TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Veikko Tunnel

KAUGSEIRE SEADE VALVESÜSTEEMI VAHEJAAMALE

bakalaureusetöö

Juhendaja: Olev Märtens
tehnikateaduste doktor

Tallinn 2019

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Veikko Tunnel

19.05.2019

Annotatsioon

Käesoleva bakalaureusetöö eesmärk on luua lihtne ja odav seade, mis võimaldaks jälgida valvesüsteemi keskserverist eemal asuva signaalide vastuvõtupunkti seadmekilbi keskkonnaparameetreid ja toitega varustamist. Võimalikest kõrvalekalletest peab seade teavitama keskserverit. Samuti peab olema võimalik serveril soovi korral pärida seadmekilbis asuvalt kontrollerilt parameetreid.

Töö käigus uuriti SNMP protokolli, SNMP agent-programmi toimimist, sõnumite saatmist ning serveri päringutele vastamist ATmega 8-bitise RISC arhitektuuriga protsessori baasil ehitatud kontrolleril. Keskkonna-ja toiteparameetrite seireks leiti sobivad sensorid, uuriti nende sobitamist kontrolleri sisenditega ning võimalikult täpse ja müravaba mõõtetulemuse saavutamist.

Töö tulemusena on välja töötatud soodne ja lihtsalt valmistatav lahendus seadmekilbi toite- ja keskkonnaparameetrite kaugseireks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 6 peatükki, 29 joonist, 4 tabelit.

Abstract

The Remote Monitoring Device for the Intermediate Station of the Security System

The purpose of this Bachelor's thesis is to create a simple and inexpensive device to monitor the environmental parameters and the power supply of the equipment rack of the signaling point located away from the central server of the surveillance system. The device must notify the central server of possible deviations. It must also be possible for the server to request parameters from the controller on the Intermediate Station.

During the work, the operation of the SNMP protocol, the SNMP agent program, the sending of messages and the response to the server queries were investigated on an ATmega 8-bit RISC-based processor-based controller. Suitable sensors for monitoring environmental and power parameters were found, matched them to controller inputs, and achieved as accurate and low-noise measurement as possible.

As a result of the work, a convenient and easy-to-prepare solution for remote monitoring of the power and environmental parameters of the equipment rack has been developed.

The thesis is in estonian and contains 28 pages of text, 6 chapters, 29 figures, 4 tables.

Lühendite ja mõistete sõnastik

agent	protsess, mis jookseb hallatavas seadmes
SNMP	*Simple Network Management Protocol*
IETF	*Internet Engineering Task Force*
ITU-T	*International Telecommunication Union*
UDP	*User Datagram Protocol*
PEN	*Private Enterprise Number*
IANA	*Internet Assigned Numbers Authority*
ASN.1	andmestruktuuride liideste kirjeldamise standard
TRAP	SNMP agendi sõnum serverile
INT	*Integer*
RISC	Kärbitud käsustikuga arvuti
IDE	integreeritud arenduskeskkond
GNU	täielikult vabavarast koosnev UNIXi-laadne opsüsteem
GNU GPL	GNU Üldine Avalik Litsents
GNU LGPL	GNU vähem üldine avalik litsents
ADM	analoog-digitaalmuundi
PWM	pulsi-laius modulaator
RFI	Raadiosageduslikud häired
Wireshark	Võrguliikluse pakettide uurimise tarkvara

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

Käesoleva töö eesmärgiks on luua lihtne ja odav seade, mis võimaldab saada teavet eemal asuva vastuvõtupunkti seadmekilbi toite- ja keskkonnaparameetrite kohta.

1.1 Jälgimist vajava süsteemi kirjeldus

Turvafirma teostab objektide tehnilise valve teenust. Valvesüsteemid edastavad oma staatuse info ja testi raportid raadiolinkide abil. Suurem osa signaale on ULL sagedusalas 146 – 154 MHz. 4W võimsusega saatja (SATELCODE i8) [Joonis 22] töökindel saateraadius on linna tingimustes, kus sageli signaali otsetee puudub, umbes 5 km ja maapiirkonnas 20km. Tagamaks töökindla raportite vastuvõtu on näiteks Tallinnas ehitatud 11 vastuvõtupunkti [Joonis 25]. Nii saab viia vastuvõtuseadmed objektidele piisavalt lähedale ja saavutada vajalik signaal/müra suhe. Vastuvõtupunktid on kesk-serveriga ühenduses VDSL võrgu kaudu.

Signaalide vastuvõtupunkt koosneb VHF diapasooni FM vastuvõtjast (SATELNODE X85) [Joonis 22], mis võtab vastu sõnumeid erinevatelt valveseadmete saatjatelt ja töötleb neid ning edastab RS-232 liidese kaudu. RS232 to Ethernet konverter (MOXA nport 5450) [Joonis 23] annab paketid edasi DSL ruuterile (tavaliselt Telia renditud toode), mis on VPN kanali kaudu ühenduses turvafirma sisevõrgus oleva SNMP serveriga.

Seadmete toide on teostatud kas läbi 230VAC/ 14VDC toiteploki [Joonis 24] või 70Ah aku [Joonis 26], mis moodustavad koos katkematu toiteallika (edaspidi UPS).

1.2 Probleemi kirjeldus

1.Kui Vastuvõtupunkti 230VAC toide kaob, ei tule mingit infot selle kohta enne kui lakkab sõnumite edastus. Samas 70Ah aku kindlustab vastuvõtupunkti töö vähemalt mitmeks tunniks, mille jooksul jõuaks tehniline meeskond rikkele reageerida.

2. Ei ole mingit infot UPS aku seisukorra kohta. Vananedes pliiaku lekkevool suureneb ja aku temperatuur hakkab tõusma. Aku konkreetne eluiga sõltub suuresti keskkonnast ja tööreziimist. Temperatuuritõusust teatamine võimaldaks ennetavalt õigeaegselt hooldada toitesüsteemi.

3. Seadmekappide õhutemperatuuri monitooring aitaks välja selgitada probleemsed vastuvõtupunktid ja olukorrad ning võtta meetmeid seadmete töökindluse tagamiseks.

Kõrgemal lubatud töötemperatuurist seadmete eluiga lüheneb (kuivavad elektrolüüt-kondensaatorid) ja suutlikkus väheneb (protsessorite kiirust vähendatakse ülekuumenemise vältimiseks)

4.Välitingimustes võib temperatuuri kõikumisel õhuniiskus kondenseeruda seadmete välis- ja siseosadel. See on rikete üks tekkepõhjusi. Saades informatsiooni liiga kõrgest õhuniiskusest on võimalik kas lisaküte või ventilatsioon käivitada.

5. Kui vastvõtupunktiga on tekkinud mingi probleem, siis oleks hea operatiivselt pärida mõõdetavate parameetrite suurusi eemalt, enne tehnikute kohale saatmist. Samuti võib olla oluline seadmete katsetamisel või mõne uue töökoormuse lisamisel jälgida mingit kindlat parameetrit jooksvalt tihemini kui etteantud raporteerimisvälp.

2 Kaugseireseadme kontseptsioon (ülesande püstitus)

Käesoleva töö eesmärgiks on luua kaugseire seade, mis edastab toite- ja keskkonna parameetreid SNMP protokolli kasutades üle olemasoleva VDSL võrgu. Seade peab saama toite olemasolevast 12V pliiakust ning olema võimeline töötama pingevahemikus 9...15V Seade peab töötama temperatuurivahemikus -25°C...+85°C ja suhtelise õhuniiskuse 0.. 90% juures.

2.1 Edastatavad sõnumid

Seiresadme üleasnded:

1. Teavitada olulisest toitekatkestusest (lühiajalised 230VAC toitekatkestused pole olulised) ja teavitama backup-aku pingelangusest. Seda saab teha sama akupinge mõõtmisega, sest pliiaku laadmisreziimis on pinge ca. 13,8V ja toitekatkestuse ajal langeb akupinge suhteliselt kiiresti 12,6V-ni. Seega saab seada esimese lävendi 13V kohale, millal teavitatakse toitekatkestusest.

   Saadetakse Trap teade sisuga „toitekatkestus“

   Seadmete garanteeritud vähim tööpinge on 9 VDC. Teine lävend võks olla 12V, siis on teada et toide on pikemat aega puudu ja tuleb hooldustiim saata probleemi lahendama.

   Saadetakse Trap teade sisuga „akupinge langes alla 12V“

2. Jälgida backup aku temperatuuri. See võimaldab saada informatsiooni aku seisukorra kohta. Vananedes suureneb pliiaku sisemine juhtivus sulfateerumise ja kristalliseerumise tagajärjel lühiste tekkimisega akuplaatide vahel [5]. See tõstab aku temperatuuri ja tekib tulekahjuoht [vt. Joonis 26]. Lävendiks võks olla +60°C. Nii kõrge temperatuur on selge märk et aku vajab vahetust

   Saadetakse Trap teade sisuga "Aku temp yle +60C"

3. Jälgida seadmekapi õhutemperatuuri. Seadmete normaalne töö on tagatud temperatuurivahemikus -25°C...+55°C. Seadmekilbile on paigaldatud sundventilatsioon, mis käivitub +40°C juures. Lävendi +50°C korral saadetakse ülekuumenemise ohust.

   Saadetakse Trap teade sisuga "Temperatuur yle +50C"

4. Jälgida seadmekilbi suhtelist õhuniiskust. Kui niiskus ületab 80%, saadetakse Trap teade „Suhteline niiskus yle 80%“

5. Serveripoolsele päringule vastab agent vastava numbrilise parameetriga (INT väärtus).

3 Kaugseireseadme teostus

Kaugseireseade koosneb mikrokontrollerist, Ethernet võrguliidesest ja temperatuuri- ja suhtelise õhuniiskuse anduritest. Seade saab toite backup aku pingest ja ühendub DSL ruuteri LAN porti.

3.1 Mikrokontrolleri valik

Tellija eesmärgiks on saada odav ja lihtne, kergesti hangitavatest komponentidest seade, mis on piisava jõudlusega ja mille toiteks sobib seadmekilbis olemasolev backup-aku.

Võimalikud variandid oleks Arduino Uno rev3 ja Ethernet shield2, Raspberry Pi3 model B+, Texas Instruments ARM® Cortex®-M4F baasil loodud arendusplaadil EK_TM4C1294XL , Siemens-Thomsoni stm32f767 jne.

Esimene valik on Arduino Uno rev3 ja Ethernet2 shield. Eeliseks tuleb lugeda lihtsus, kerge laiendusvõimalus, hea saadavus. Samuti mitmekesine standard-teekide ja vabavaralise tarkvara olemasolu. Soovi korral saab seadmele lisada lokaalse LCD displei. Arduino Uno rev3 toiteks sobib alalispinge vahemikus 6...20V

Arduino Uno rev3 töötab ATmega328P 8-bitisel 16Mhz taktiga RISC protsessoril [6], evib 32KB välkmälu, 6 analoog-sisendit (0-5V, Rsis = 100MΩ), mida mõõdetakse 10-bitise täpsusega, 6 PWM-väljundiga sisend/väljundporti. Programmeerimine toimub micro-USB pordist lihtsa Arduino IDE abil. Arduino IDE lähtekoodid on GNU litsentsiga vabavara, mis toetab C++. Arduino standardteegid [7] ja kommuuni arendatud tarkvara on GNU GPL, GNU LGPL [8], creative commons [9] või mõne muu vabavaralise litsentsiga kaitstud.

3.2Sensorite valik

Sensorite valiku kriteeriumideks on võimalikult lineaarne väljundpinge antud töös vajaliku skaala ulatuses, piisav täpsusklass, kompaktsus, mõistlik hinnatase ja hea saadavus.

3.2.1 Temperatuuriandur

Temperatuurianduriks sobib Analog Devices TMP-36 [12] , mis on +/- 2% täpsusega Celsiuse skaala suhtes lineaarse karakteistikuga pooljuht-tempeatuurisensor ning töötab vahemikus -40°...+125°C. Keskkonnakindluselt Automotive klassi andur ei vaja välist kalibreerimist ja madala väljundimpedantsi tõttu sobib otse ADM sisendisse. Toitepinge võib olla vahemikus 2,7V...5.5V. Madal voolutarve (tüüpiliselt 50 μA) ja vähene isesoojenemine (alla 0,1C seisva õhu korral) on sobivad. TMP36 annab 25°C korral väljundpinge 750mV ja see muutub 10mV/°C

Väljundpinge - 40°C juures on 0,750 - [(25+40) · 0,01] = 0,1V

Väljundpinge +125°C juures on 0,750 + [(125-25) · 0,01] = 1,75V

ATmega328P analoogsisendid mõõdavad 0-5V 10 bitise täpsusega, seega jääb tempearatuuri mõõtmise resolutsiooniks (1,75-0,1) / (5/1024) = 337 nivood. See teeb mõõtmistäpsuseks (125+40) / 337 = 0,49° C. Mõõdetava skaala vähendamiseks saab vaikeväärtusena valitud 5V referentspinge asemel kasutada välist väiksemat referentspinget. Kui kasutada plaadil olemasolevat 3,3V stabilisaatori väljundit referentspinge allikana, saame mõõtmise resolutsiooniks (1,75-0,1) / (3,3/1024) = 512 astet. Mõõtmistäpsus tuleb seega (125+40) / 512 = 0,32 °C

3.2.2 Realtiivse õhiuniiskuse (RH) andur

Relatiivse niiskuse anduriks sobib mahtuvusliku mõõteelemendiga Honeywell HIH_4030 [13] , mis mõõdab RH 0-100% -40°C kuni +125°C juures, on lineaarne täpsusega 3,5% ja tarbib 200μA 4 - 5,8V toitepinge juures. Väljundpinge on antud valemiga Vout = (Vtoide) · (0,0062 · (sensor RH) + 0,16) (1)

0% juures on seega teoreetiline väljundpinge Vout = 0,16V

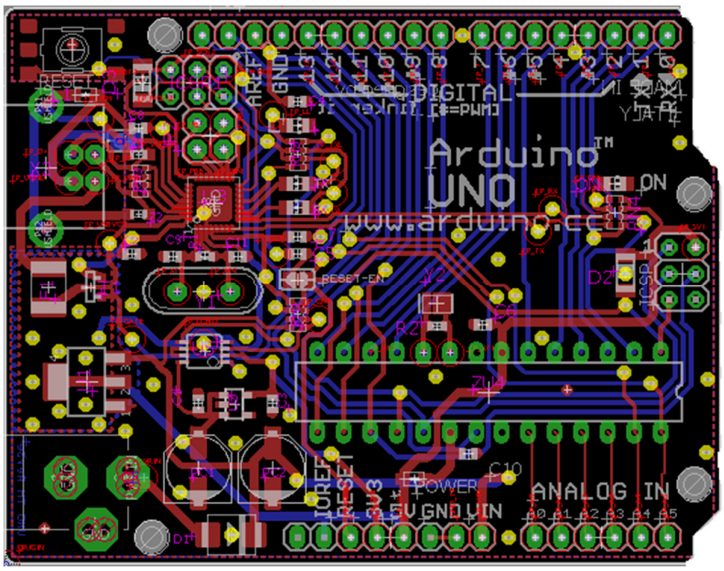
Valides toitepingeks lubatud miinimumi lähedase 4,2V, on tulemuseks hea sobivus 3,3V sisendiga:
100% RH juures on teoreetiline väljundpinge Vout 4,2 · (0,0062 · 100 + 0,16) = 3,28V
Siis eelnevast väljundpinge valemist (1)

Sensor RH [%] = (3,3 / 1024 · lugem / 4,2 -0.16) / 0,0062 (2)

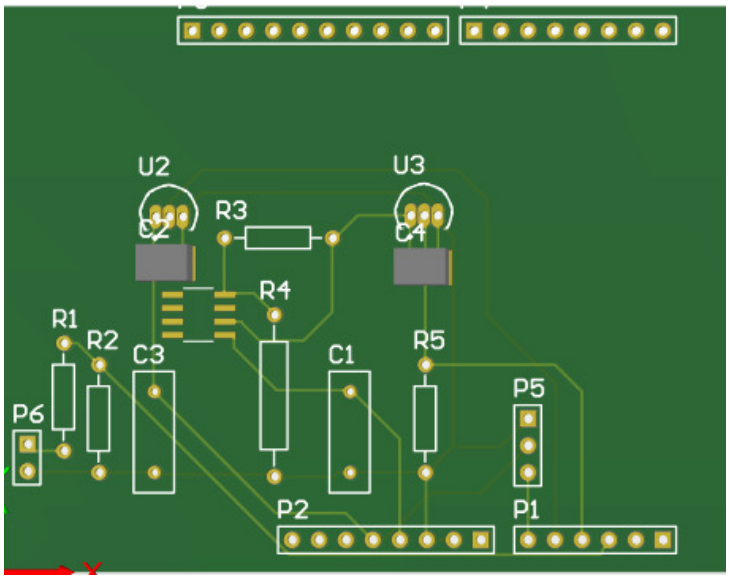
4 Skeemi koostamine

4.1 Mikrokontroller

Arduino Uno rev 3 ja Ethernet shield2 on realiseeritud sama mõõduga (53 x 68mm) teineteise peale ühendatavate moodulitena. Omaloodud andurite sobitus on mõistlik luua samade mõõtude ja terminalide asetusega kolmanda moodulina.



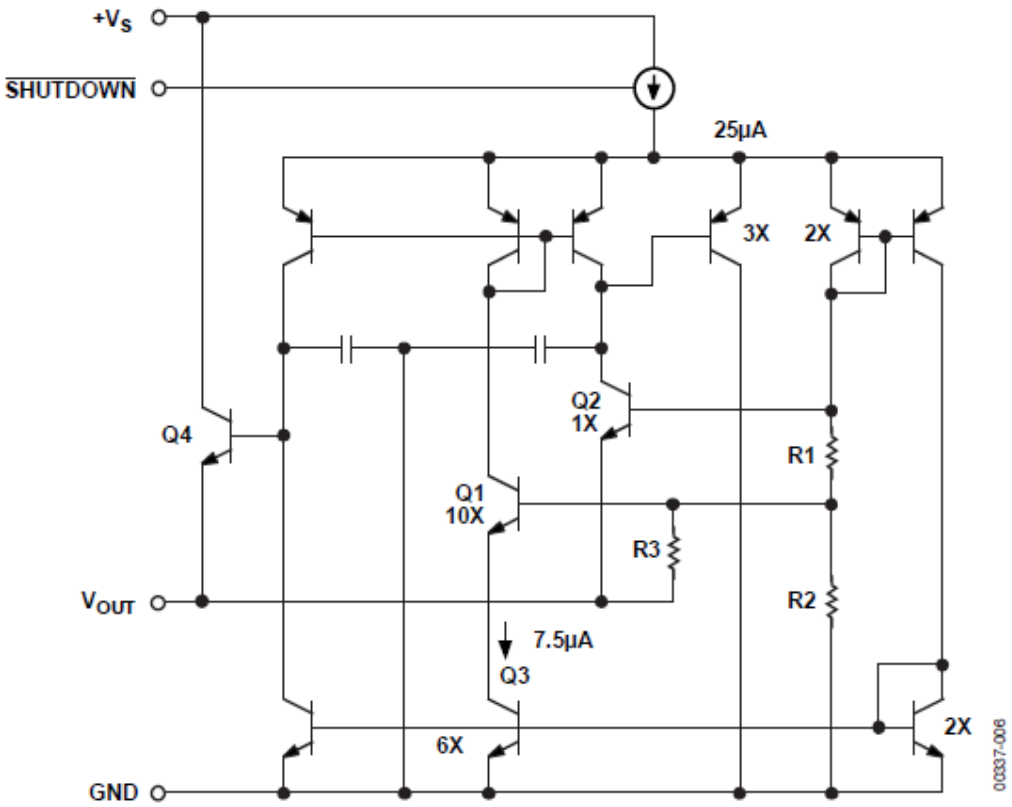
Joonis 1. Arduino Uno rev3, primaarkülje vaade [14]



Joonis 2. Omaloodud plaadi kavand, primaarkülje vaade

4.2 Sensorite ühendamine

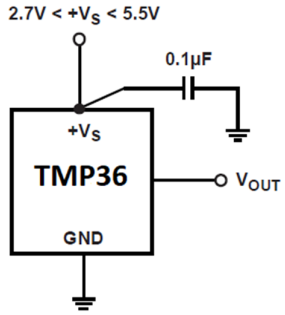
4.2.1 Temperatuurisensor TMP36



Joonis 3. Temperatuurisensori TMP36 põhimõtteskeem [12]

Sensor TMP36 töötab transistorite Q1 ja Q2 kollektor-emitterpingete võrdlemisel. Q3 hoiab mõlema transistori voolu umbes 7,5 µ A juures. Q1 emittersiirde laius on 10 korda suurem kui Q2 emittersiirde laius sama kollektorvoolu juures, seega Q1 ja Q2 baasi ja emitteri vaheliste pingte erinevus

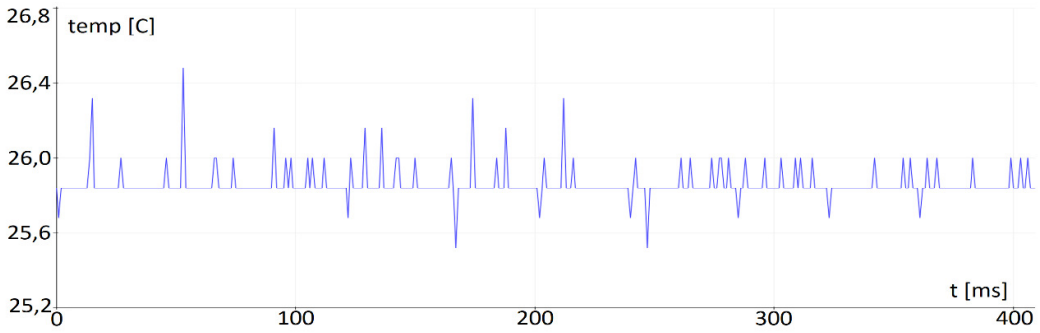
avaldub $\Delta V_{BE} = V_T \cdot \ln\left(\frac{AE,Q1}{AE,Q2}\right)$ (3) [12] , kus AE on emittersiirde laius.
Takistitega R1 ja R2 määratakse transistoride väljundpinge muutus temperatuurimuutuse suhtes (sensori tundlikkus [mV/C]) , R2 ja R3 määravad Q1 baas-emitterpinge, mis on omakorda väljundpinge algnihe (DC offset [V]). Sensori väljundiks on vooluvõimendi Q4 väljund, mille lühisvool on piiratud 250 µ A-ga.[12]



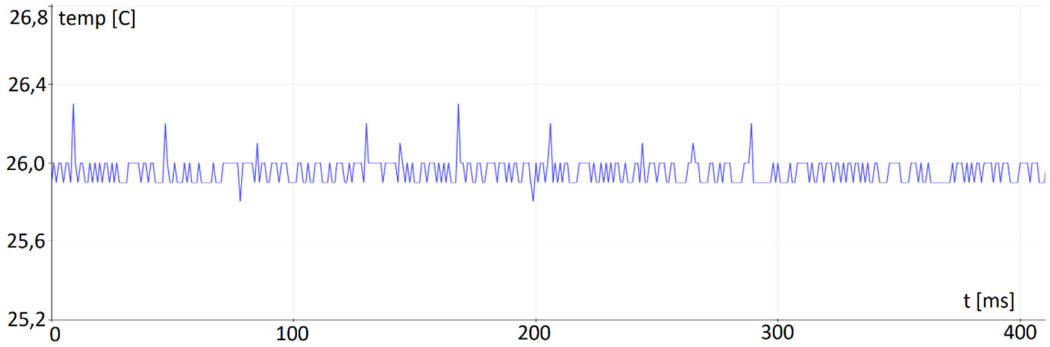
Joonis 4. TMP36 üendamise skeem [12]

0,1 µF shunteeriv kondensaator toitel peaks olema keraamiline, minimaalse pikkusega väljaviikudega (soovitatav on SMD) ja asuma võimalikult lähedal temperatuurisensori toiteklemmidel. Kuna TMP36 töötab väga väikese toitevooluga, on oluline mürarikkas keskkonnagas vähendada raadiosagedushäirete (RFI) mõju sensorile. RFI mõju nendele temperatuurisensoritele avaldub üldjuhul väljundpinge ebanormaalsete alalispinge-nihetena, mis on tingitud sensori kõrgsagedusliku ümbritseva mürasignaali alaldamisest. Kui seadmeid kasutatakse kõrgsagedusliku kiirguse või vooluringi müra juuresolekul, võib 0,1 µF keraamilise kondensaatori peale paigutada suure väärtusega (2,2 µF) tantaalkondensaatori täiendavaks mürasummutuseks.[12]

Aku temperatuuri mõõtmiseks kasutatav sensor monteeritakse koos summutuskondensaatoritega termilisse kontakti mõõdetava kehaga (aku korpusega)

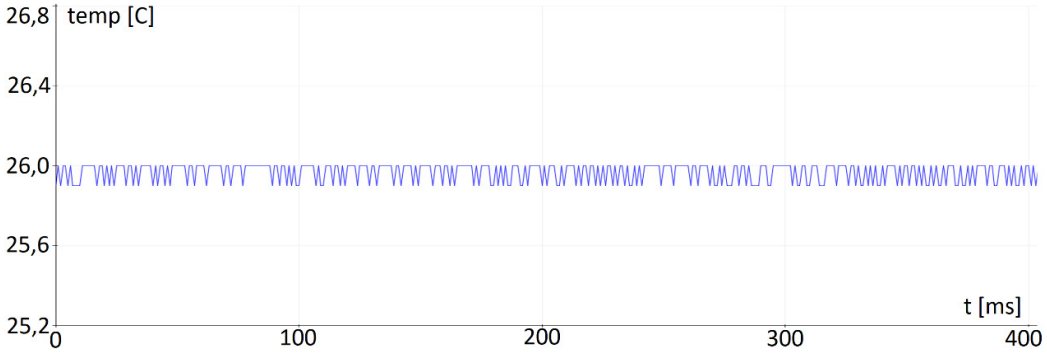


Joonis 5. Müra mõõdetuna Arduino jadapordis , TMP36 5V Arduino terminalist toidetuna, ilma summutuskondensaatorita. Lugem väreleb +/- 0,5 kraadi



Joonis 6. Müra mõõdetuna jadapordis, TMP36 3,3V Arduino terminalist toidetuna, ilma 0,1 µF kondensaatorita. Müra on endiselt olemas, tipud ulatuvad +/-0,3kraadi

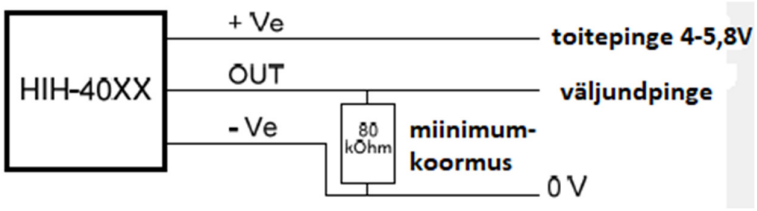
ADM muunduril paremini stabiliseeritud 3,3V pinge kasutamine referents-pingena võimaldab vähendada müra amplituudi ja läbi parema sobituse saada täpsem mõõtetulemus.



Joonis 7. Müra mõõdetuna Arduino jadapordis. TMP36 Arduino 3,3V terminalist toidetuna, silutud 0,1 µF-ga. Väljundpinge värelus mahub ühe digiteerimisastme sisse.

4.2.2 Suhtelise õhuniiskuse (RH) anduri ühendusskeem

Toitepingeks on valitud 4,2V parima sobituse saavutamiseks (vt.3.2.2.)



Joonis 8. Suhtelise õhuniiskuse (RH) anduri ühendusskeem [13]

4.2.3 Akupinge mõõtmise skeem

Akupinge võib maksimaalselt küündida kuni 15 voldini. Samas on ADM sisendi mõõteulatus 3,3 volti maksimaalselt. Kasutan 820/220 Ω pingejagurit, sest 15-voldise sisendpinge korral on jaguri väljundis

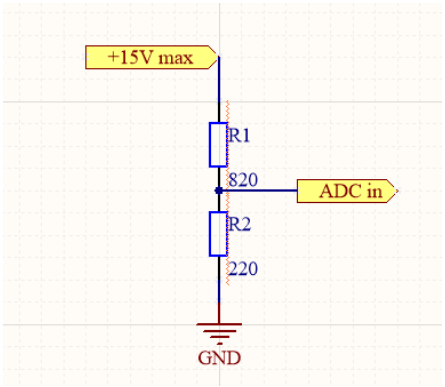
15 · 220 / (820+220) = 3,17 V

See on ka lähim väärtus alla 3,3V kasutades E24 nominaale

Mõõtmistäpsus 3,1/3,3 · 1024 = 962 astet, mis teeb 15/ 962 = 0,016 V

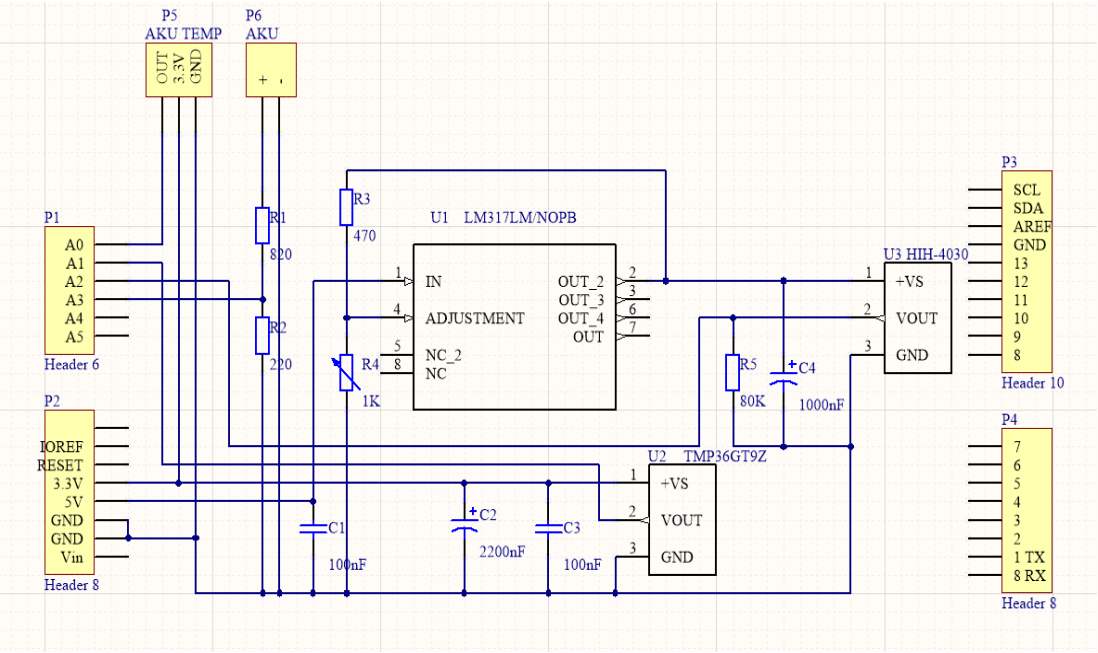
ADM sisendtakistuse Rsis=100M Ω mõju pingejaguri alumisele õlale võib lugeda tühiseks: Rsis || R2 = $\frac{220 \cdot 100 \cdot 10^6}{220 + 100 \cdot 10^6} = \frac{2{,}2 \cdot 10^{10}}{220 + 10^8}$ = 219,9995 Ω

Sellise jaguriga 3,3V ADM sisend suudab mõõta kuni 3 / 220 = 15,6V



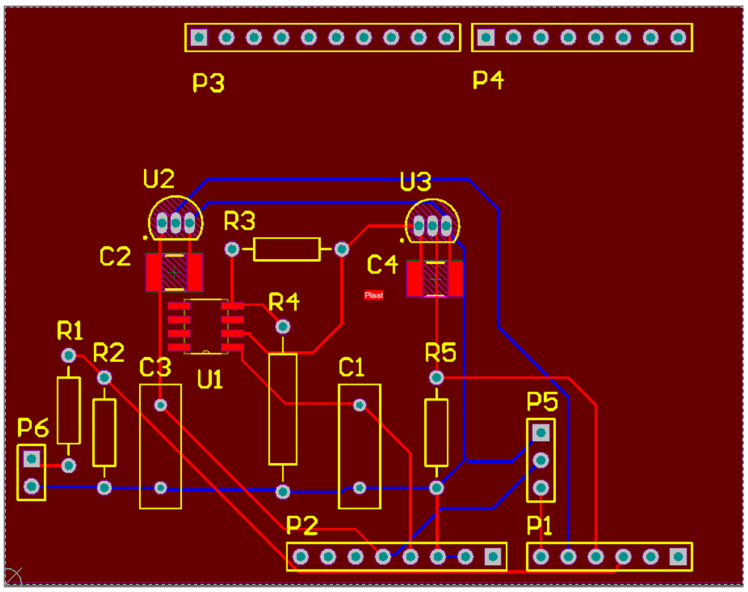
Joonis 9. Akupinge jagur 3,3V ADM jaoks .

4.3 Andurite plaadi skeem

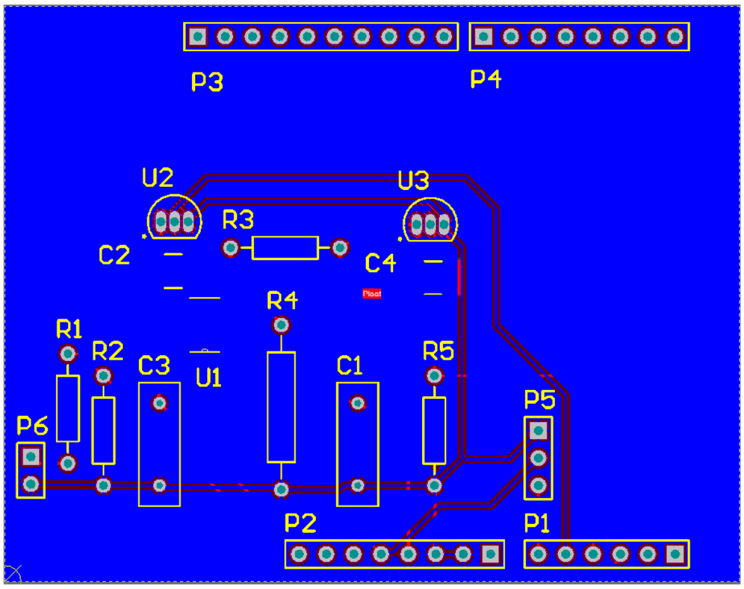


Joonis 10. Andurite plaadi skeem

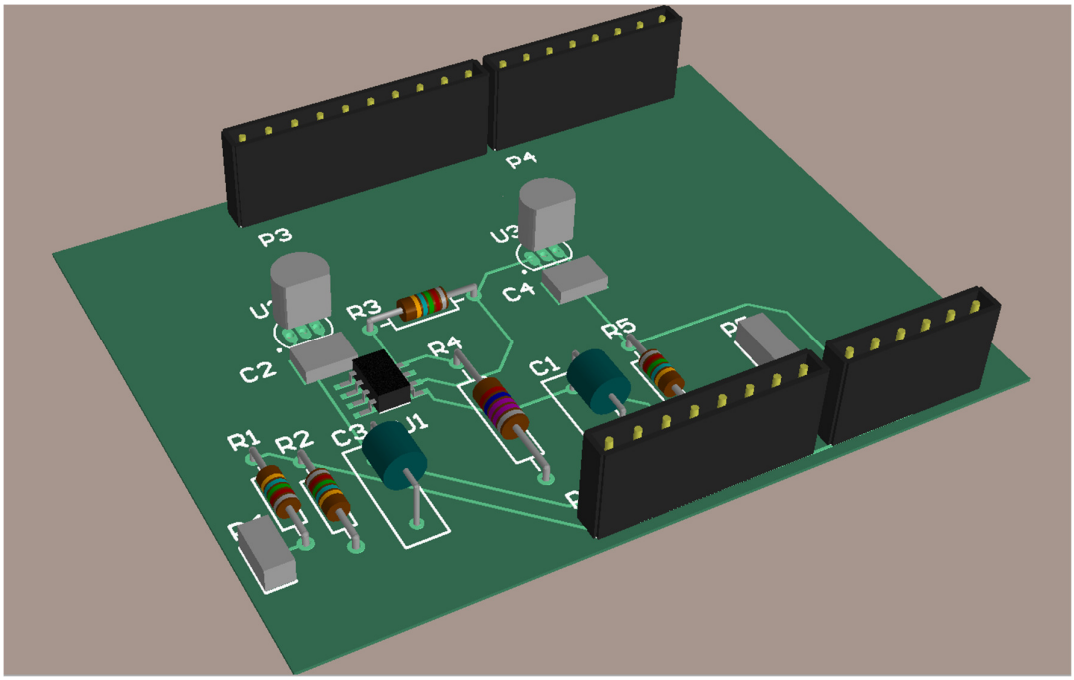
R4 potensiomeetri abil tuleb pingeregulaatori U1 (LM317) väljundpinge seada 4,2V peale. Aku temperatuuri sensor ühendtakse eraldi kaabliga P5 terminali.



Joonis 11. Arduino uno rev3 peale sobiv sensorite trükkplaat, primary side 2D vaade



Joonis 12. Sensorite plaadi bottom layer, polügoniga kaetud varjestamiseks



Joonis 13. 3D vaade, primaarkülg

5 Kontrolleri tarkvara.

Kontrolleri tarkvara peab koguma ettentud regulaarsusega ADM väljundist lugemit, arvutama neist mõõdetava parameetri tõesed numbrilised väärtused, võrdlema tulemit etteantud piirtingimustega. Tingimuste täitmisel saatma SNMP Trap sõnumi vastavalt iga mõõdetud parameetri OID numbri ja ajatempliga varustatult UDP datagrammina etteantud serveri IP aadressi ja pordi peale. Samas peab olema kuuldel serveripoolse päringu vastuvõtmise ja täitmise jaoks. Serveri päringule vastab konrtoller konkreetse päritud OID mõõtetulemusest arvutatud numbrilise väärtusega.

5.1 SNMP protokoll.[1]

SNMP on võrguseadmete haldamise protokoll.
SNMP protokoll jookseb manageri ja hallatava üksuse vahel ning võimaldab nii andmete kogumist kui juhtimiskäskude edastamist.

5.2 SNMP arhitektuur.[1]

SNMP arhitektuur koosneb hallatavast seadmest ja manager-seadmest.

Hallatav seade koosneb riistvarast ja tarkvarast (kuhu kuulub ka agent – protsess, mis jookseb hallatavas seadmes ja võtab info edastamiseks ühendust manager-seadmega ja kuulab võrku manageri teadete vastuvõtuks) .
Hallatavas seadmes võib olla mitu haldusobjekti,. Need on konkreetsed riistavara osad (näiteks võrgukaart vms.). Igal objektil on oma identifitseeritav number (OID)

Näiteks RFC1213-MIB OID puu [10][11]

// .iso (.1)
// .iso.org (.1.3)
// .iso.org.dod (.1.3.6)
// .iso.org.dod.internet (.1.3.6.1)
// .iso.org.dod.internet.mgmt (.1.3.6.1.2)
// .iso.org.dod.internet.mgmt.mib-2 (.1.3.6.1.2.1)
// .iso.org.dod.internet.mgmt.mib-2.system (.1.3.6.1.2.1.1)

// .iso.org.dod.internet.mgmt.mib-2.system.sysDescr (.1.3.6.1.2.1.1.1)
const char sysDescr[] PROGMEM = "1.3.6.1.2.1.1.1.0"; // *read-only (DisplayString)*
// .iso.org.dod.internet.mgmt.mib-2.system.sysObjectID (.1.3.6.1.2.1.1.2)
const char sysObjectID[] PROGMEM = "1.3.6.1.2.1.1.2.0"; // *read-only (ObjectIdentifier)*
// .iso.org.dod.internet.mgmt.mib-2.system.sysUpTime (.1.3.6.1.2.1.1.3)
const char sysUpTime[] PROGMEM = "1.3.6.1.2.1.1.3.0"; // *read-only (TimeTicks)*
// .iso.org.dod.internet.mgmt.mib-2.system.sysContact (.1.3.6.1.2.1.1.4)
const char sysContact[] PROGMEM = "1.3.6.1.2.1.1.4.0"; // *read-write (DisplayString)*
// .iso.org.dod.internet.mgmt.mib-2.system.sysName (.1.3.6.1.2.1.1.5)
const char sysName[] PROGMEM = "1.3.6.1.2.1.1.5.0"; // *read-write (DisplayString)*
// .iso.org.dod.internet.mgmt.mib-2.system.sysLocation (.1.3.6.1.2.1.1.6)
const char sysLocation[] PROGMEM = "1.3.6.1.2.1.1.6.0"; // *read-write (DisplayString)*
// .iso.org.dod.internet.mgmt.mib-2.system.sysServices (.1.3.6.1.2.1.1.7)
const char sysServices[] PROGMEM = "1.3.6.1.2.1.1.7.0"; *// read-only (Integer)*

Hallatavate objektide info (OID numbrid) on kogutud infobaasi- *Management Information Base* (MIB)

IETF kasutab ISO [ISO X.680 2002] MIB moodulite standardiseerimisel , mis seostuvad ruuterite, host-arvutite ja muude võrguseadmetega. Seisuga 2006 oli rohkem kui 200 standardset (public) MIB moodulit ja veel enam vendor-specific (private) MIB moodulit. [RFC 5000] loetleb kõik standardiseeritud MIB moodulid seisuga 2008. Hierarhia tipus on ISO ja Telecommunication Standardization Sector of the International Telecommunication Union (ITU-T)

Kõigile süsteemidele omased objekti süsteemi andmed on igasse MIB baasi "sisse ehitatud"

Tabel 1. MIB süsteemi andmed [1, lk.772]

Objekti ID	nimi	tüüp	Kirjeldus
1.3.6..2.1.1.1	sysDescr	octet string	Süsteemi täisnimi,versioon, riistvara tüüp, op. süsteem,ja võrgu tarkvara tüüp
1.3.6..2.1.1.2	sysObjectID	Object Identifier	Vendoripõhine ID
1.3.6..2.1.1.3	sysUpTime	TimeTicks	Aeg (1/100sek) viimasest süsteemi initsialiseerimisest
1.3.6..2.1.1.4	sysContact	octet string	üksuse manageri kontakt
1.3.6..2.1.1.5	sysName	octet string	administratiivne üksuse nimi
1.3.6..2.1.1.6	sysLocation	octet string	üksuse füüsiline asukoht
1.3.6..2.1.1.7	sysServices	Integer32	üksusel kasutatavate teenuste hulk

Manager-seadmes kasutatakse tavaliselt MIB brauserit - tarkvara, mis sisaldab MIB andmebaasi, ja kasutajaliidest hallatavate seadmetega suhtlemiseks.
Käesoleva töö raames on tegemist nädiskoodiga kinnises võrgus. Reaalselt töötavas seadmes peaks ettevõte kasutama oma PEN numbrit või kui seda pole avalikus loetelus[3] siis taotlema omale tasuta unikaalse PEN numbri vastava IANA vormi täitmisel [4]

5.3 SNMP protokolli operatsioonid

SNMP protokolli kasutatakse agentidelt info kogumiseks ja nende juhtimiseks manageri poolt.
Kõige levinum on kasutada päringu-vastuse režiimi, kus manager saadab päringu (request) SNMP agendile, mis võtab päringu vastu, teeb nõutavad toimingud ja saadab vastuse (response) tagasi managerile.
Teine levinum SNMP agendi kasutus on saata iseseisvalt teateid managerile (Trap message). Trap sõnumeid kasutatakse hallatava MIB objekti ebatavalise seisundi või mõõdetava parameetri saatmiseks.

Tabel 2. SNMPv2 defineerib 7 erinevat käsklust [1, lk.774]

SNMP v2 PDU tüüp	saatja-vastuvõtja	kirjeldus
GetRequest	managerilt agendile	Küsib ühe MIB objekti väärtuse
GetNextRequest	managerilt agendile	Küsib järgmise väärtuse listist või tabelist
GetBulk	managerilt agendile	Küsib ploki kaupa väärtusi
InformRequest	managerilt managerile	Informeerib eemal asuvat manageri MIB väärtustest
SetRequest	agendilt managerile	Asetab MIB objekti väärtuseks
Response	agendilt managerile või managerilt managerile	Vastus päringule GetRequest, GetBulk, InformRequest
SNMPv2-TRAP	agendilt managerile	Informeerib manageri sündmusest

Tabel 3. SNMP GET/SET käsu formaat [1, lk.774]

	GET/SET header			muutujad					
Käsu tüüp (0-3)	Päringu id	Veakood (0-5)	Vea indeks	Nimi	Väärtus	Nimi	Väärtus	….	

Tabel 4. SNMP TRAP sõnumi formaat [1, lk.774]

	TRAP header					Trap Info			
Käsu tüüp (4)	Enterprise	Agendi aadress	TRAP tüüp	kood	Ajatempel	Nimi	Väärtus	….	

SNMP päringud võivad kasutada erinevaid transpordikihi protokolle, kuid RFC3417 määrab eelistavaks UDP protokolli.[1, lk.775]

5.4 Arduino teegid [15]

Kontrolleri tarkvarana on kasutatud vabavaralist C++ keeles Arduino jaoks kirjutatud tarkvara. Standardteegid [7] on vajadusel modifitseeritud oma kasutaja-andmetega, põhiprogramm [15] on Arduino community liikmete poolt alustatud ja autori poolt täiendatud. Kasutatav tarkvara koosneb üheksast failist:

• Agent.ino – põhiprogramm, milles toimub sensoritelt lugemite kogumine, lävendiga võrdlemine ja Trap sõnumi initsialiseerimine.

• Agentuino.cpp – SNMP agendi teek. Sissetuleva päringu dekodeerimine ja vastamine, UDP paketi sisu kokkupanek.

• Agentuino.h – header file Agentuino.cpp-le, OID dekodeerimne, ASN.1 funktsioonide dekodeerimine ja kodeerimine (int, int16, int32, uint64, octet string, opaque syntax, boolean formaatide konverteerimine)

• Ethernet2.h – standardteek , kus defineeritakse Ethernet shield2 protsessorisse pärast iga reseti kirjutatavad parameetrid (mac, IP, gateway, subnet, hiljem ka dns Srveri IP, domeeni nimi, host-arvuti nimi)

• EthernetUdp2.h – standardteek, mille abil toimub UDP datagrammide saatmine ja vastuvõtt Ethernet shield2 kaudu.

• MIB.h – teek, kus on salvestatud agendi OID puu

• Streaming.h – standardteek, (Arduino.h –le Streaming library version 4 tugi)

• Variable.cpp – teek, kus on kasutaja süsteemi standard-andmed

• Variable.h – Variable.cpp header fail , võimaldab kasutaja parameetreid hoida Ethernet shield2 SD kaardil

5.4.1 Trap sõnumi saatmine -Agent,ino

Agent.ino on põhiprogramm, mida kasutatakse sensorite lugemiseks, ADM sisendväärtusest saadetava näidu või teate lävendi arvutamisel. Arduino sketch (laetav programm) koosneb seadistustest (setup osast), mida jooksutatakse pärast igat alglaadimist üks kord ja tsüklilisest osast (void loop), mis jookseb pidevalt.

eeldeklaratsioonide osas kaastakse vajalikud teegid:

```
#include "Streaming.h"
#include <Ethernet2.h>
#include <SPI.h>
#include "Agentuino.h"
#include "MIB.h"
#include "Variable.h"
```

Määratakse oma MAC aadress ja IP aadress,Trap vastuvõtja IP aadress ning vajadusel gateway ja võrgumask

```
static byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xEF}; static byte
ip[4] = {10, 0, 0, 166};            //oma ip aadress
//static byte gateway[] = {192, 168, 2, 1};
```

```
//static byte subnet[] = {255, 255, 255, 0};
static byte RemoteIP[4] = {10, 0, 0, 165};// Trap receiver ip aadress
```

määratakse ADM sisendid, kuhu on ühendatud sensorid, kirjeldatakse muutujad

```
int tempA0=0;                    //aku temperatuur A0
int tempA1=0;                    //õhu temperatuur A1
int humidA2=0;                   //õhu niiiskus A2
int voltageA3=0;                 //akupinge A3
```

seadistuste osas (void setup) määratakse ADM referentspingeks väline pinge:

`analogReference(EXTERNAL);` // kasutame välist ref. pinget (3.3V)

initsialiseeritakse võrguühendus:

```
Ethernet.begin(mac, my_IP_address);
    //Ethernet.begin(mac,ip,gateway,subnet);
IPAddress address = Ethernet.localIP();
    for (uint8_t i=0;i<4;i++) {
        my_IP_address[i] = address[i];
        Serial.print(my_IP_address[i]);
        Serial.print(".");
    }
    Serial.println("");
```

Ja lokaalseks jälgimiseks avatakse serial port:

```
Serial.begin(115200);
    Serial.println("CPU Start");
```

Tsüklilises osas (void loop) käivitatakse kõigepealt sissetulevate SNMP sõnumite kuulamine:

```
Agentuino.listen();
```

Selles osas toimub ka sensorite väljundpingete lugemine, väljundi arvutus, lävendiga võrlemine ja tingimuse täitmisel Trap sõnumi saatmise algatamine.

Sensorite müra vähendamiseks on kasutatud 10-kordse lugemise keskmistamist. Mõõteseeria periood on pikendatud 10mS viidete abil juhuslike müratippude mõju hajutamiseks.

Näitena analoogsisendist A0 aku temperatuuri mõõtmine ja arvutus:

```
 //aku tempearatuuri keskmistatud 10x mõõtmine A0-st
   int i=0;
   float readingA0 =0;
    for (i = 0; i < 10; i++) {
      readingA0 = readingA0+analogRead(A0);
     delay(10);
      }
    readingA0 = readingA0/10;

 //LM36 lugemist reaalse C temperatuuri arvutus  (3308/1024=3.23)
//sest mõõdetud ref pinge on antud isendil  3.308V
  float tempA0C=(((readingA0*3.23)- 500)/10);
  int tempA0= tempA0C;                      //A0 temperatuurimuutuja
int kujule
   if (tempA0 > 60){                                  // trap
saatmise tingimus

   Serial.println("aku temp korge!");
   Serial.println(tempA0);
Agentuino.Trap("Aku temp yle +60C", RemoteIP, locUpTime,
"1.3.6.1.2.1", "1.3.6.1.2.1.11.19.0");
```

lõplik TRAP sõnumi koostamine ja saatmine toimub teegis Agentuino.cpp, funktsioonis void AgentuinoClass::Trap(char Message[], byte RemIP[4], uint32_t Time, char enterprise_oid[], char oid_[]) [vt. 5.2]

5.4.2 Agentuino.cpp , GET NEXT request

Kui Ethernet Shield2 sisendpuhvris on pakette ja pointer pole null, käivitatakse funktsioon

```
Udp.parsePacket();
    if (Udp.available() && _callback != NULL) (*_callback)();
```

ja loetakse pakett

```
  Udp.read(_packet, _packetSize);
    //      Udp.readPacket(_packet, _packetSize, _dstIp, &_dstPort);
```

Seejärel eraldatakse UDP datagrami päis, valideeritakse pakett, määratakse SNMP sõnumi versioon (0 = snmp ver1), valideeritakse community nime pikkus

```
pdu->version = 0;
    for (i = 0; i < verLen; i++) {
        pdu->version = (pdu->version << 8) | _packet[5 + i];
```

vea korral tagastatakse teade: `return SNMP_API_STAT_NAME_TOO_BIG;`

valideeritakse community nimi OID puu vastu

vea korral tagastatakse teade: `return SNMP_API_STAT_NO_SUCH_NAME;`

eraldatakse objekti identifikaator

```
memset(pdu->OID.data, 0, SNMP_MAX_OID_LEN);
    pdu->OID.size = obiLen;
    for (i = 0; i < obiLen; i++) {
        pdu->OID.data[i] = _packet[eriEnd + 7 + i];
```

määratakse väärtuse tüüp

```
 pdu->VALUE.syntax = (SNMP_SYNTAXES) valTyp;
```

väärtuse pikkus

```
if (obiLen > SNMP_MAX_VALUE_LEN) {
```

kui liiga pikk, tagastatakse veateade `return SNMP_API_STAT_VALUE_TOO_BIG;`

kui kõik eelnevad tingimused on täidetud, tagastatakse väärtus
`return SNMP_API_STAT_SUCCESS;`

käivitatakse funktsioon

```
void AgentuinoClass::onPduReceive(onPduReceiveCallback pduReceived) {
    _callback = pduReceived;
}
```

Ja lõpuks vabastatakse mälu

```
void AgentuinoClass::freePdu(SNMP_PDU *pdu) {
    //
    memset(pdu->OID.data, 0, SNMP_MAX_OID_LEN);
    memset(pdu->VALUE.data, 0, SNMP_MAX_VALUE_LEN);
    free((char *) pdu);
```

seejärel koostab funktsioon vastuse ja saadab küsijale tagasi

```
void AgentuinoClass::Trap(char Message[], byte RemIP[4], uint32_t
Time,  char enterprise_oid[], char oid_[])
```

pärast parameetrite hankimist koostatakse UDP datagramm

```
Udp.beginPacket(RemIP, 162); // Sihtkoha IP ja port, kuhu saadetakse trap
Udp.write(Type_and_leng, 2); // Sõnumi tüüp ja pikkus baitides
Udp.write(Version, 0); // 0 = snmp versioon 1
Udp.write(Comunity_string, 2 + strlen(_getCommName)); //Saaja comm. nimi
Udp.write(PDU_type_and_leng, 4); // type 4= octet string
Udp.write(OID, sizeof (OID) / sizeof (OID[0])); // obkjekti identifikaator
number
Udp.write(my_IP_address, 4); //oma IP aadress
Udp.write(Type_Trap, 3); // Trap sõnumi tüüp
Udp.write(extra_OID, 3); //
Udp.write(Type_time_stick, 2); //ajatempli tüüp
Udp.write(hexadecimalNumber, 4); //ajatempli väärtus
Udp.write(Var_Bind, 4);// muutuja , pikkus baiti
Udp.write(Var_Bind1, 4); );// muutuja , pikkus baiti
Udp.write(OID1, sizeof (OID1) / sizeof (OID1[0]));
Udp.write(Value1, 2); // saadetav väärtus
Udp.write(Message, strlen(Message)); //segmendi pikkus, kontrollsumma
Udp.endPacket();
```

5.4.3 EthernetUDP2.h teek

SNMP kasutab transpordikihtiks UDP datagrammi sõnumite saatmiseks ja vastuvõtuks.

	okt	0								1								2								3							
okt	bitt	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Allika port																Sihtkoha port														
4	32	Segmendi pikkus																Kontrollsumma														

Joonis 14. UDP datagrammi päis, pikkused bittides [16]

EthernetUDP2.h teek saadab UDP datagrammi :

UDP datagrammi saatmine

Määratakse paketi saaja IP aadress ja port

```
virtual int beginPacket(IPAddress ip, uint16_t port);
```

Tagastab 1, kui see on edukas, 0, kui tarnitud IP-aadressi või -portiga ilmnes viga

Alustatakse UDP datagrammi loomist, mis saadetakse eelpool määratud IP ja porti

```
virtual int beginPacket(const char *host, uint16_t port);
```

Tagastab 1, kui see on edukas, 0, kui hosti nime või pordi lahendamisel tekkis probleem

```
virtual int endPacket();
```

lõpetab paketi ja saadab selle, tagastab 1, kui pakett saadeti edukalt, 0, ilmnes viga

```
virtual size_t write(uint8_t);
```

Kirjutab paketti ühe baidi

```
virtual size_t write(const uint8_t *buffer, size_t size);
```

Kirjutab puhvrist const uint8_t jagu baite paketti

UDP paketi vastuvõtmine

Järgmise saadaval oleva paketi töötluse algus

```
virtual int parsePacket();
```

Tagastab paketi suuruse baitides või 0, kui pakette pole saadaval

```
virtual int available();
```

Praeguses paketis jäänud baitide arv

```
virtual int read();
```

Loeb ühe baidi praegusest paketist

```
virtual int read(unsigned char* buffer, size_t len);
```

Loeb kuni „len“ arvu baite praeguste pakettide baitidest ja asetab need puhvrisse

Tagastab loetud baitide arvu või 0, kui ükski pole saadaval

```
virtual int read(char* buffer, size_t len) { return read((unsigned
char*)buffer, len); };
```
Loeb praegusest paketist kuni „len“ arvutähemärki ja asetab need puhvrisse
Tagastab loetud baitide arvu või 0 kui ühtegi ei loetud

```
virtual int peek();
```

Tagastab järgmise baidi praegusest paketist ilma edasi liikumata

```
virtual void flush();     // Lõpetab praeguse paketi lugemise
```

Tagastab sissetulnud paketi saatja IP aadressi

```
virtual IPAddress remoteIP() { return _remoteIP; };
```

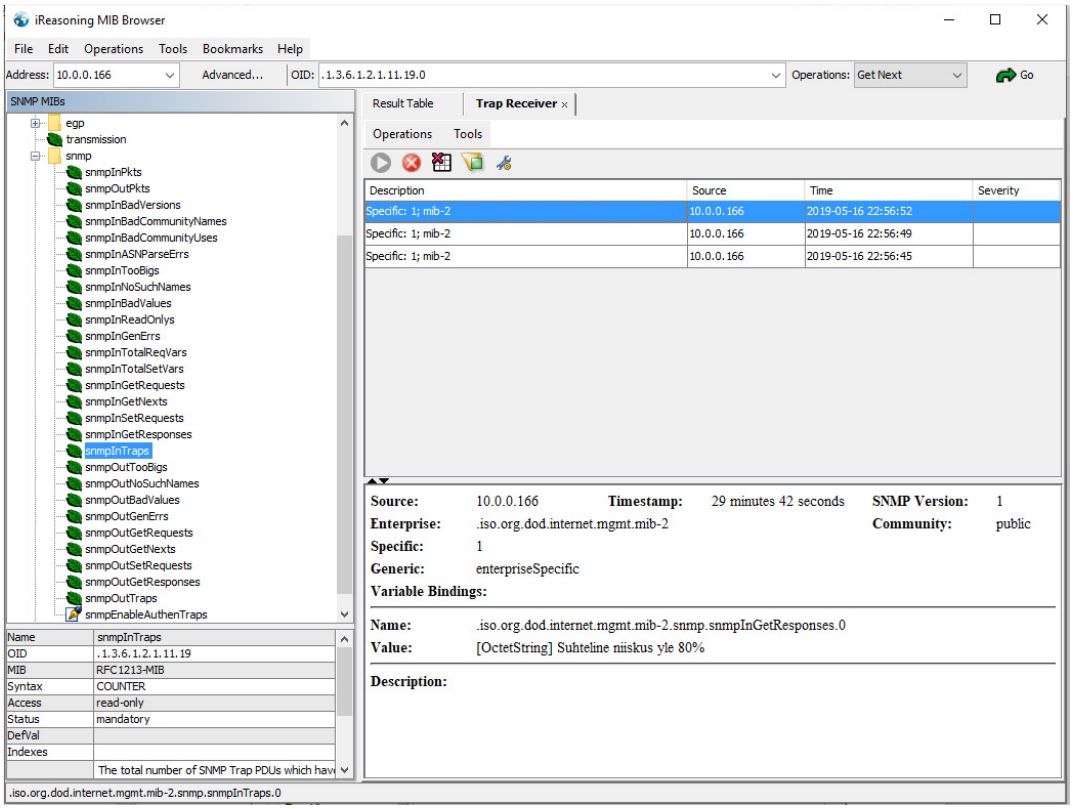
Tagastab sissetulnud paketi pordi numbri

```
virtual uint16_t remotePort() { return _remotePort; };
```

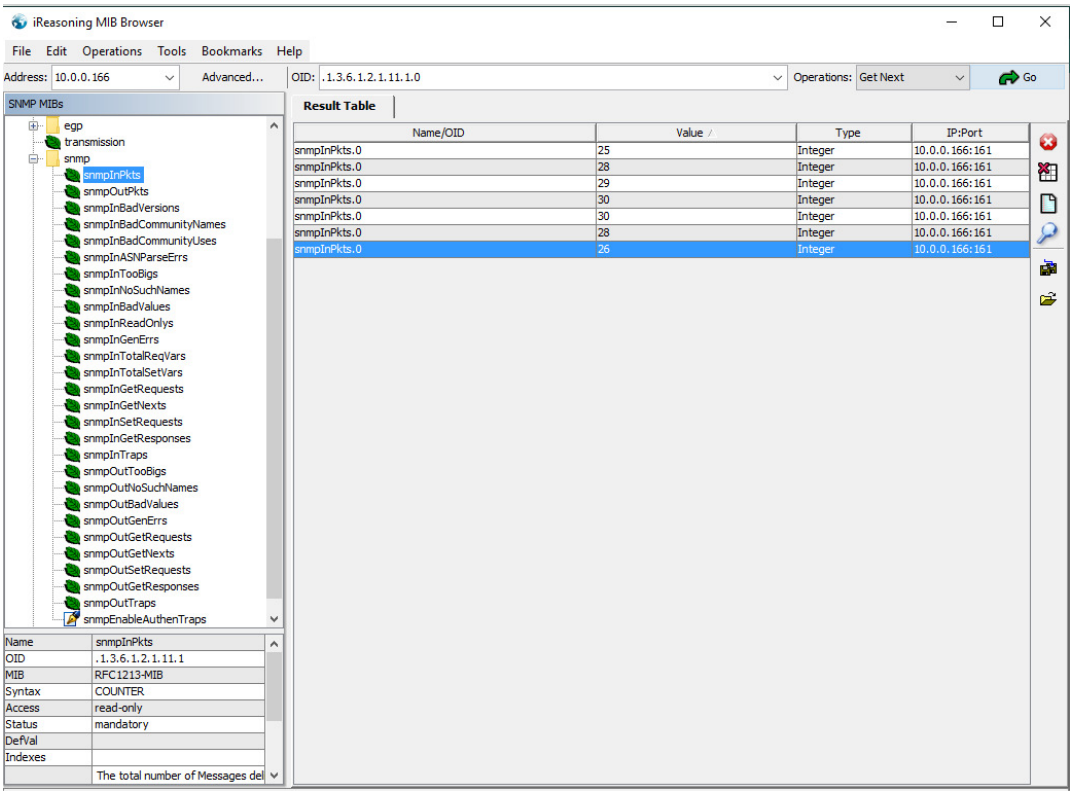
5.5 MIB browser iReasoning version 12.0 [17]

Arendatava seadme katsetmisel on kasutatud TRAP resiiverina ja päringute saatmiseks iReasoning Networks toodetud MIB brauseri tarkvara erakasutuseks mõeldud vabavaralist versiooni 12.0. Lõpliku SNMP serveri tarkvara valiku teeb Turvafirma lähtuvalt oma ülejäänud tarkvaraga ühildumisest ja kasutusmugavusest ning majanduslikest kaalutlustest lähtudes. Samuti peab orgaisatsioon endale taotlema unikaalse PEN numbri [vt.6.2] ja kujundama oma vajadustele vastava OID puu. Igale OID numbrile vastab ühe mõõdetava parameetri kirjeldus MIB brauseris.

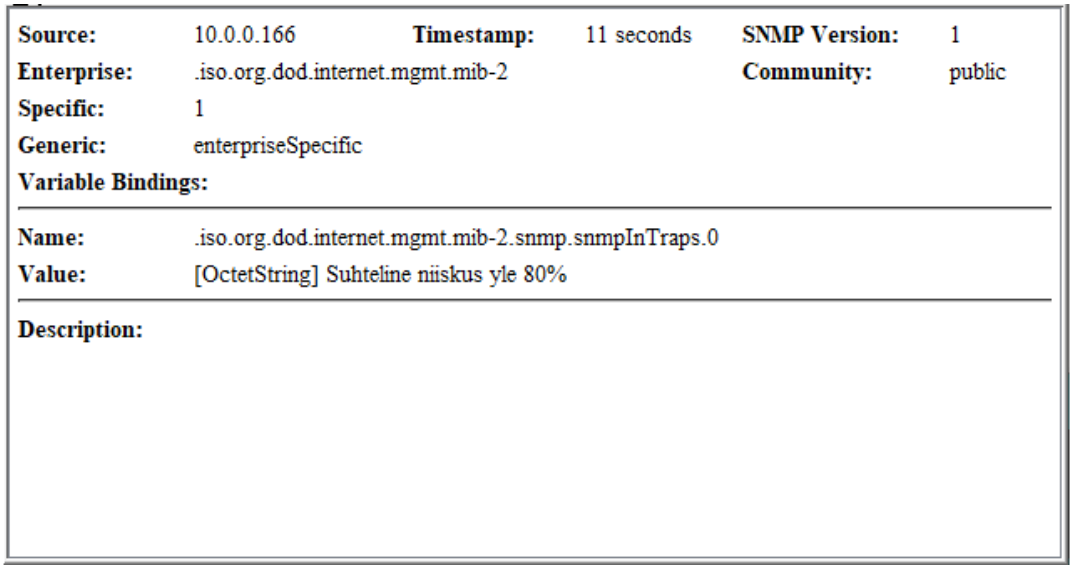
Katse ajal on kasutatud standardset iso.org.dod.internet.mgmt.mib-2 [4] infobaasi. Isoleeritud võrgu tõttu ei ole probleemiks võimalik pakettide leke väljaspoole ettevõtte tulemüüri. Katsetustel on seatud MIB brauseri (serveri) IP aadressiks 10.0.0.165 (seireseadme IP on katses 10.0.0.166). Kuulatakse porti 161, saadetakse pordist 162



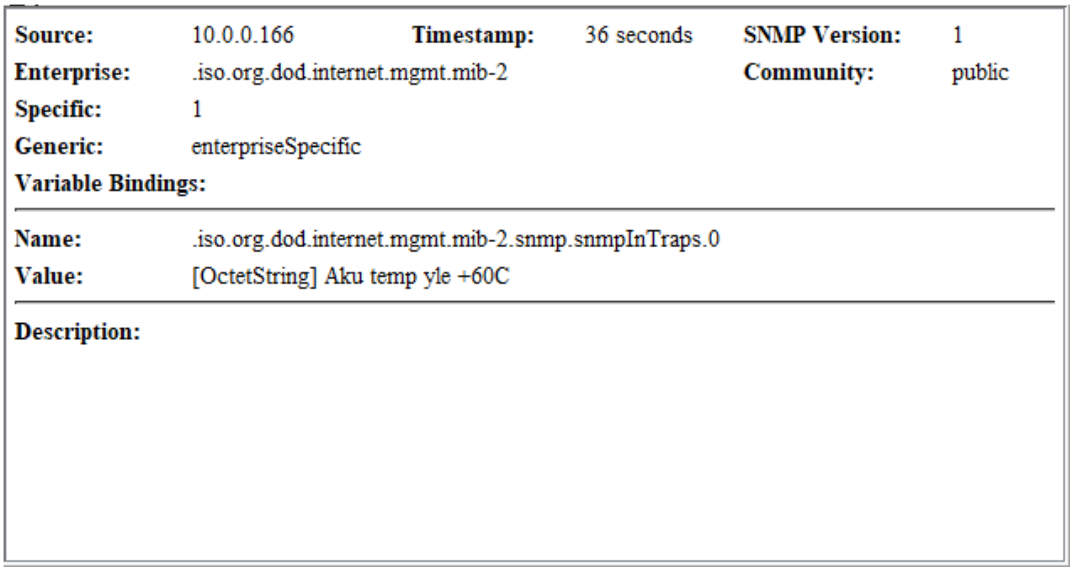
Joonis 15. MIB brauseri kasutajavaade. 3 seireseadmelt vastuvõetud TRAP sõnumit



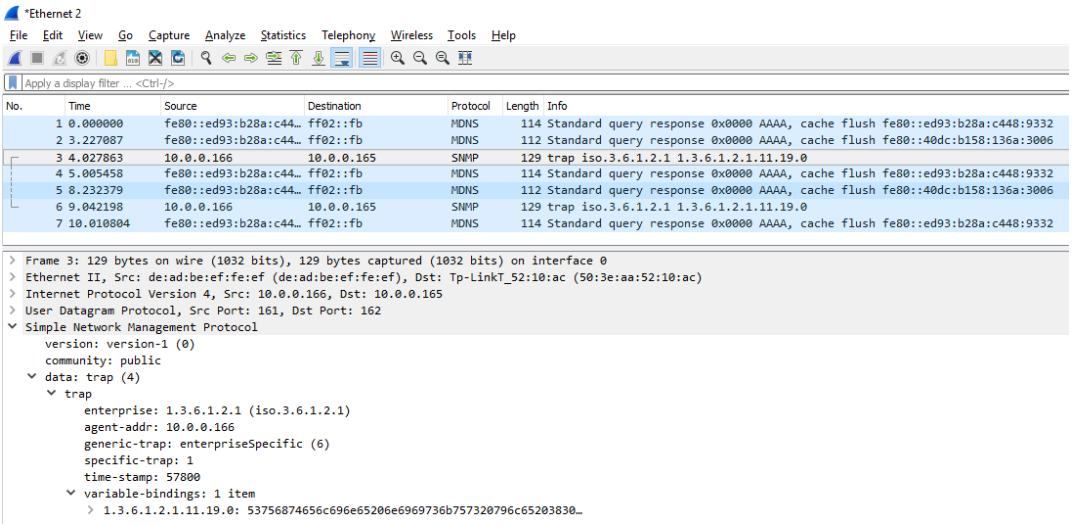
Joonis 16. MIB brauseri kasutajavaade. 7 päringut vastust aku temperatuuri kohta



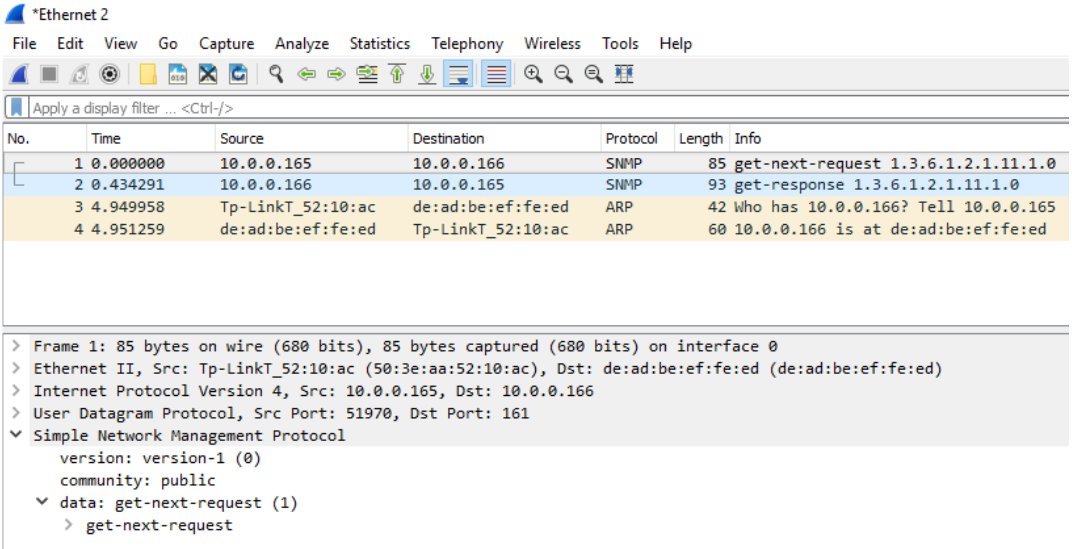
Joonis 17. Vastuvõetud TRAP sõnum „Suhteline õhuniiskus on üle 80%“



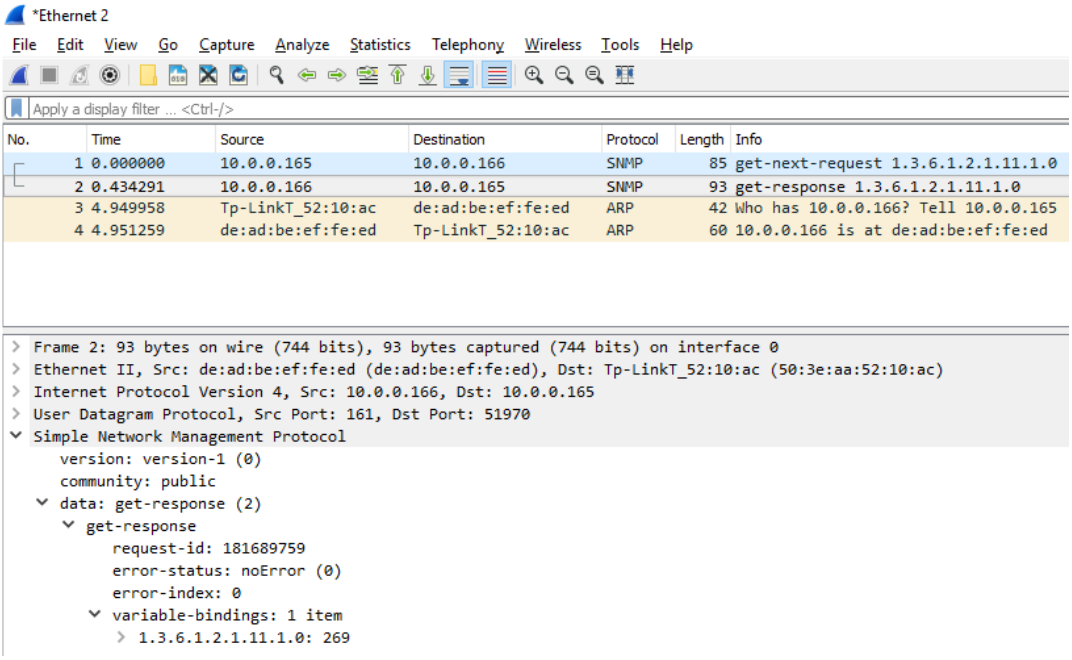
Joonis 18. Vastuvõetud TRAP sõnum „Aku temperatuur on üle +60C“



Joonis 19. Wireshark salvestus TRAP sõnumi saatmisest.Sõnumi sisu on Octetstringina



Joonis 20. Serveri(IP 10.0.0165) GET NEXT päring agendile (IP 10.0.0 166)



Joonis 21. Agendi (IP 10.0.0.166:161) vatsus serverile (IP 10.0.0.165:51970) OID 1.3.6..1.2.1.11.0 väärtus on 269 (10x näit °C) et saaks INT formaadis edastada 0,1 °C täpsusega tempearatuuri.

Lõplikus versioonis tuleks saadetav float väärtus teha stringiks ja saata sõnum octet string formaadis kui on soov saada 0,1°C kraadi täpsusega näit.

6 Kokkuvõte

Bakalaureusetöö eesmärgiks oli luua lihtne ja odav seade, mis võimaldab saada teavet eemal asuva vastuvõtupunkti seadmekilbi toite- ja keskkonnaparameetrite kohta. Ette oli antud SNMP protokoll ja toitepinge pliiaku pealt. Ülesandepüstituses määrati jälgitavad parameetrid ja nendest teavitamise viis.

Töö esimeses pooles tehti valik kontrolleri platvormi osas ja uuriti sobivaid sensoreid. Järgnevalt uuriti sensorite sobitamist kontrolleri ADM sisenditega, võimalikult müravaese ja täpse lugemi saavutamist erinevate skeemilahendustega. Seejärel projekteeriti leitud lahendusele mikrokontrolleri plaadiga sobiv sensorite trükkplaat.

Töö teises pooles uuriti SNMP protokolli ülesehitust ja sõnumite saatmist ning vastuvõttu. Samuti uuriti Arduino platvormile kirjutatud ethernet liidese ja SNMP tarkvara. Kirjutati sensoritelt andmete lugemise ja etteantud lävendeid arvestav SNMP sõnumite saatmise programm, kontrolliti võrguliikluses pakettide edastust ja kasutati MIB brauserit andmete kogumiseks ja päringute tegemiseks.

Tulemite järgi saab väita, et töö resultaadina on välja töötatud kaugseire seadme prototüüp valvesüsteemi vahejaamale. Seadme juurutamisel võib seda soovi korral täiendada näiteks lokaalse displeiga, kus kuvada mõõdetavate parameetrite jooksvad näidud. Samuti võib lisada sama mikrokontrolleri pealt seadmekilbi ventilatsiooni ja kütte juhtimise.

Kasutatud kirjandus

[1] Kurose, James F.
Computer networking : a top-down approach / James F. Kurose, Keith W. Ross.—6th ed.
Copyright © 2013, 2010, 2008, 2005, 2003 by Pearson Education, Inc., ISBN-10: 0-13-285620-4

[2] https://tools.ietf.org/html/rfc3418 (30.04.2019)

[3] https://pen.iana.org/pen/PenApplication.page (30.04.2019)

[4] https://www.iana.org/assignments/enterprise-numbers/enterprise-numbers (30.04.2019)

[5] https://www.sciencedirect.com/science/article/abs/pii/S0378775303009340 (30.04.2019)

[6] https://www.microchip.com/wwwproducts/en/ATmega328P (30.04.2019)

[7] https://www.arduino.cc/en/reference/libraries (30.04.2019)

[8] https://www.gnu.org/licenses/gpl-3.0.en.html (30.04.2019)

[9] https://creativecommons.org/licenses/by-sa/3.0/ (30.04.2019)

[10] https://www.rfc-editor.org/info/rfc1213 (30.04.2019)

[11] http://www.oidview.com/mibs/0/RFC1213-MIB.html (30.04.2019)

[12] https://www.analog.com/media/en/technical-documentation/data-sheets/TMP35_36_37.pdf (30.04.2019)

[13] https://www.pololu.com/file/0J324/HIH-4030-datasheet.pdf (30.04.2019)

[14] https://store.arduino.cc/arduino-uno-rev3 (30.04.2019)

[15] https://github.com/johnyHV/Arduino-NMP/tree/master/Simple%20version/Agent (30.04.2019)

[16] Ivo Müürsepp Side loengumaterjal nr. 11 2018.

[17] http://ireasoning.com/mibbrowser.shtml (30.04.2019)

[18] https://store.arduino.cc/arduino-ethernet-shield-2 (30.04.2019)

https://github.com/rexpark/Arduino-SNMP (30.04.2019)

https://www.arduinolibraries.info/libraries/ethernet2 (30.04.2019)

Lisa 1 Fotod ja skeemid

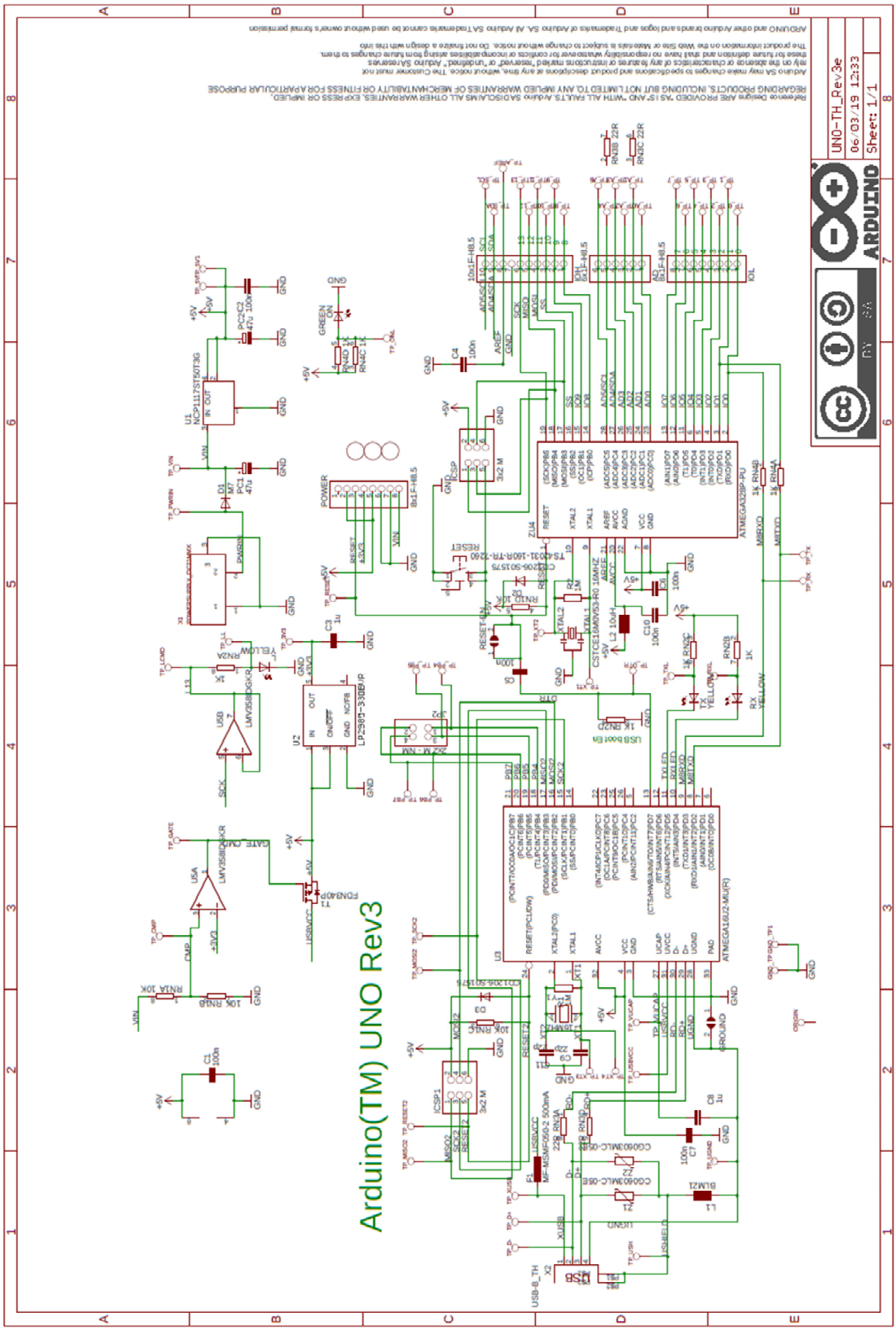
Joonis 22. SATELCODE saatja ja SATELNODE vastuvõtja

Joonis23. RS232 to Ethernet konverter MOXA NPort

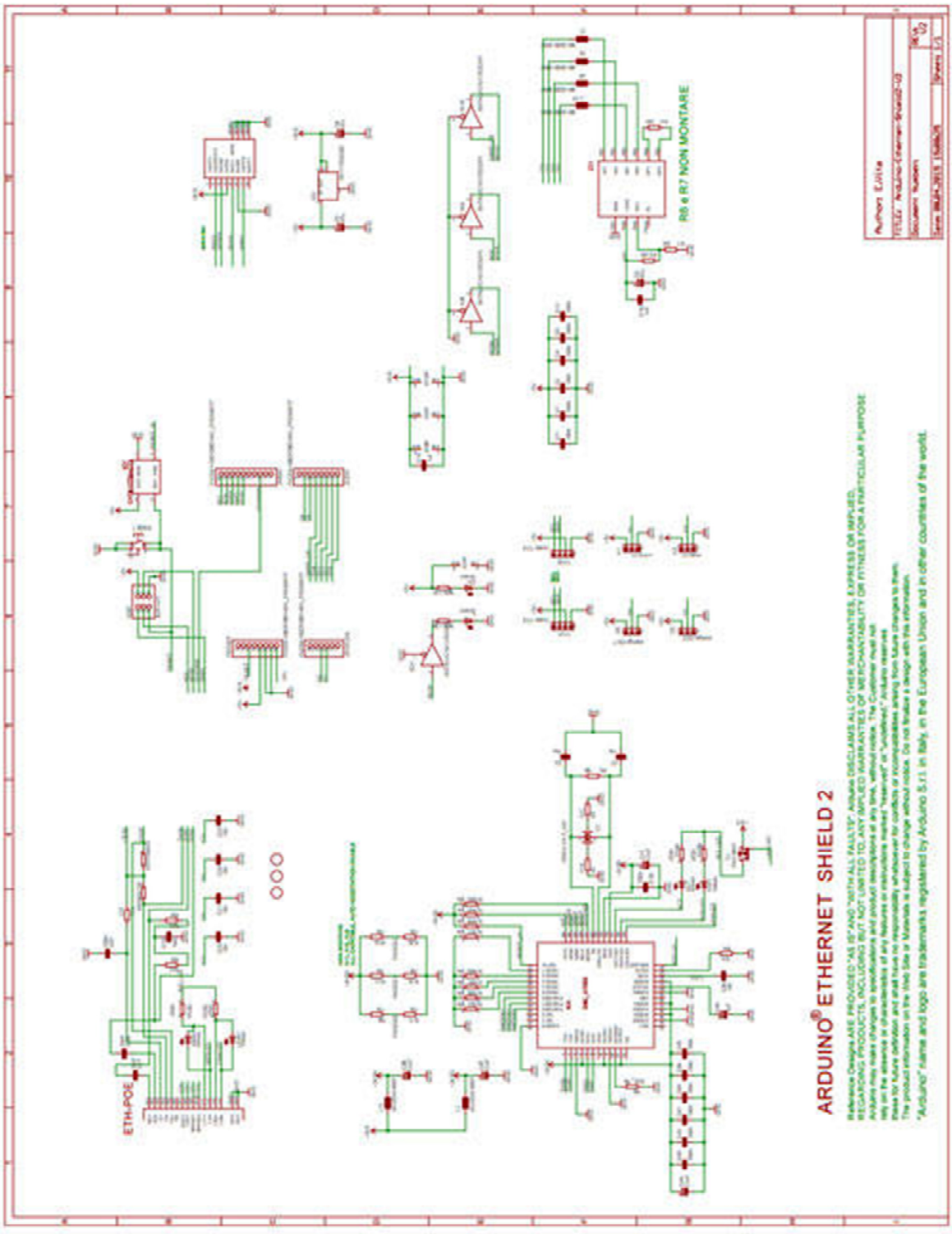
Joonis 24. Rantelon OÜ toodetud spetsiaalne toiteplokk-laadija.

Joonis 25. Vastuvõtupunkt nelja erineva sagedusega vastuvõtjaga.

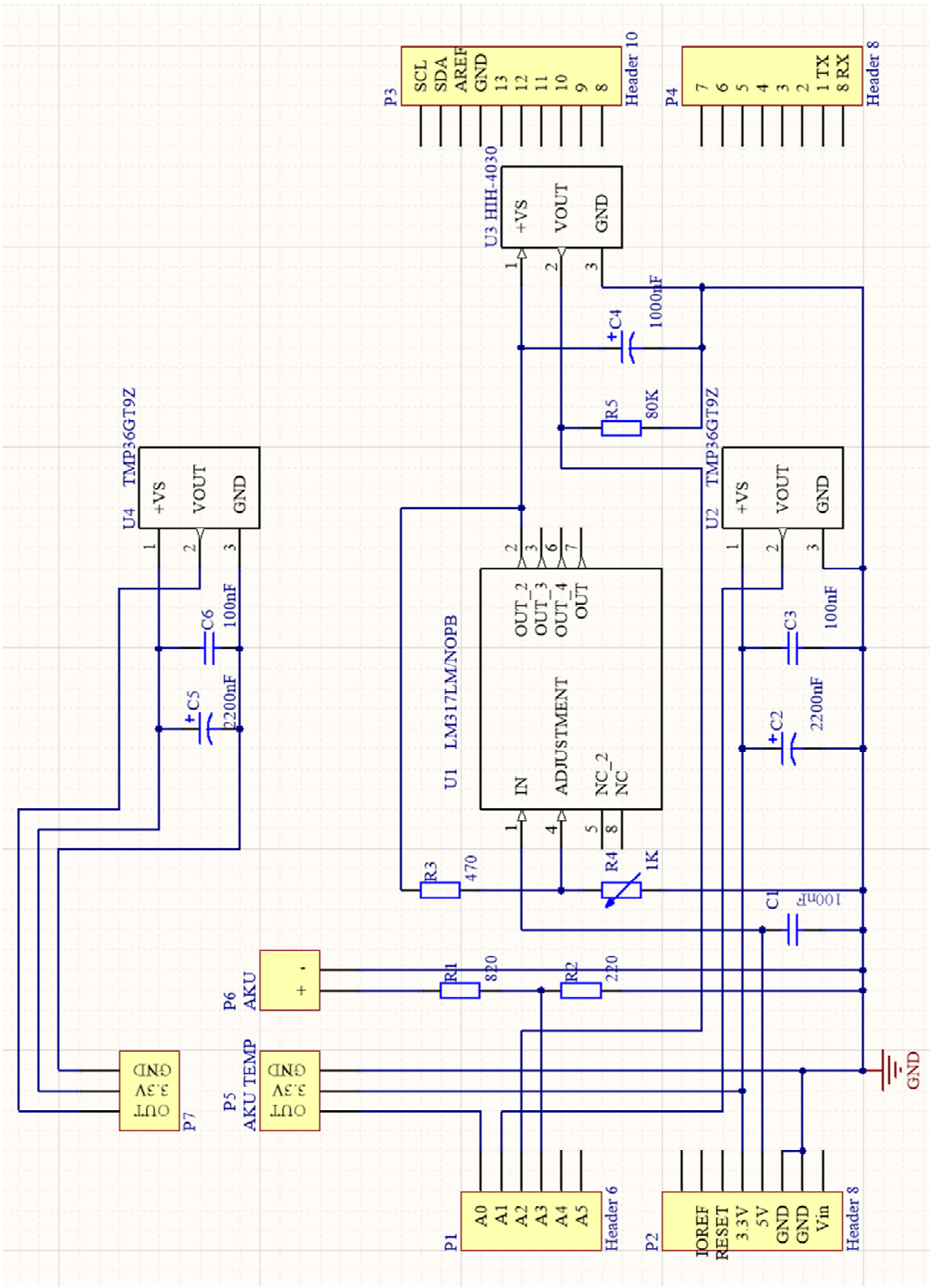
Joonis 26 Termokahjustusega pliiaku



Joonis 27. Arduino Uno rev3 skeem [14]



Joonis 28. Arduino Ethernet shield2 skeem [18]



Joonis 29. Seireseadme sensorite ühendusskeem