondi leidmisel. Lennuks kasutati Hwoyee 600 g õhupalli.

Autor vastutas kogu tehnika töötamise eest. Autor paigaldas sondi esimesele korrusele kogu vajaliku tehnika, ühendas sensori ja mootori Raspberry Pi külge ja seadis sondi lennuvalmis. Autor kirjutas arvutiprogrammi, mis mõõtis andmeid, saatis andmeid ja avas õigel hetkel klapid tolmu kogumiseks. Autor seadis üles arvuti ja antenni, millega saadi lennu käigus jooksvalt andmeid kätte.



Joonis 3. Pilt sondist

Allikas: http://pildid.real.edu.ee/main.php?g2_itemId=84935

2.4 Katseeadmed

Kogu tehnelist poolt juhtis Raspberry Pi arvuti. Raspberry külge kinnitati lisaks Pi In The Sky (PITS) plaat. PITS plaadi külge kinnitati GPS-antenn, mille järgi saadi teada geograafilisi koordinaate, kõrgust ja kellaaega, ja raadioantenn, millega saadeti mõõdetud andmed maapinnale, et jooksvalt jälgida sondi lendu. Raspberry Pi arvuti küljes oli temperatuuri andur, millega mõõdeti sisetemperatuuri. Raspberry Pi külge kinnitati BME280 sensor. Sensor mõõtis temperatuuri, õhurõhku ja õhuniiskust. Sensor viidi juhtmetega sondist välja, et mõõta välistingimusi, mitte sondi sisetingimusi. Raspberry Pi külge kinnitati ka väike mootor. Mootori pööramisel avanesid klapid ja sulgus vooluring, millega pandi ventilaatorid tööle. Sondi sees oli akupank, mis oli ühe juhtmega ühendatud Raspberry Pi külge toiteks ja teise juhtmega ühendatud vooluringi, kus asusuid ventilaatorid.

Andmeid kogus käesoleva uurimistöö autori poolt kirjutatud programm. Programm leidis GPS-antenni kaudu enda asukoha ja lisas sinna sensori poolt mõõdetud tulemused.

Saadud andmerea salvestas programm logifaili ja lisaks saatis PITS plaadi külge kinnititud raadioantenni kaudu info laiali. Programm kontrollis igal ajahetkel kõrgust ja kui see ületas 20 km, siis saatis programm signaali mootorile, pöörates mootorit, millega hakkati tolmu koguma. Kui kõrgus oli sellest väiksem, siis pandi mootor tagasi algasendisse.

Raadiosignaal saadi kätte raadioantenniga, mille signaal edastati arvutisse. Kasutades tarkvaralist raadiot, muudeti saadud signaal heliks ja suunati virtuaalse helijuhtme abil helikaardi dekodeerimistarkvarra. Seal muuudeti heli tekstiks, kust oli võimalik välja lugeda mõõdetud andmed.

Eraldi väikessesse korpusesse pandi GL300 jälgija ja kinnitati suurema korpuse külge. Jälgija pandi eraldi korpusesse, et signaalid erinevate seadmete vahel ei hakkaks segama üksteist. Jälgija kasutas GPS'i, et leida oma asukohta ja siis saatis selle mobiilset andmesidet kasutades Internetti, kust oli võimalik teada saada jälgija asukohta. Seade pandi sondiga kaasa, et pärast maandumist lihtsalt sond üles leida. Kuna nõrk raadiosignaal ei levi hästi läbi metsa, siis raadiosignaali abil leida sondi üles maandumiskohast on aeganõudev. Kuid kuna teatud kõrgusel kaob ära mobiilne võrk, siis oli võimalik sellist meetodit kasutades jälgida sondi lennu alguses ja lennu lõpus, kui sond oli maapinna lähedal.

2.5 Katse läbiviimine

Lennu start oli planeeritud kell 11:00 10. veebruaril 2019. Libedad teeolud külavaheteedel pikendasid stardikohale jõudmise aega, lükates starti edasi. Varblasse jõudes otsiti sobiv koht, kus oli stardi jaoks vajalik vaba ala ja lage ala ida suunas, et sondi oleks võimalik lennates kaua jälgida, kuna tugev tuul puhus läänest. Õhupalli täideti balloonis olevast heeliumiga. Balloonis oli 4000 l heeliumi. Soovitud heeliumikogus mõõdeti balloonit küljes oleva rõhumõõdikuga. Lennus kasutati 600 g massiga lateksist õhupalli. Õhupalli täitmiseks võeti plastmassist pastaka toru korpus ja selle ümber mässiti tihedalt õhupalli suu. Kinnituseks kasutati nipukaid. Pastaka teise otsa ühendati voolik, mis oli ühendatud ballooniga. Pastakat kasutati, et oleks võimalik teha võimalikult tihe ühendus vooliku ja õhupalli vahel ilma, et sulguks heeliumi liikumine balloonist õhupalli. Peale õhupalli täitmist volditi voolik õhupalli lähedalt mitmekordsest kokku ja kinnitati see nipukatega. Seejärel lõigati voolik läbi. Pika nööriga kinnitati sond õhupalli suu külge. Kokku pandi õhupalli umbes 2800 l heeliumi. Tugeva tuule tõttu pidi õhupalli väga maa lähedal

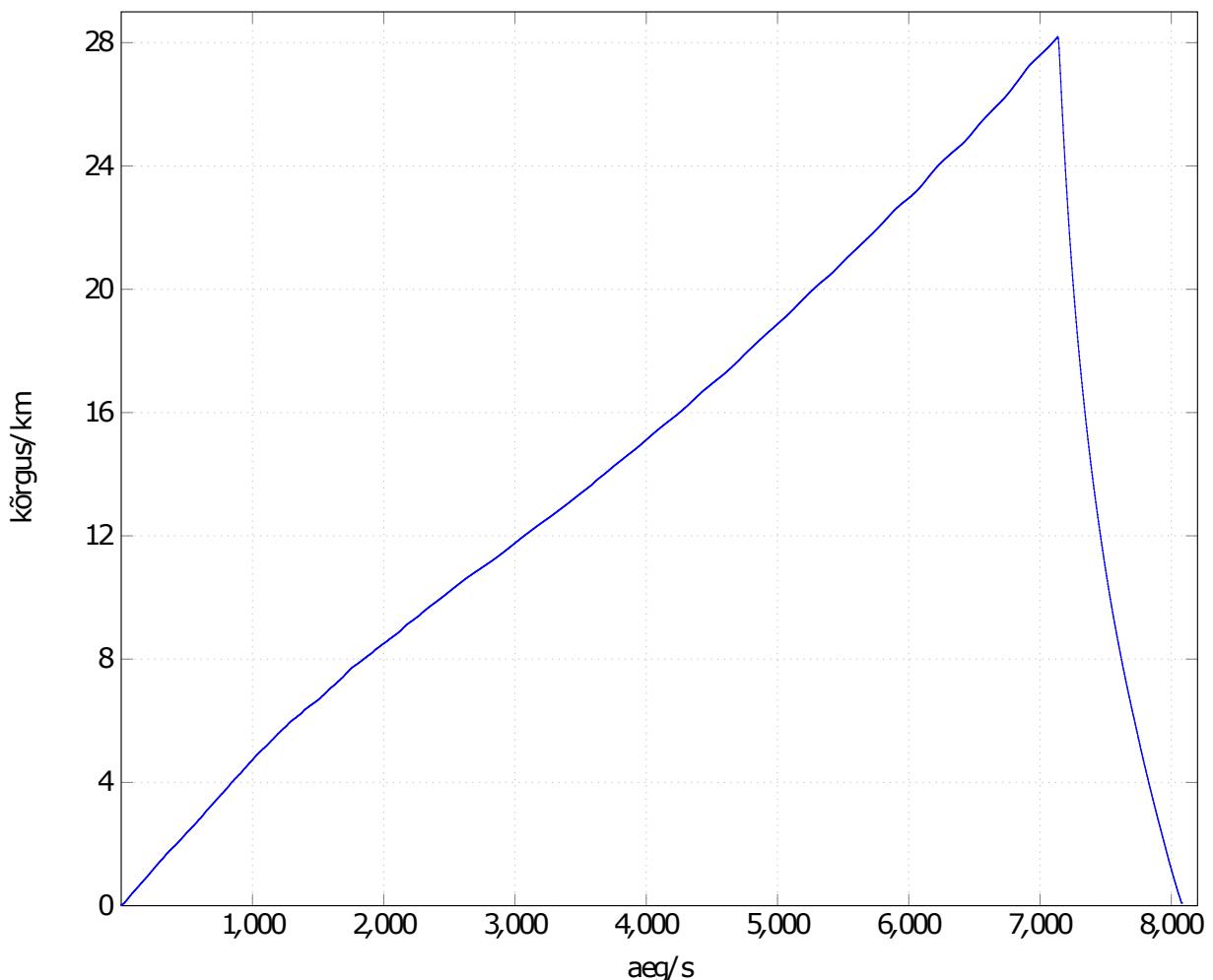
täitma ja lisaks kätega õhupalli kinni hoidma. Lisaks enne lõplikku vooliku läbilõikamist kontrolliti, kas õhupall suudab sondi õhku tõsta ja hinnati tõstejõu piisavust.

Vahetult enne lendu helistati Lennuametisse ja küsiti viimast kinnitust lennuks. Lennu start toimus kell 11:51. Sond kadus pilvise ilma tõttu mõne minutiga vaateväljast. Raadiosidet suudeti hoida umbes 20 min. Peale seda polnud võimalik puhest signaali kätte saada. Siis kadus GPS jälgija ühendus mobiiliside me teenuspakkujaga kõrguse tõttu. Peale sideühenduse kadumist hakati liikuma ennustatava maandumiskoha poole Paidesse. Peale maandumist ühendas GPS jälgija ennast uuesti teenusepakkuja võrku ja saadi teada sondi kukkumise asukoht. Sond maandus kell 14:06 Paide lähedal paarkümmend meetrit Tallinn-Tartu maanteest. Sondi maandumisest saadi teada umbes 15 min peale seda, kui kontrolliti sondi asukohta GPS jälgija kaudu.

3 Katseandmete analüüs

Andmete analüüsimisks kasutati autori poolt kirjutatud programmi. Programm aitas suurest andmekogust välja sorteerida vajalikud andmed ja kontrollida katseandmete kokkuluangemist teoreetiliste seostega.

3.1 Lennu asukohaline ülevaade



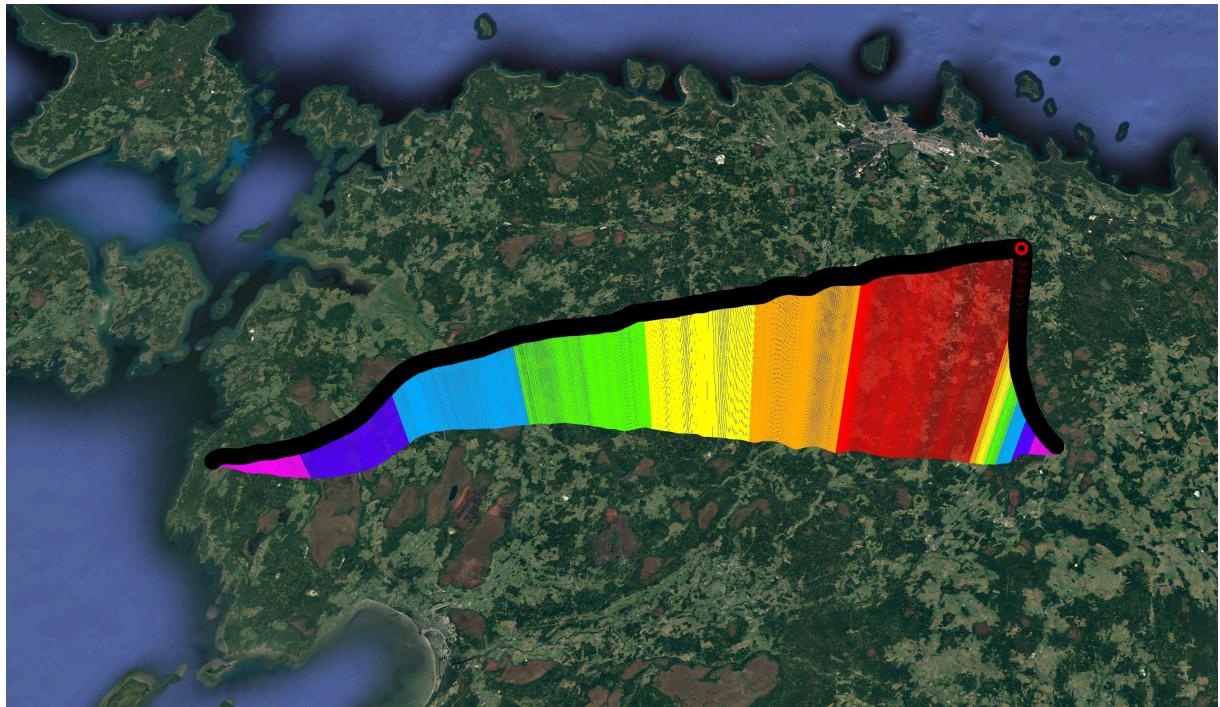
Joonis 4. kõrguse sõltuvus ajast

Joonisel 4 on näha sondi kõrguse muutumist ajas. Kokku kestis lend 8083 sekundit, ehk 2 tundi, 14 min ja 43 sekundit. Selle aja jooksul tehti kokku 2460 mõõtmist. Mõõtmised

on tehtud sekundilise täpsusega iga 3 või 4 sekundi tagant. Keskmiselt tehti mõõtmisi iga 3.29 s tagant.

Tõusmisel oli sondil ühtlane tõusukiirus. Keskmine kiirus tõustes oli 3.95 m/s. Laskudes kiirus varieerus. Peale kukkumise algust langes sond kiiresti väikese õhutiheduse tõttu. Keskmine kiirus peale langemise algust esimesel 4 km oli 78 m/s. Keskmine kiirus vahetult enne kokkupõrget maaga oli 14 m/s. Kogu kukkumise keskmine kiirus oli 29.8 m/s.

Joonisel 5 on näha sondi liikumise trajektori. Värvidega on näidatud sondi kõrgus. Roosa on kõrgusel 0 km kuni 4 km, lilla on kõrgusel 4 km kuni 8 km, sinine on kõrgusel 8 km kuni 12 km, roheline on kõrgusel 12 km kuni 16 km, kollane on kõrgusel 16 km kuni 20 km, oranž on kõrgusel 20 km kuni 24 km ja punane on kõrgusel 24 km kuni suurima kõrguseni, milleks oli 28.188 km. Tõusmisel läbis sond suure horisontaalse nihke, milleks oli 109.86 km. See tähendab, et andmed pole tõustes kogutud ühe vertikaalse joone peal, vaid üpris laiaala peal. Allakukkumisel läbis sond horisontaalselt ainult 14.5 km ja seda palju väiksema ajaga. Seega kukkumisel mõõdetud andmed on tehtud väikese ajaga umbes sama vertikaalse joone peal.

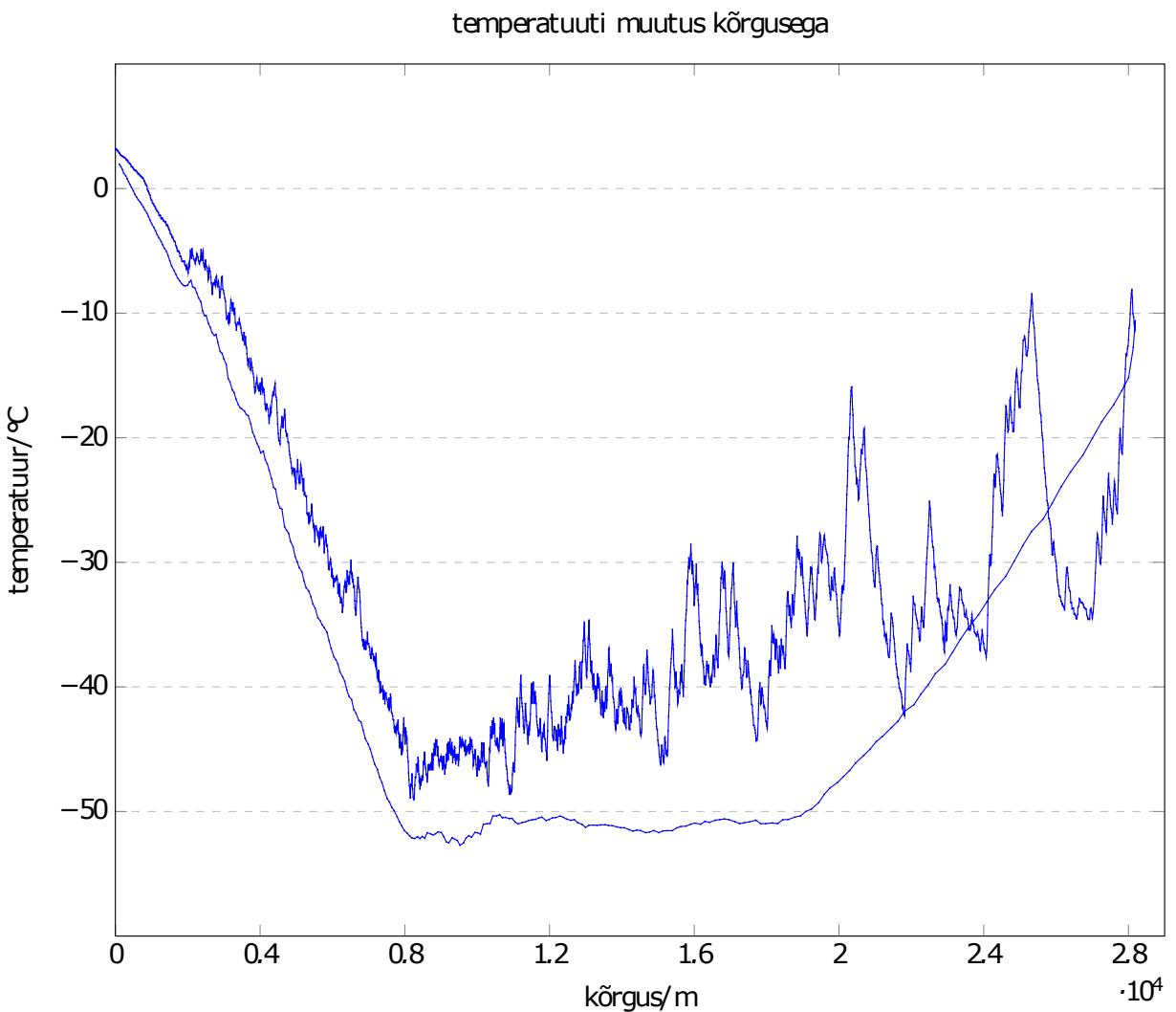


Joonis 5. Sondi trajektoor

Allikas: Autori erakogu ja Google Earth

3.2 Temperatuuri muutus kõrgusega

Joonisel 6 on näha temperatuuri näit kõrguse muutumisel. Joonisel on kaks joont, sest andmeid mõõdeti igalt kõrguselt kaks korda, sondi tõusmisel ja sondi laskumisel. Lennu stardi ajal oli kõrgem temperatuur, kui lennu lõppemise ajal.



Joonis 6. Temperatuuri sõltuvus kõrgusest

Joonisel on näha, kuidas temperatuur kõigub, mitte ei muudu ühtlaselt. See on tingitud päikesekiirgusest. Kui sensor on Päikese poole, siis soojendab Päike sensorit. Kui sensor on sondi varjus, siis peale mahajahtumist mõõdab sensor jälle tegeliku õhutemperatuuri. Sensor ei mõõda kunagi madalamat temperatuuri, kui tegelik õhu temperatuur. Kõrguse kasvades muutub temperatuuri kõikumise amplituud suuremaks. See on tingitud madalast rõhust. Kui õhk hõreneb, siis hakkab sondi temperatuuri rohkem mõjutama Päike, kui õhk ise.

Kukkumise alguses on sensor pikalt olnud Päikese poole ja soojenened. Seega sensor kukumise ajal jahtub, kuid kuna nende kõrguste juures kõrguse vähenemisel langeb ka temperatuur, ei saavuta sensor välistemperatuuri varem kui 20 km kõrgusel maast. Sel ajal on näha sujuvat temperatuuri muutust. Sensori mitte ülessoojenemist kukkumisel võib põhjendada mitut moodi. Sond võis kukkumisel hakata tugevalt pöörlema, mille tõttu polnud sensoril aega üles soojeneda. Kogu lennu vältel liikus sond külgtuultega samal kiiruse sel ja sama suunaga. Seega sondile mõjusid tuuled, mis tulenevad üles liikumisest ja allakukkumisest. Kuna kuni 20 km'ni kukkus sond keskmise kiirusega 70 m/s, siis jahutas tuul sensorit.

Kõrgemal kui 8 km hakkab temperatuur kõrguse kasvades suurenema. See on tingitud osoonikihis neelduvast UV kiirgusest. Kuna järgnevalt uuritakse, kas temperatuur muutub adiabaatilise protsessi käigus, siis vaadeldakse ainult umbes esimest 8 km. Täpseks kõrguseks valiti 8154 m. Sellel kõrgusel tehti viimane mõõtmine, mis oli temperatuurigraafiku viimane lokaalne miinimum, peale mida hakkas temperatuur jälle tõusma. Kuna graafik on ebatasane, vastab lokaalne miinimum kõige paremini tegelikule temperatuurile. Sama kõrgus valiti ka kukkumisel.

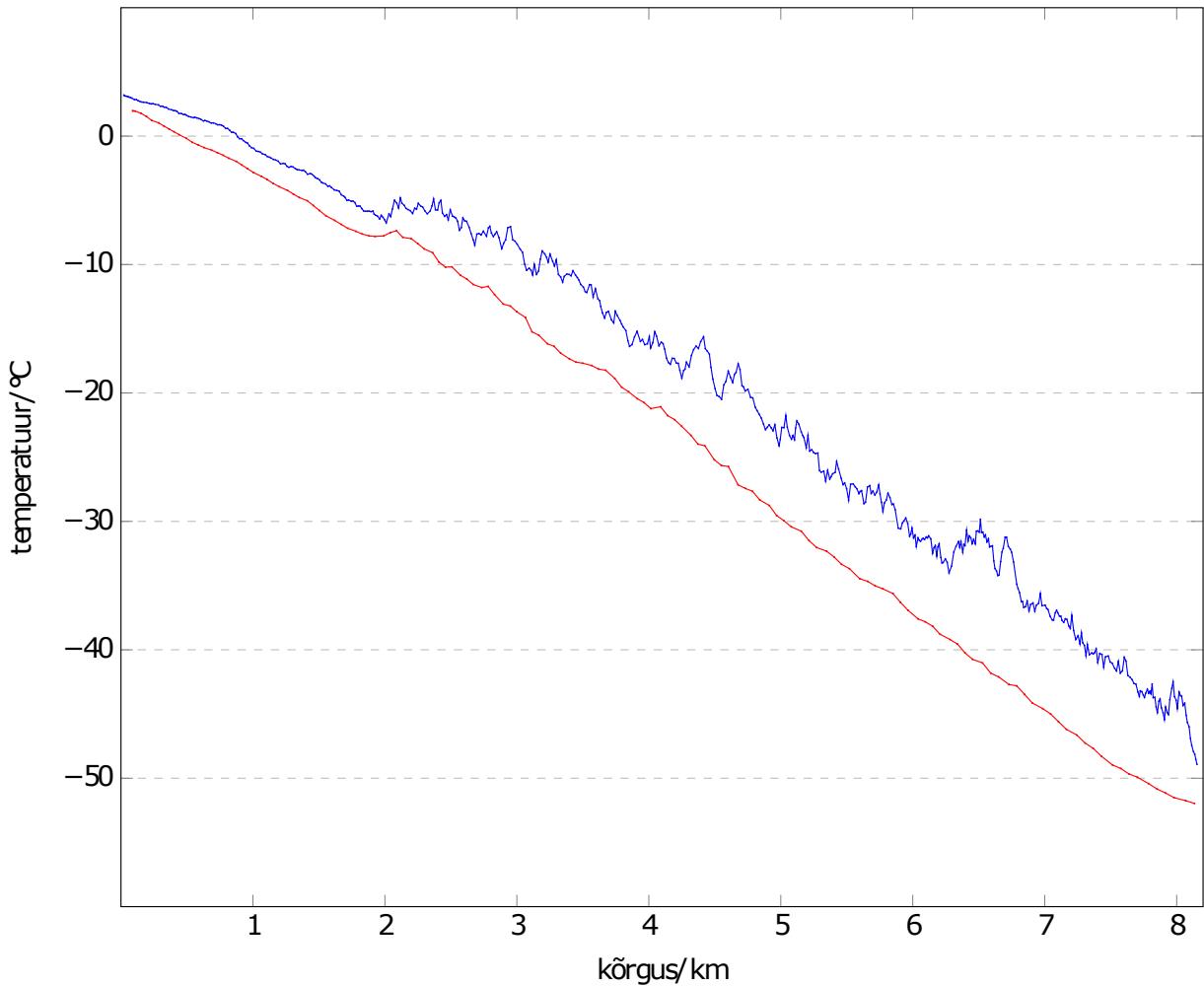
Joonisel 7 on anomalia. Umbes 2 km kõrgusel on nii laskumisel kui ka tõusmisel näha kõrguse suurenemisel väikest temperatuuri muutust. Kuna see toimub nii tõusmisel kui ka kukkumisel, siis on see suure tõenäosusega tegelik temperatuuri muutus ja mitte sensori viga või muud sellist.

Vaadates joonist 8 on näha, et kuni 2 km kõrguseni on õhuniiskus ühtlaselt kõrge. Kuid peale 2 km on näha, et õhuniiskus langeb tugevalt. Õhuniiskus langes, kuna sond väljas pilvedest ning temperatuur tõusis. Pilvedest väljumist saab ka tõestada temperatuuri kõikumise algusega. Esimesed 2 km temperatuur ei kõikunud, kuna Päikest ei paistnud sondile peale. Pilvedest väljudes hakkas aga Päike mõjutama sensori lugemit. Kuna osa mõõtmisi tehti pilvede sees ja osa pilvedest väljas, otsustati vaadelda temperatuuri muutumist eraldi pilvedest madalamal ja pilvedest kõrgemal.

Järgnevalt leitakse temperatuuri muutusele kõrgusega parim lineaarne seos:

$$T(z) = T_0 + \Gamma z.$$

kus T_0 on temperatuur algpunktis ja Γ on temperatuurigradienit, ehk temperatuuri muutus



Joonis 7. Temperatuuri sõltuvus merepinnast kuni kõrguseeni 8 km

Allikas: Autori erakogu

kõrguse kasvades. Lineaarse seose leidmiseks kasutati vähimruutude meetodit, mis avaldub järgnevalt:

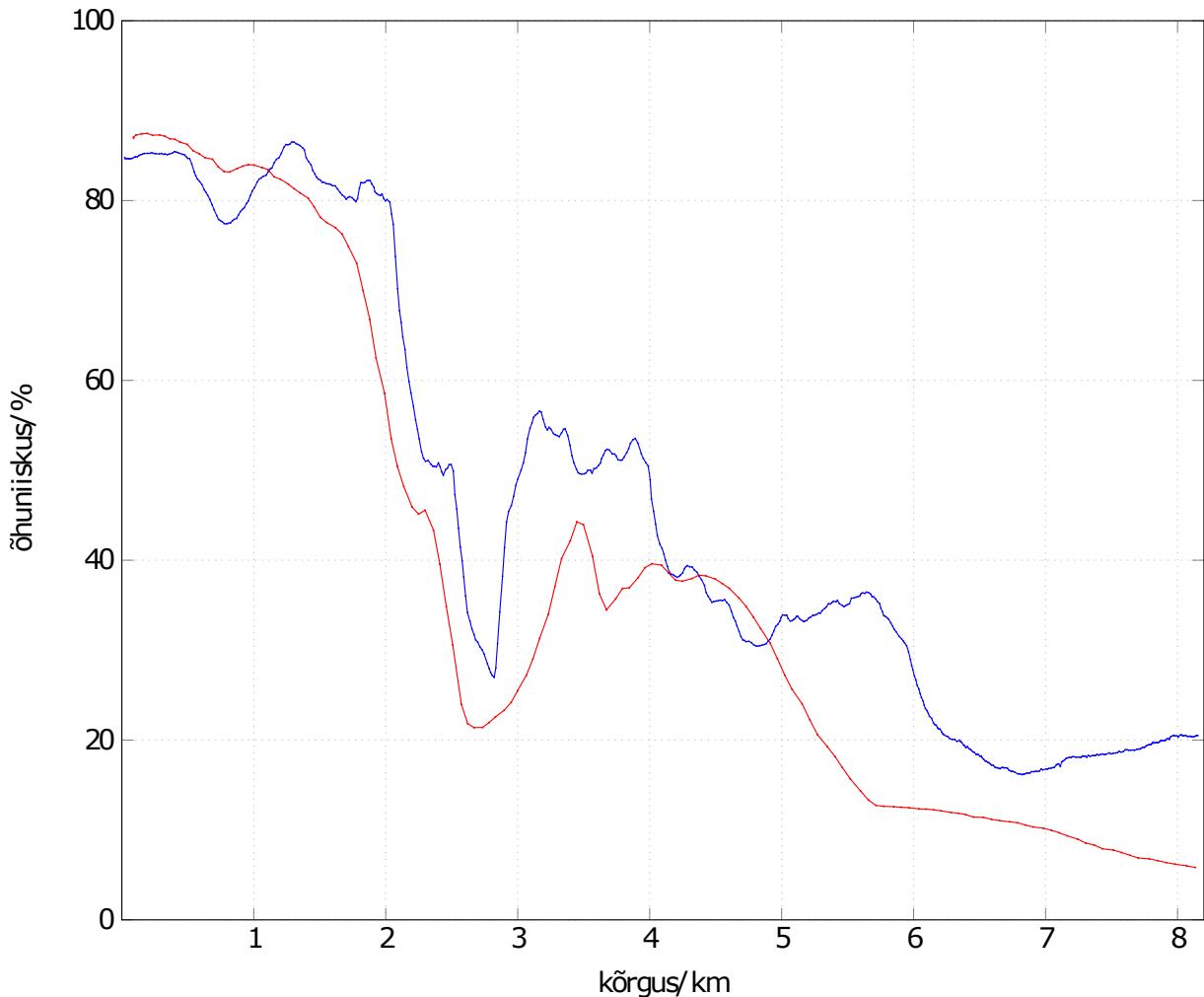
$$\Gamma = \frac{\sum_{i=1}^n (z_i - \bar{z})(T_i - \bar{T})}{\sum_{i=1}^n (z_i - \bar{z})^2}$$

$$T_0 = \bar{T} - \Gamma \bar{z}$$

kus z on kõrgus ja T on temperatuur (Kenney, Keeping 1962: 252-285).

Pilvede sees vaadati temperatuure õhupalli tõustes kuni 2010 m meetrini. Sellel kõrgusel tehti viimane mõõtmine, peale mida temperatuur tõusis pilvedest väljumise tagajärel. Samal põhjusel valiti laskumisel viimaseks andmepunktiks 1926 m kõrgusel mõõdetud andmepunkt.

Joonisel 9 on näha temperatuuri muutust selles vahemikus. Laskumisel on temperatuurigradiant $\Gamma = -5.530\,09\text{ }^{\circ}\text{C}/\text{km}$ ning temperatuur algpunktis on $T_0 = 2.126\,11\text{ }^{\circ}\text{C}$. Tõusmisel



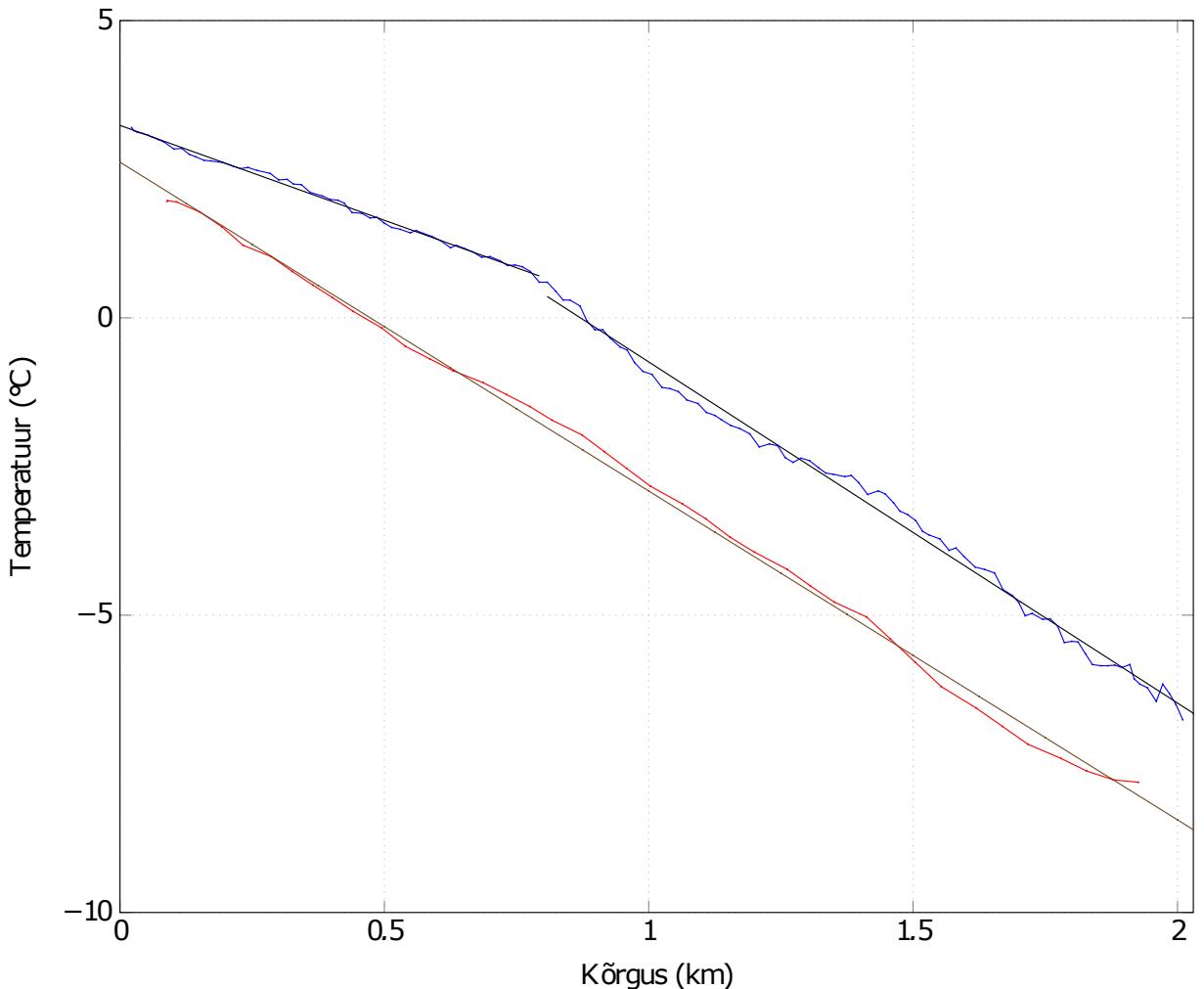
Joonis 8. Õhuniiskuse sõltuvus kõrgusest merepinnast kuni kõrguseni 8 km

Allikas: Autori erakogu

merepinnast kuni kõrguseni 2 km on näha kahte erinevat lineaarset temperatuuri muutust. Merepinnast kuni kõrguseni 793 m on temperatuurigradiant $\Gamma = -3.192\ 33\text{ }^{\circ}\text{C}/\text{km}$ ja temperatuur algpunktis $T_0 = 3.167\ 28\text{ }^{\circ}\text{C}$. Kõrguste vahemikus 808 m kuni 2010 m on temperatuuri gradient $\Gamma = -5.736\ 45\text{ }^{\circ}\text{C}/\text{km}$ ja temperatuur algpunktis $T_0 = 0.360\ 244\text{ }^{\circ}\text{C}$.

Tõusmisel valiti algpunktiks 2565 m kõrgusel mõõdetud andmepunkt. See on esimene andmepunkt, alates 2010 m kõrgusel asuvast andmepunktist, kus on madalam temperatuur, kui 2010 m kõrgusel mõõdetud temperatuur. Laskumisel valiti algpunktiks 2136 m kõrgusel mõõdetud andmepunkt. See on esimene andmepunkt, alates 1926 m kõrgusel mõõdetud andmepunktist, kus on madalam temperatuur kui 1926 m kõrgusel mõõdetud temperatuur. Antud vahemikus olevad mõõtmised on kuvatud joonisel 10.

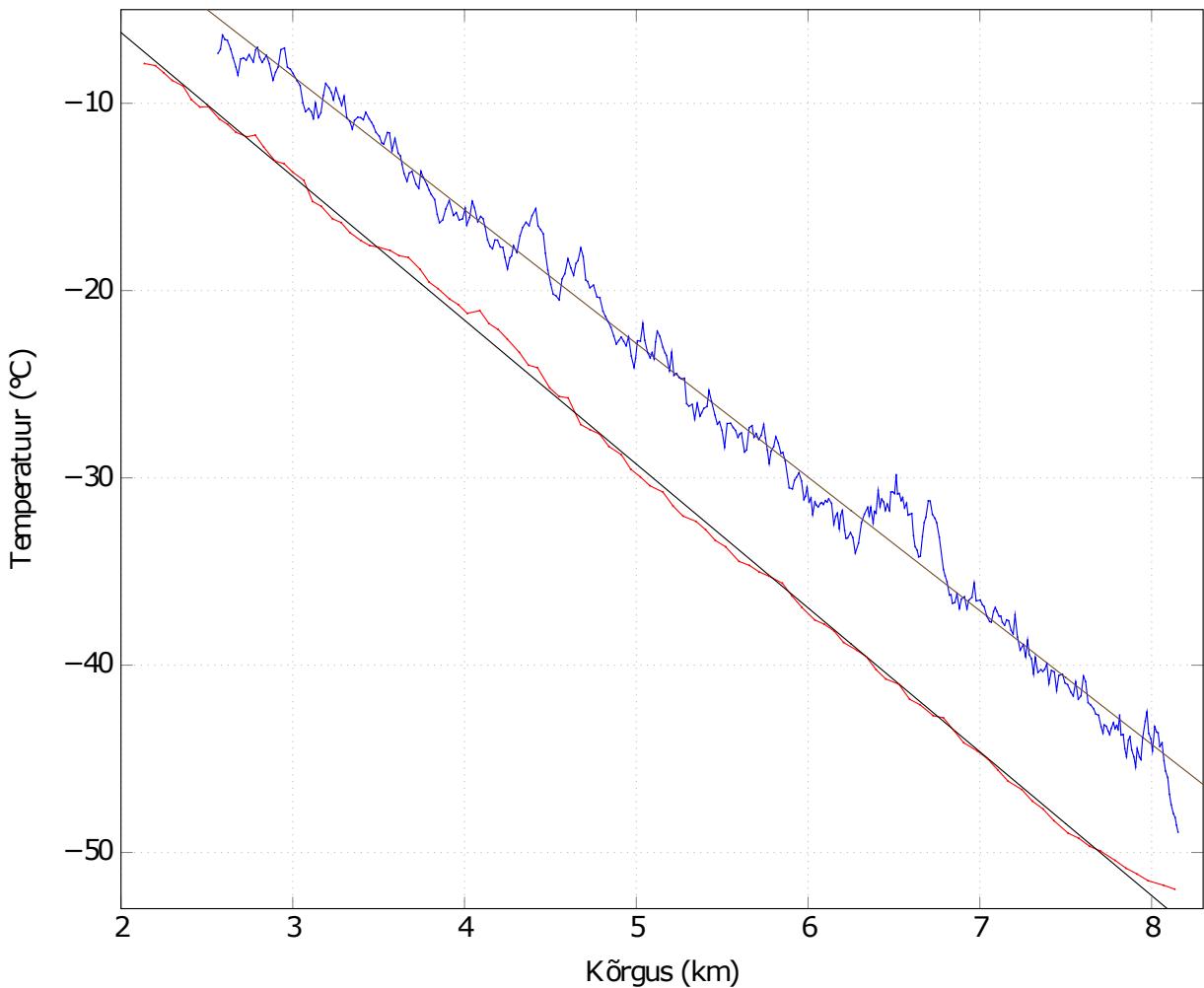
Tõusmisel on temperatuurigradiant $\Gamma = -7.133\ 99\text{ }^{\circ}\text{C}/\text{km}$ ja temperatuur algpunktis $T_0 =$



Joonis 9. Temperatuuri sõltuvus kõrgusest merepinnast kuni kõrguseni 2 km
Allikas: Autori erakogu

-5.4533°C . Laskumisel oli temperatuurigradiant $\Gamma = -7.679\,92^{\circ}\text{C}/\text{km}$ ja temperatuur algpunktis $T_0 = -7.266\,35^{\circ}\text{C}$.

Tõusmise graafik on kõikuv, mille on põhjustanud päikesekiirgus. Kuna Päike soojendas, siis sensor mõõtis tegelikust kõrgemat temperatuuri. Kui sensor jahtus, siis mõõtis sensor tegelikku temperatuuri. Kasutades asjaolu, et sensor ei mõõtnud tegelikust madalamat temperatuuri, võib eemaldada kõik kõrvalekalded. Andmeid hakati madalamast kõrgusest vaatama nii, et temperatuur pidevalt langeks. Kui kõrguse suurenedes temperatuur tõuseb, eemaldati järjest kõik andmepunktid, kuni jõuti andmepunktini, mis oli madalam viimasest võrdluspunktist. Tulemus on joonisel 11. Temperatuurigradiant on $\Gamma = -7.231\,93^{\circ}\text{C}/\text{km}$ ja temperatuur algpunktis $T_0 = -5.961^{\circ}\text{C}$.



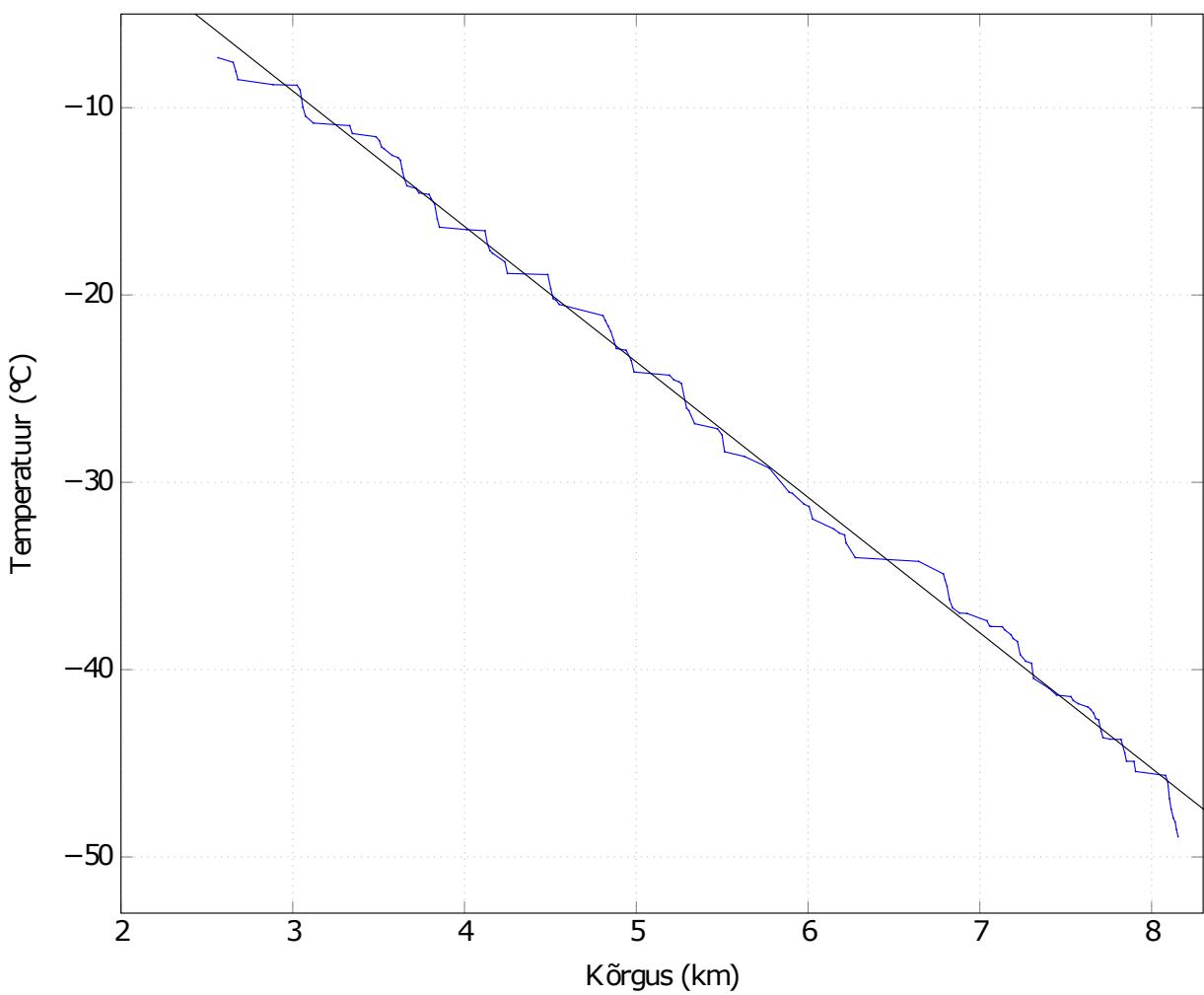
Joonis 10. Temperatuuri sõltuvus kõrgusest üle 2 km

Allikas: Autori erakogu

Tabelis 1 on kokkuvõttetabel selles osas mõõdetud andmetest. Kui kasutada valemit 14, saab temperatuurigradiendi kasutades leida veeauru massiosakaalu õhus, saades valemisks

$$\omega = \frac{g + c_o}{\Gamma(c_o - c_v)}.$$

Järgnevalt arvutatakse välja veeauru massiosakaal erinevatel kõrgusvahemikel. Kõrgusel 22 m kuni 793 m moodustab veeaur 64.9 % õhu massist. Kõrgusel 808 m kuni 2010 m moodustab veeaur 22.2 % õhu massist. Kõrgusel 2565 m kuni 8154 m moodustab veeaur 11.1 % õhu massist. Kõrgusel 89 m kuni 1926 m moodustab veeaur 24.2 % õhu massist. Kõrgusel 2136 m kuni 8154 m moodustab veeaur 8.6 % õhu massist.



Joonis 11. Temperatuuri sõltuvus kõrgusest üle 2 km

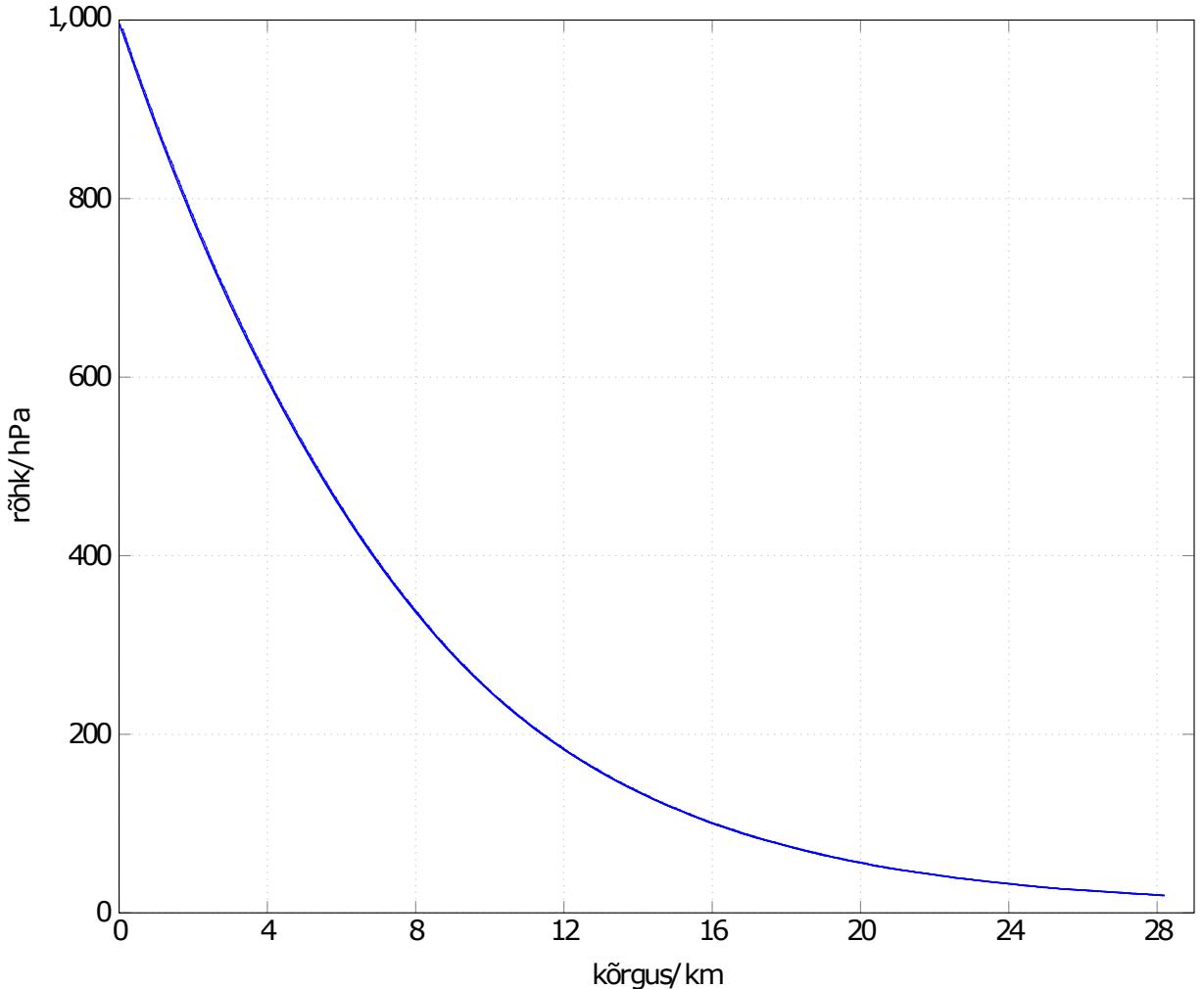
Allikas: Autori erakogu

Tabel 1. Temperatuurigradiendid erinevatel kõrgustel

alguspunkti kõrgus	lõpppunkt kõrgus	tõus/langus	T_0	Γ
22	793	tõus	3.16728	-3.19233
808	2010	tõus	0.360244	-5.73645
2565	8154	tõus	-5.961	-7.23193
89	1926	langus	2.12611	-5.53009
2136	8154	langus	-7.26635	-7.67992

3.3 Rõhu muutus kõrgusega

Joonisel 12 on näha rõhu muutust kogu lennu jooksul. Rõhu mõõtmistulemused on täpsed (kõrvalekanded on suhteliselt väikesed). Jooniselt on seda raske välja lugeda, aga tegelikult eksisteerib kaks joont. Tõusmisel on rõhk madalam kui langemisel.



Joonis 12. Rõhu muutus kõrgusega

Allikas: Autori erakogu

Käesolevas osas uuritakse, et kas teoria osas saadud valem, mis kirjeldab rõhu muutust atmosfääris,

$$p = p_0 \left(1 + \frac{\Gamma z}{T_0}\right)^{-\frac{g\mu}{RT}} \quad (15)$$

vastab katseandmetele. Kuna temperatuuri gradient on erinevatel atmosfääri osades erinev, siis vaatame üksikult osi, kus temperatuuri gradient on konstantne. Konstantite

väärtused on: $g = 9.806\,65 \frac{\text{m}}{\text{s}^2}$, $R = 8.314\,459\,8 \frac{\text{J}}{\text{mol}\cdot\text{K}}$ ja $\mu = 0.028\,964\,7 \frac{\text{km}}{\text{mol}}$. Valemi testimiseks kasutati programmi, mis arvutab algandmete põhjal välja rõhu mõõdetud punkti kõrgusel ja võrdleb sellel kõrgusel mõõdetud rõhuga. Võrdlemise valem on

$$e = \frac{p_a - p_m}{p_a},$$

kus p_a on arvutatud rõhk ja p_m on mõõdetud rõhk.

Esimene kõrgusvahemik on 22 m kuni 793 m. Kõrgusel 22 m on $p_0 = 995.142\,42 \text{ hPa}$ ja $T_0 = 276.31728 \text{ K}$. Keskmne erinevus selles kõrgusvahemikus oli $0.515\,915\%$. Kõrgusel 793 m mõõdeti rõhuks 903.92 hPa ja arvutati rõhuks 904.276 hPa . Järgmise kõrgusvahemiku alguspunktis 808 m kõrgusel arvutas programm rõhuks 902.585 hPa .

Teine kõrgusvahemik on 808 m kuni 2010 m. Kõrgusel 808 m on $p_0 = 902.01955$ ja $T_0 = 273.510244 \text{ K}$. Keskmne erinevus selles kõrgusvahemikus oli $0.229\,426\%$. Kõrgusel 2010 m mõõdeti rõhuks 774.026 hPa ja arvutati rõhuks 774.777 hPa . Järgmise kõrgusvahemiku alguspunktis 2565 m kõrgusel arvutas programm rõhuks 721.283 hPa . Kui kasutada eelnevas kõrgusvahemikus arvutatud rõhku 808 m kõrgusel, mis oli 902.585 hPa siis on keskmne erinevus $0.717\,021\%$. Sellisel juhul on järgmise kõrgusvahemiku alguspunktis 2565 m kõrgusel programmi järgi rõhuks 721.735 hPa .

Kolmas kõrgusvahemik on 2565 m kuni 8154 m. Kõrgusel 2565 m on $p_0 = 720.86625$ ja $T_0 = 267.189 \text{ K}$. Keskmne erinevus selles kõrgusvahemikus oli $4.720\,04\%$. Kõrgusel 8154 m mõõdeti rõhuks 329.341 hPa ja arvutati rõhuks 332.165 hPa . Kui kasutada eelnevas kõrgusvahemikus arvutatud rõhku 2565 m kõrgusel, mis oli 721.735 hPa , siis on keskmne erinevus $5.913\,81\%$.

Vaadates langemisel mõõdetud andmeid, on esimene kõrgusvahemik 89 m kuni 1926 m. Kõrgusel 89 m on $p_0 = 989.325\,23 \text{ hPa}$ ja $T_0 = 275.276\,11 \text{ K}$ ning $\Gamma = -5.530\,09 \text{ K/km}$. Keskmne erinevus selles kõrgusvahemikus oli $2.856\,39\%$. Kõrgusel 1926 m mõõdeti rõhuks 786.723 hPa ja arvutati rõhuks 784.252 hPa . Järgmise kõrgusvahemiku alguspunktis 2136 m kõrgusel arvutas programm rõhuks 763.269 hPa .

Teisel kõrgusvahemikul 2136 m kuni 8154 m. Kõrgusel 2136 m on $p_0 = 765.636\,45 \text{ hPa}$ ja $T_0 = 265.883\,65 \text{ K}$ ning $\Gamma = -7.679\,92 \text{ K/km}$. Keskmne erinevus selles kõrgusvahemikus oli $4.456\,47\%$. Kõrgusel 8154 m mõõdeti rõhuks 332.17 hPa ja arvutati rõhuks

328.404 hPa. Kui kasutada eelnevas kõrgusvahemikus arvutatud rõhku 2136 m kõrgusel, mis oli 763.269 hPa, siis on keskmene erinevus 7.552 64 %.

3.4 Mitteadiabaatilised vahemikud

Atmosfääri ei ole täielikult adiabaatiline. Üks ala, kus atmosfääri ei ole adiabaatiline, on pilvedes, kuna pilvedes toimub veeauru kondenseerumine, mille käigus antakse õhule soojusenergiat. Selline protsess toimub umbes 2 km kõrgusel nii tõustes kui ka langedes. Visuaalselt on seda võimalik näha jooniselt 7. Nüüd arvutatakse välja, kui palju kondenseerub vett umbes 1 kg õhu kohta. Selleks kasutatakse teoria osas leitud valemit

$$\frac{m_v}{m_a} = \frac{i}{2} \frac{R}{\mu L} (T_2 - T_0 - \Gamma z).$$

Tõustes on temperatuurigradient vahetult enne pilvi $\Gamma = -5.736\,45 \text{ K/km}$. Temperatuur kõrgusel 2010 m on $T_0 = -6.76^\circ\text{C}$ ja sellest $z = 555 \text{ m}$ kõrgemal kõrgusel 2565 m on $T_2 = -7.33^\circ\text{C}$, ning vee aurustumissoojus on $L = 2\,257\,000 \frac{\text{J}}{\text{kg}_\text{s}}$. Kasutades valemis katseandmeid saadakse, et õhust eraldub veeauru

$$\frac{m_v}{m_a} = 0.83 \frac{\text{g}}{\text{kg}}.$$

Laskudes on temperatuurigradient vahetult enne pilvi $\Gamma = -5.530\,09 \text{ K/km}$. Temperatuur kõrgusel 1926 m on $T_0 = -7.81^\circ\text{C}$ ja sellest $z = 210 \text{ m}$ kõrgemal kõrgusel 2136 m on $T_2 = -7.88^\circ\text{C}$. Kasutades valemis katseandmeid saadakse, et õhust eraldub veeauru

$$\frac{m_v}{m_a} = 0.16 \frac{\text{g}}{\text{kg}}.$$

Ka stratosfääris pole atmosfääri adiabaatiline, sest siseenergiat antakse õhule juurde Päikesest tuleva UV kiirguse neeldumisel. Seda on visuaalselt näha jooniselt 6, kus alates umbes 8 km'ist alates temperatuur kasvab. Nüüd arvutatakse, kui palju oleks erinevus õhu siseenergias, kui UV kiirgust ei neelduks stratosfääris. Selleks kasutatakse teoria osas leitud valemit:

$$\frac{U_2}{U_1} = \frac{T_2}{T_0 + \Gamma z}.$$

Temperatuur kõrgusel 8154 m on $T_0 = 224.21 \text{ K}$ ja sellest $z = 18\,753 \text{ m}$ kõrgemal kõrgusel 26 907 m on $T_2 = 238.6 \text{ K}$. See kõrgus valiti, sest see on viimane lokaalne miinimum õhupalli tõustes ja seega viimane võimalikult täpne õhu temperatuur. Seega, kui päike paistab, on

siseenergia

$$\frac{U_2}{U_1} = 2.69$$

korda suurem, võrreldes olukorraga kui päikest ei paistaks.

Need arvutused on hinnangulised näitamaks, et atmosfääri ei ole adiabaatiline pilvede sees ja stratosfääris.

3.5 Järeldus

Eelnevast järeldub, et atmosfääris toimuvad termodünaamilised protsessid pole kõik adiabaatilised. Pilvedes veeaur kondenseerub ning sellisel juhul pole termodünaamiline protsess adiabaatiline ja temperatuurigradienit, mis tuletati adiabaatilise protsessi jaoks, ei ühti. Teiseks, kui päike paistab, siis osoonikihis neeldunud kiurguse tõttu pole termodünaamiline protsess adiabaatiline ja temperatuurigradienit, mis tuletati adiabaatilise protsessi jaoks, ei ühti katseandmatega.

Pilvedest madalamal ja pilvedest kõrgemal tropsfääris, kus pole osoonikihi muutub temperatuur kõrguse muutumisel vastavalt temperatuurigradiendile, mis tuletati adiabaatilise protsessi jaoks. Temperatuurigradienit sõltub õhus olevast veeaurust. Kuna veeaurul on suurem erisoojus, siis mida rohkem on veeauru õhus, seda vähem muutub temperatuur kõrguse muutumisel. Seda kinnitasid ka katseandmed, et pilvedest madalamal, kus on rohkem veeauru, on temperatuurigradiendi absoluutväärus väiksem, kui pilvedest väljas, kus on vähem veeauru.

Eriti täpseks osutus rõhu muutuse arvutamise valem 15, mis tuletati eeldusel, et atmosfääris toimuvad termodünaamilised protsessid on adiabaatilised. Seega, kui on teada temperatuurigradienit ja algandmed algpunktis, on võimalik täpselt välja arvutada rõhk soovitud punktis.

Kokkuvõte

Uurimistöö hüpotees, et atmosfääri madalamates kihtides toimuvad termodünaamilised protsessid on adiabaatilised protsessid, leidis uurimistöö tulemusena kinnitust, kuid ilmnesid järgmised erandid: adiabaatilisus ei kehti pilvedes ja stratosfääris.

Adiabaatiline protsess on termodünaamiline protsess mille käigus soojusvahetust ei toimu. Stratosfääris toimuvad termodünaamilised protsessid ei ole adiabaatilised, sest Päikese poolt kiiratud UV kiirgus neeldub stratosfääris asuvas osoonikihis, eraldades soojust, mille käigus soojeneb õhk. Pilvedes toimuvad termodünaamilised protsessid ei ole adiabaatilised, sest õhus oleva veeauru kondenseerumisel eralduv soojus läheb õhu soojendamiseks.

Kuna aga troposfääris soojendab õhku ainult maa pind ja õhk on seal pidevas liikumises on troposfääris toimuvad termodünaamilised protsessid adiabaatilised.

Teoreetiliselt leiti ja katseandmetega kinnitati, et temperatuur muutub kõrguse kasvades lineaarselt ning temperatuurigradienit sõltub veeauru kogusest õhus. Mida rohkem on õhus veeauru, seda vähem muutub temperatuur kõrguse muutumisel. Teoreetiliselt leiti ka valem, mis kirjeldab rõhu muutumist kõrgusega kui on teada temperatuurigradienit ja algpunktis mõõdetud algandmed. Katseandmetega kontrollides leiti, et valem kirjeldab rõhu muutust väga täpselt. Sellega sai täidetud uurimistöö eesmärk.

Kasutatud materjalid

Boeker, E., Grondelle, R. van. (2011) Environmental Physics: Sustainable Energy and Climate Change. Wiley

Fitzpatrick, R. (2006) The adiabatic atmosphere. Loetud: <http://farside.ph.utexas.edu/teaching/sm1/lectures/node56.html>, 05.01.2019

Kalda, J. (2014) „Termodynäamika“

Kenney, J. F., Keeping, E. S. (1962) Mathematics of statistics. D. van Nostrand company inc.

Khan, S. (2016) What is the ideal gas law? Loetud: <https://www.khanacademy.org/science/physics/thermodynamics/temp-kinetic-theory-ideal-gas-law/a/what-is-the-ideal-gas-law>, 03.03.2019

Marshall, J., Plumb, R. A. (1959) Atmosphere, Ocean and Climate Dynamics: An Introductory Text. Elsevier, Academic Press

Wallace, J. M., Hobbs, P. V. (2006) Atmospheric Science, Second Edition: An Introductory Survey. Academic Press

Lisa 1 Andmete kogumise kood

```
from pytrack.rtty import *
from pytrack.lora import *
from pytrack.led import *
from pytrack.temperature import *
from pytrack.cgps import *
from pytrack.camera import *
from pytrack.telemetry import *

from time import sleep
import threading
import configparser

import bme280
import RPi.GPIO as GPIO

class Tracker(object):
    # HAB Radio/GPS Tracker

    def __init__(self):
        """
        This library uses the other modules (CGPS, RTTY etc.) to build a complete
        tracker.
        """
        self.camera = None
        self.lora = None
        self.rtty = None
        self.SentenceCallback = None
        self.ImageCallback = None
        self.andmed = []

    def _TransmitIfFree(self, Channel, PayloadID, ChannelName, ImagePacketsPerSentence):
        if not Channel.is_sending():
            # Do we need to send an image packet or sentence ?
            # print("ImagePacketCount = ", Channel.ImagePacketCount,
            #      ImagePacketsPerSentence)
            if (Channel.ImagePacketCount < ImagePacketsPerSentence) and self.
                camera:
                Packet = self.camera.get_next_ssdv_packet(ChannelName)
            else:
                Packet = None

            if Packet == None:
                print("Sending telemetry sentence for " + PayloadID)
```

```

    Channel.ImagePacketCount = 0

    # Get temperature
    InternalTemperature = self.temperature.Temperatures[0]

    # Get GPS position
    position = self.gps.position()

    # Build sentence
    Channel.SentenceCount += 1

    fieldlist = [PayloadID,
                 Channel.SentenceCount,
                 position.time,
                 "{:.5f}".format(position.lat),
                 "{:.5f}".format(position.lon),
                 int(position.alt),
                 position.sats,
                 "{:.2f}".format(InternalTemperature)]

    self.andmed = fieldlist

    if self.SentenceCallback:
        fieldlist.append(self.SentenceCallback())

    sentence = build_sentence(fieldlist)
    print(sentence, end="")

    f = open("/home/pi/lend/log.txt", "a")
    f.write(sentence)

    # Send sentence
    Channel.send_text(sentence)
else:
    Channel.ImagePacketCount += 1
    print("Sending SSDV packet for " + PayloadID)
    Channel.send_packet(Packet[1:])

def set_rtty(self, payload_id='CHANGEME', frequency=434.200, baud_rate=50,
            image_packet_ratio=4):
    """
    This sets the RTTY payload ID, radio frequency, baud rate (use 50 for
    telemetry only, 300 (faster) if you want to include image data), and ratio
    of image packets to telemetry packets.

    If you don't want RTTY transmissions, just don't call this function.
    """
    self.RTTPayloadID = payload_id
    self.RTTFrequency = frequency

```

```

        self.RTTYBaudRate = baud_rate
        self.RTTYImagePacketsPerSentence = image_packet_ratio

        self.rtty = RTTY(self.RTTYFrequency, self.RTTYBaudRate)

    def set_lora(self, payload_id='CHANGEME', channel=0, frequency=424.250, mode=1, DIO0=0, camera=False, image_packet_ratio=6):
        """
        This sets the LoRa payload ID, radio frequency, mode (use 0 for telemetry-only; 1 (which is faster) if you want to include images), and ratio of image packets to telemetry packets.

        If you don't want LoRa transmissions, just don't call this function.

        Note that the LoRa stream will only include image packets if you add a camera schedule (see add_rtty_camera_schedule)
        """
        self.LoRaPayloadID = payload_id
        self.LoRaChannel = channel
        self.LoRaFrequency = frequency
        self.LoRaMode = mode
        self.LORAImagePacketsPerSentence = image_packet_ratio

        self.lora = LoRa(Channel=self.LoRaChannel, Frequency=self.LoRaFrequency, Mode=self.LoRaMode, DIO0=DIO0)

    def add_rtty_camera_schedule(self, path='images/RTTY', period=60, width=320, height=240):
        """
        Adds an RTTY camera schedule. The default parameters are for an image of size 320x240 pixels every 60 seconds and the resulting file saved in the images/RTTY folder.
        """
        if not self.camera:
            self.camera = SSDVCamera()
        if self.RTTYBaudRate >= 300:
            print("EnableucamerauforuRTTY")
            self.camera.add_schedule('RTTY', self.RTTPayloadID, path, period, width, height)
        else:
            print("RTTYucamerauscheduleunotuaddedubaudurateutooulowu(300uminimumuneeded")"

    def add_lora_camera_schedule(self, path='images/LORA', period=60, width=640, height=480):
        """
        Adds a LoRa camera schedule. The default parameters are for an image of size 640x480 pixels every 60 seconds and the resulting file saved in the images/LORA folder.
        """
        if not self.camera:

```

```

        self.camera = SSDVCamera()
    if self.LoRaMode == 1:
        print("Enable camera for LoRa")
        self.camera.add_schedule('LoRa0', self.LoRaPayloadID, path, period,
                               width, height)
    else:
        print("LoRa camera schedule not added to LoRa mode needs to be set to 1
              not 0")

    def add_full_camera_schedule(self, path='images/FULL', period=60, width=0, height=0):
        """
        Adds a camera schedule for full-sized images. The default parameters are for
        an image of full sensor resolution, every 60 seconds and the resulting
        file saved in the images/FULL folder.
        """
        if not self.camera:
            self.camera = SSDVCamera()
        self.camera.add_schedule('FULL', '', path, period, width, height)

    def set_sentence_callback(self, callback):
        """
        This specifies a function to be called whenever a telemetry sentence is built.
        That function should return a string containing a comma-separated list
        of fields to append to the telemetry sentence.
        """
        self.SentenceCallback = callback

    def set_image_callback(self, callback):
        """
        The callback function is called whenever an image is required. **If you
        specify this callback, then it's up to you to provide code to take the
        photograph (see tracker.md for an example)**.
        """
        self.ImageCallback = callback

    def __ImageCallback(self, filename, width, height):
        self.ImageCallback(filename, width, height, self.gps)

    def __transmit_thread(self):
        while True:
            if self.rtty:
                self._TransmitIfFree(self.rtty, self.RTTPayloadID, 'RTTY',
                                     self.RTTPacketsPerSentence)
            if self.lora:
                self._TransmitIfFree(self.lora, self.LoRaPayloadID, 'LoRa0',
                                     self.LORAPacketsPerSentence)
            sleep(0.01)

    def start(self):

```

```

"""
Starts the tracker.
"""

LEDs = PITS_LED()

self.temperature = Temperature()
self.temperature.run()

self.gps = GPS(when_lock_changed=LEDs.gps_lock_status)

if self.camera:
    if self.ImageCallback:
        self.camera.take_photos(self.__ImageCallback)
    else:
        self.camera.take_photos(None)

t = threading.Thread(target=self.__transmit_thread)
t.daemon = True
t.start()

GPIO.setmode(GPIO.BCM)
GPIO.setup(25, GPIO.OUT)

p = GPIO.PWM(25,50)
p.start(7.5)

def openfan():
    p.ChangeDutyCycle(2.5)

def closefan():
    p.ChangeDutyCycle(7.5)

def extra_telemetry():
    tmp,pre,hum = bme280.readBME280All()
    return "{:.2f},{:.5f},{:.5f}".format(tmp) + ',' + "{:.5f},{:.5f},{:.5f}".format(pre) + ',' + "{:.5f},{:.5f},{:.5f}".format(hum)

MT = Tracker()

MT.set_rtty(payload_id='Reaalkool', frequency=434.250, baud_rate=300)

MT.start()

MT.set_sentence_callback(extra_telemetry)

while True:
    sleep(5)
    if(MT.andmed[5] > 20000):
        openfan()

```

```
else:  
    closefan()
```

Lisa 2 Sensori kood

```
#!/usr/bin/python
#_____
#
# / \ \_ \(_)_/ \_\_\_
# / , \_\_/_\ \ \_ \/_ // /
#/ / \_/_ / / \_/_ / .\_/\_, /
#           / \_ / \_/
#
#           bme280.py
# Read data from a digital pressure sensor.
#
# Official datasheet available from :
# https://www.bosch-sensortec.com/bst/products/all_products/bme280
#
# Author : Matt Hawkins
# Date   : 21/01/2018
#
# https://www.raspberrypi-spy.co.uk/
#
#_____
import smbus
import time
from ctypes import c_short
from ctypes import c_byte
from ctypes import c_ubyte

DEVICE = 0x76 # Default device I2C address

bus = smbus.SMBus(1) # Rev 2 Pi, Pi 2 & Pi 3 uses bus 1
# Rev 1 Pi uses bus 0

def getShort(data, index):
    # return two bytes from data as a signed 16-bit value
    return c_short((data[index+1] << 8) + data[index]).value

def getUShort(data, index):
    # return two bytes from data as an unsigned 16-bit value
    return (data[index+1] << 8) + data[index]

def getChar(data, index):
    # return one byte from data as a signed char
    result = data[index]
```

```

if result > 127:
    result -= 256
return result

def getUChar(data,index):
    # return one byte from data as an unsigned char
    result = data[index] & 0xFF
    return result

def readBME280ID(addr=DEVICE):
    # Chip ID Register Address
    REG_ID      = 0xD0
    (chip_id, chip_version) = bus.read_i2c_block_data(addr, REG_ID, 2)
    return (chip_id, chip_version)

def readBME280All(addr=DEVICE):
    # Register Addresses
    REG_DATA = 0xF7
    REG_CONTROL = 0xF4
    REG_CONFIG = 0xF5

    REG_CONTROL_HUM = 0xF2
    REG_HUM_MSB = 0xFD
    REG_HUM_LSB = 0xFE

    # Oversample setting - page 27
    OVERSAMPLE_TEMP = 2
    OVERSAMPLE_PRES = 2
    MODE = 1

    # Oversample setting for humidity register - page 26
    OVERSAMPLE_HUM = 2
    bus.write_byte_data(addr, REG_CONTROL_HUM, OVERSAMPLE_HUM)

    control = OVERSAMPLE_TEMP<<5 | OVERSAMPLE_PRES<<2 | MODE
    bus.write_byte_data(addr, REG_CONTROL, control)

    # Read blocks of calibration data from EEPROM
    # See Page 22 data sheet
    cal1 = bus.read_i2c_block_data(addr, 0x88, 24)
    cal2 = bus.read_i2c_block_data(addr, 0xA1, 1)
    cal3 = bus.read_i2c_block_data(addr, 0xE1, 7)

    # Convert byte data to word values
    dig_T1 = getUShort(cal1, 0)
    dig_T2 = getShort(cal1, 2)
    dig_T3 = getShort(cal1, 4)

    dig_P1 = getUShort(cal1, 6)

```

```

dig_P2 = getShort(cal1, 8)
dig_P3 = getShort(cal1, 10)
dig_P4 = getShort(cal1, 12)
dig_P5 = getShort(cal1, 14)
dig_P6 = getShort(cal1, 16)
dig_P7 = getShort(cal1, 18)
dig_P8 = getShort(cal1, 20)
dig_P9 = getShort(cal1, 22)

dig_H1 = getUChar(cal2, 0)
dig_H2 = getShort(cal3, 0)
dig_H3 = getUChar(cal3, 2)

dig_H4 = getChar(cal3, 3)
dig_H4 = (dig_H4 << 24) >> 20
dig_H4 = dig_H4 | (getChar(cal3, 4) & 0x0F)

dig_H5 = getChar(cal3, 5)
dig_H5 = (dig_H5 << 24) >> 20
dig_H5 = dig_H5 | (getUChar(cal3, 4) >> 4 & 0x0F)

dig_H6 = getChar(cal3, 6)

# Wait in ms (Datasheet Appendix B: Measurement time and current calculation)
wait_time = 1.25 + (2.3 * OVERSAMPLE_TEMP) + ((2.3 * OVERSAMPLE_PRES) + 0.575) + ((2.3 *
OVERSAMPLE_HUM)+0.575)
time.sleep(wait_time/1000) # Wait the required time

# Read temperature/pressure/humidity
data = bus.read_i2c_block_data(addr, REG_DATA, 8)
pres_raw = (data[0] << 12) | (data[1] << 4) | (data[2] >> 4)
temp_raw = (data[3] << 12) | (data[4] << 4) | (data[5] >> 4)
hum_raw = (data[6] << 8) | data[7]

#Refine temperature
var1 = (((temp_raw>>3)-(dig_T1<<1)))*(dig_T2)) >> 11
var2 = (((((temp_raw>>4) - (dig_T1)) * ((temp_raw>>4) - (dig_T1))) >> 12) * (dig_T3)) >> 14
t_fine = var1+var2
temperature = float(((t_fine * 5) + 128) >> 8);

# Refine pressure and adjust for temperature
var1 = t_fine / 2.0 - 64000.0
var2 = var1 * var1 * dig_P6 / 32768.0
var2 = var2 + var1 * dig_P5 * 2.0
var2 = var2 / 4.0 + dig_P4 * 65536.0
var1 = (dig_P3 * var1 * var1 / 524288.0 + dig_P2 * var1) / 524288.0
var1 = (1.0 + var1 / 32768.0) * dig_P1
if var1 == 0:
    pressure=0

```

```

else:
    pressure = 1048576.0 - pres_raw
    pressure = ((pressure - var2 / 4096.0) * 6250.0) / var1
    var1 = dig_P9 * pressure * pressure / 2147483648.0
    var2 = pressure * dig_P8 / 32768.0
    pressure = pressure + (var1 + var2 + dig_P7) / 16.0

# Refine humidity
humidity = t_fine - 76800.0
humidity = (hum_raw - (dig_H4 * 64.0 + dig_H5 / 16384.0 * humidity)) * (dig_H2 / 65536.0 *
(1.0 + dig_H6 / 67108864.0 * humidity * (1.0 + dig_H3 / 67108864.0 * humidity)))
humidity = humidity * (1.0 - dig_H1 * humidity / 524288.0)
if humidity > 100:
    humidity = 100
elif humidity < 0:
    humidity = 0

return temperature/100.0,pressure/100.0,humidity

def main():

    (chip_id, chip_version) = readBME280ID()
    print("ChipID\u00000000:", chip_id)
    print("Version\u00000000:", chip_version)

    temperature , pressure , humidity = readBME280All()

    print("Temperature\u0000", temperature , "C")
    print("Pressure\u0000", pressure , "hPa")
    print("Humidity\u0000", humidity , "%")

if __name__=="__main__":
    main()

```

Lisa 3 Andmete analüüs kood

```
#include <bits/stdc++.h>
using namespace std;

int line(vector<float> Xa, vector<float> Ya, int n){
    float lug = 0; float nim = 0;
    float X = 0; float Y = 0;
    for(int i = 0; i < n; i++){
        if(true){
            X+=Xa[i]; Y+=Ya[i];
        }
    }
    X = X/n; Y = Y/n;
    for(int i = 0; i < n; i++){
        if(true){
            lug += (Xa[i] - X)*(Ya[i] - Y);
            nim += pow(Xa[i]-X, 2);
        }
    }
    float beta = lug/nim;
    float alpha = Y - beta*X;
    cout << beta << " " << alpha << endl;
    //y = x*beta + alpha
}

int pre(vector<float> Xa, vector<float> Ya, int n){
    float L = -0.00767992; float g = 9.80665; float R = 8.3144598; float mu = 0.0289647;
    float aste = -(mu*g)/(R*L);
    float meanE = 0;
    float P = 763.269; float T = 265.88365; float p = 0; float H = 2136;
    for(int i = 0; i < n; i++){
        p = P*pow( (1+(L/T)*(Xa[i]-H)) ,aste );
        float e = (p-Ya[i])/p;
        meanE += abs(e);
    }
    float er = meanE/n*1000;
}

int main(){
    ifstream fi; ofstream fo;
    fi.open("log.txt"); fo.open("sis.txt");
    string A; int B; string C; float D; float E; float F; int G; float H; float I; float J;
    float K; string L;
```

```

vector<float>nr ; vector<float>time ; vector<float>lat ; vector<float>lon ; vector<float>height ;
vector<float>itemp ; vector<float>otemp ; vector<float>pre ; vector<float>hum ;
float n = 0; int T = 0;
while( fi >> A){
    n++;
    fi >>B; fi >>C; fi >>D; fi >>E; fi >>F; fi >>G; fi >>H; fi >>I ; fi >>J ; fi >>K; fi >>L;
    nr . push_back(B) ;
    int t=36000*C[0]+3600*C[1]+600*C[3]+60*C[4]+10*C[6]+C[7]-1933008;
    if(n == 1){T = t;}
    time . push_back(t-T) ;
    lat . push_back(D) ;
    lon . push_back(E) ;
    height . push_back(F) ;
    itemp . push_back(H) ;
    otemp . push_back(I) ;
    pre . push_back(J) ;
    hum. push_back(K) ;
}
line( height ,otemp ,n);

return 0;
}

```

Lisa 4 Logifail

Kinnitusleht

Kinnitan, et

- koostasin uurimistöö iseseisvalt. Kõigile töös kasutatud teiste autorite töödele ja andmeallikatele on viidatud;
 - olen teadlik, et uurimistööd ei edastata teistele tulu teenimise eesmärgil ega jagata teadlikult plagieerimiseks.
-

kuupäev / nimi / allkiri

Tunnistan uurimistöö kaitsmisvalmiks.

Juhendaja

.....

kuupäev / nimi / allkiri

Resümee

Abstract