Šolski center Celje

Srednja šola za kemijo, elektrotehniko in računalništvo

RF NADZORNI SISTEM

RAZISKOVALNA NALOGA

Avtorja:

Gašper Gril, E-4B

Kristjan Šoln, E-4B

Mentor:

Matej Kališek, inž. el.

Mestna občina Celje, Mladi za Celje

Celje, 2019

Vsebina

KAZALO SLIK

POVZETEK

Glavni cilj te naloge je bil ustvariti sistem, ki kot del pametne inštalacije omogoča brezžično merjenje podatkov kot so temperatura, vlažnost zraka, število trdih delcev ipd. Meritve izvajajo brezžični moduli, nameščeni v neki sobi. Preko radio frekvenčnega modula s frekvenco 433 MHz podatke posredujejo centralnemu modulu, ki jih zbira v SQL bazi in prikaže na spletni strani. Brezžični modul temelji na mikokrmilniku ATMega328p, centralni modul pa je sestavljen iz računalnika Raspberry Pi ter mikrokrmilnika Arduino Mega 2560. Najprej sva določila zahteve oz. hipoteze, nato določila okvirno delovanje sistema, izbrala željene komponente, izdelala vezje ter ga preizkusila. Čeprav na končnem izdelku nisva odpravila vseh težav, sva z končnim izdelkom zadovoljna.

Ključne besede: ATMega328p, pametna inštalacija, brezžično merjenje, SQL baza, Raspberry Pi, Arduino Mega 2560

The main goal of this research paper was to develop a system which could provide wireless temperature, humidity, CO_2 etc. measurements in a smart installation. These measurements are carried out by wireless modules installed in the room. A 433 MHz RF transmitter provides communication with what we call a central module, where data is stored using an SQL database and displayed on a hosted web page. The wireless module mentioned before is based on an ATMega328p microcontroller. The central module consists of an Arduino Mega 2560 microcontroller and a Raspberry Pi. We began by specifying our hypothesis and requirements for the system. Then we created an approximate idea of its functions, selected most appropriate components, developed the software, created a printed circuit board and tested it. Although we did not solve all the issues with the final product, we are satisfied with the result.

Keywords: ATMega328p, smart installation, wireless measurement, SQL database, Raspberry Pi, Arduino Mega 2560

1 UVOD

1.1 PREDSTAVITEV PROBLEMA

Ključni del vsakega sistema za nadzor temperature ali klime v prostoru so senzorji v prostoru. Merijo lahko marsikaj, povezava s centralnim krmilnikom pa je lahko izvedena na mnogo načinov, vsak s svojimi prednostmi in slabostmi. Izvedba je lahko žična, kar je najbolj zanesljivo, vendar je napeljevanje kablov zamudno in dražje. Alternativa je brezžična izvedba, kar pomeni, da lahko senzor namestimo skoraj kamorkoli, a se pojavi problem zanesljive povezave ter napajanja.

V tej nalogi sva skušala narediti senzor, uporaben tudi v praksi, ki bi bil povsem brezžičen, napajan s pomočjo baterij in bi centrali oz. v najinem primeru mikrokrmilniku Arduino Mega 2560 in računalniku Raspberry Pi preko radijskih valov pošiljal podatke, izmerjene z nekim tipalom. V najinem primeru je tipalo DHT22 merilnik temperature in vlage zraka, lahko pa ga nadomestimo z drugimi tipali.

1.2 HIPOTEZE

Na začetku sva postavila naslednje hipoteze:

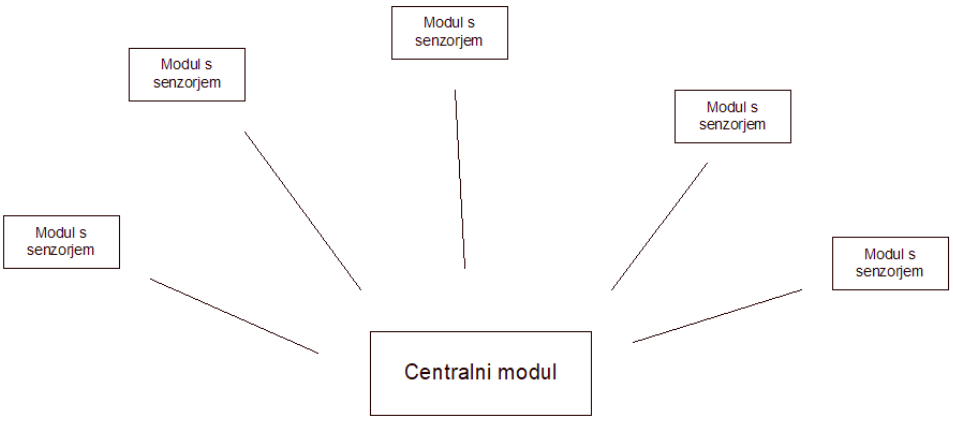
- *Vezje senzorja bo imelo večino komponent v tehnologiji SMD.*
- *Baterije senzorja ne bo potrebno zamenjati vsaj pol leta.*
  Regulator napetosti mora biti dovolj učinkovit, poraba energije pa čim manjša.
- *Sistem bo omogočal razširitev na več senzorjev oz. modulov, ne da bi bilo ob tem potrebno vse preprogramirati.* Brez večjih težav se bo dalo v sistem vključiti dodatne temperaturne in nove tipe senzorjev oz. modulov.
- *Sistem bo mogoče spremljati s spletne strani, ki bo gostovana na računalniku Raspberry Pi.*
- *Računalnik Raspberry Pi bo shranjeval podatke v SQL bazo.*

1.3 PREDSTAVITEV RAZISKOVALNIH METOD

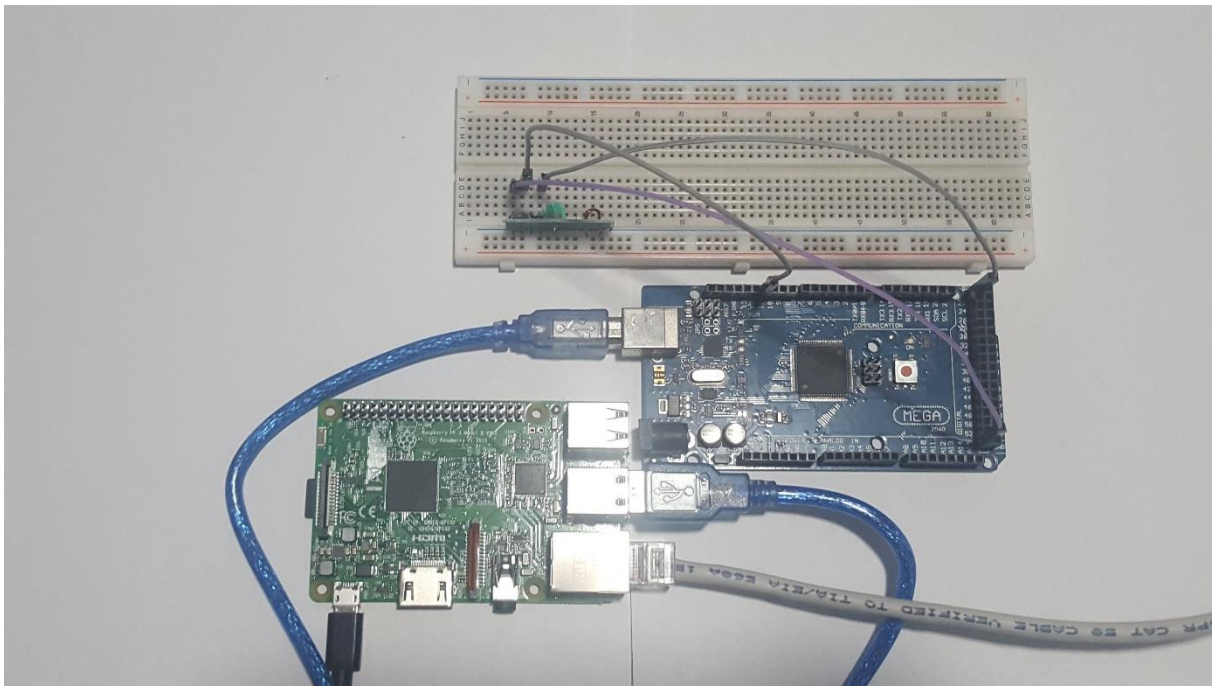
Za izdelavo programa in vezja sva najprej morala preučiti delovanje in izvedljivost posameznih funkcij senzorja, kot je recimo brezžični prenos podatkov z radijskimi valovi. To sva dosegla s pomočjo dokumentacije, poleg tega pa sva marsikateri podatek, predvsem pri porabi energije, morala preprosto izmeriti, saj sva tako dobila najbolj realno sliko o delovanju. Potrebno je bilo preučiti, katere komponente, materiale ipd. imava na razpolago in kako jih povezati v celoto, če je to sploh mogoče. Za simulacijo in načrtovanje napajalnega vezja sva si pomagala z brezplačnim spletnim orodjem Webench Power Designer podjetja Texas Instruments. Program za ATMega328p mikrokontroler sva sestavila v računalniškem programu Arduino IDE (Arduino Integrated Development Environment). Delovanje vezja sva preizkusila na prototipni ploščici (ang. protoboard, breadboard), kasneje pa še na dejanski PCB ploščici.

2 OPIS DELOVANJA SISTEMA

2.1 OSNOVNI KONCEPT DELOVANJA



Slika 1: Blok shema sistema s petimi brezžičnimi moduli.



Slika 2: Centralni modul.

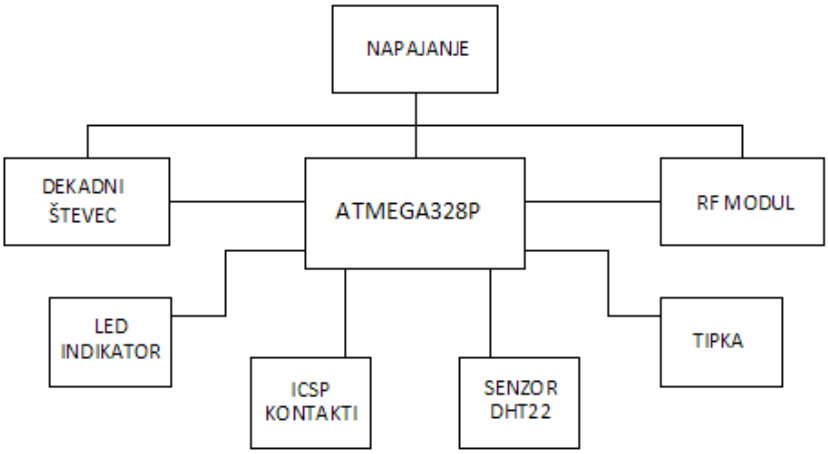
Sistem je sestavljen iz najmanj dveh komponent: centralnega modula in vsaj enega senzorja oz. brezžičnega modula.

- *Centralni modul* sprejema, shranjuje in obdeluje podatke posredovane podatke. V najinem primeru je sestavljen iz treh komponent. Prva je mikrokontroler Arduino Mega 2560. Ta je povezan z modulom za brezžični sprejem podatkov preko radijskih valov (ang. RF receiver). Zadnji je računalnik Raspberry Pi za shranjevanje in obdelavo podatkov.
  Centralni modul bi bil v praksi fiksno nameščen nekje v prostoru, napajan iz električnega omrežja ter povezan z preostalo pametno inštalacijo.
- Naslednja komponenta je *brezžični modul* (v najinem primeru *temperaturni modul* zaradi uporabe DHT22 temperaturnega senzorja). Ta meri željeno količino in jo posreduje centralnemu modulu preko radijskih valov s frekvenco 433 MHz. Sistem je lahko sestavljen iz

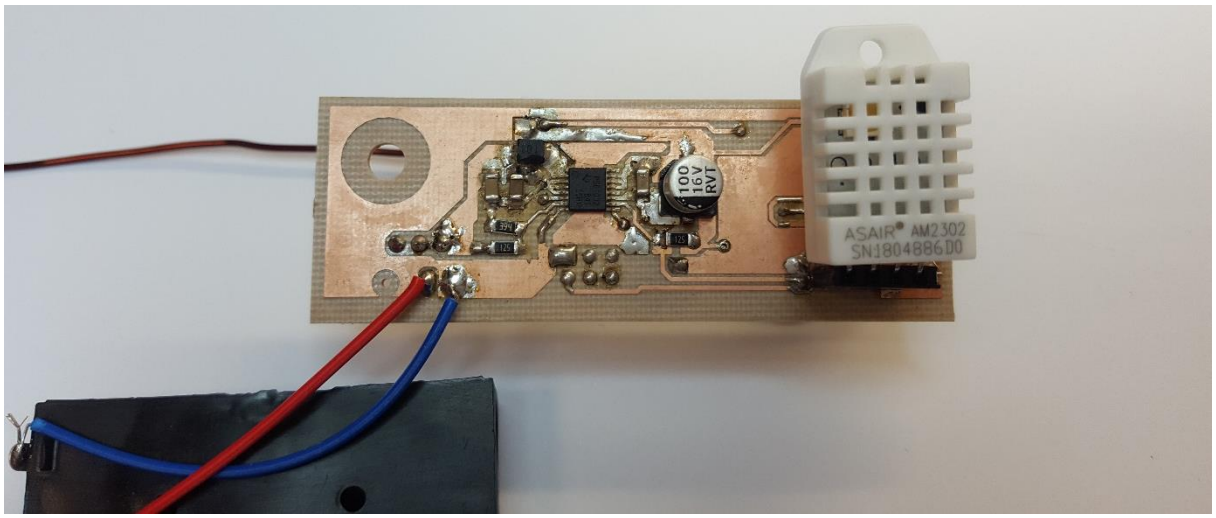
več brezžičnih modulov, ki so lahko nameščeni na poljubnih lokacijah v prostoru. Ti na vsakih nekaj minut preberejo vrednost temperature, vlage, delcev v zraku ipd. in jo pošljejo centralnemu modulu, ki to vrednost zabeleži.

2.2 DELOVANJE TEMPERATURNEGA MODULA

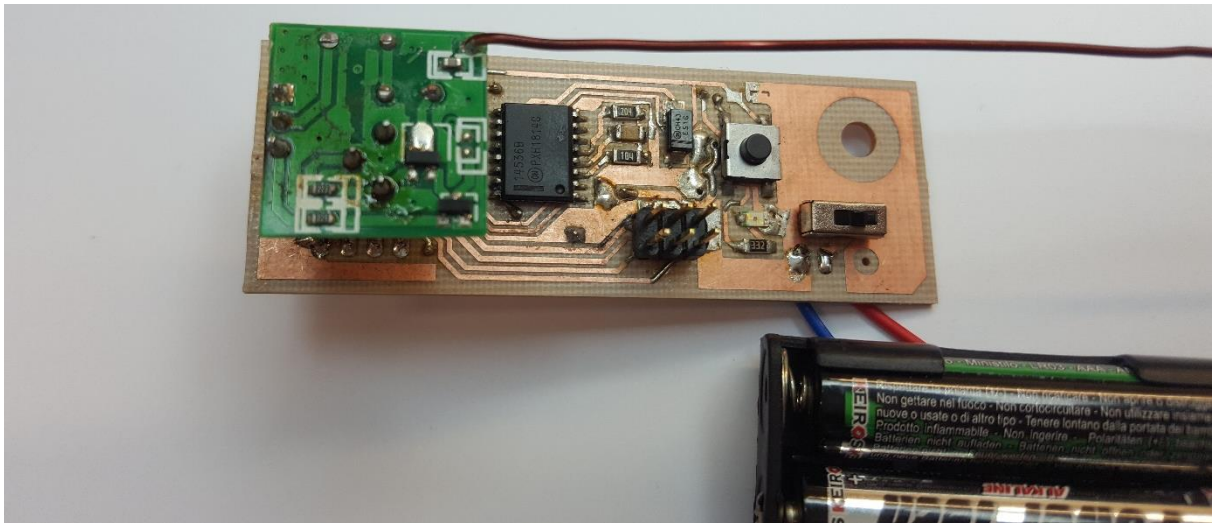
2.2.1 SPLOŠNO



Slika 3: Shema vezja modula.



Slika 4: Spodnja stran vezja.



Slika 5: Zgornja stran vezja.

Samo vezje ima napajalni del, ki obsega dve AAA bateriji, ki zagotavljata 3 V napetosti »boost« regulatorju TPS61032, ki to napetost pretvori na 5 V. Ta je potrebna za napajanje oddajnika radijskih valov (v nadaljevanju RF modul), ki potrebuje vsaj 5 V napajalne napetosti. Ker sva želela čim daljši čas delovanja, sva morala zmanjšati tok skozi vezje na najnižjo možno vrednost. To sva storila tako, da sva mikrokontroler med neuporabo dala v spanje. Napajanje je torej predstavljalo enega od glavnih problemov te naloge.

Poleg napajalnega dela je prisoten tudi RF modul, katerega funkcija je komuniciranje med temperaturnim in centralnim modulom.

Naslednja komponenta je senzor DHT22, ki lahko meri temperaturo in vlago prostora oziroma okolice. Je popularna izbira, saj je eden redkih senzorjev, ki ponuja meritve vlage in temperature hkrati, poleg tega pa je dobavljiv po nizki ceni.

Ključni del modula je mikrokontroler ATMega328p. Lahko bi uporabila razvojno ploščico Arduino, kot je recimo Pro Mini, Nano ali Uno, vendar sva želela uporabiti SMD komponente. Ker sva uporabila vgrajeni 8 MHz oscilator, sva se lahko znebila tudi zunanjega oscilatorja in tako zmanjšala število komponent. Ker sva morala zmanjšati porabo energije na najnižjo možno vrednost, sva se odločila, da bova med 11-minutnim čakanjem med posameznimi branji uporabila način varčevanja z energijo, tj. spanje. Ker pa se ATMega328p ne more po 11 minutah zbuditi sam oziroma vmes meriti, koliko časa je že v načinu spanja, potrebuje zunanji impulz, da ga »zbudi«. Tako sva naletela še na eno težavo, ki sva jo rešila z implementacijo števca MC14536 z vgrajenim oscilatorjem.

MC14536 je astabilni multivibrator s 24 stopnjami oz. dekadami in vgrajenim oscilatorjem. Ta vsakih 11 minut mikrokrmilniku ATMega328p pošlje pulz oz. prekinitev, ki ga zbudi iz spanja.

Poleg navedenih ključnih komponent so na vezju tudi ICSP kontakti za programiranje čipa, LED indikator, ki pove, kdaj krmilnik bere senzor oz. prenaša podatke, ter tipka, s katero lahko predčasno zbudimo mikrokontroler in sprožimo branje senzorja ter prenos podatkov.

2.2.2 NAPAJANJE – MOŽNE REŠITVE IN KONČNA IZBIRA

Za napajanje vezja sva imela sledeče zahteve.

- *Najmanj 23 mA izhodnega toka.*
  Med glavne porabnike na vezju spadajo LED indikator, mikrokontroler, dekadni števec ter RF modul kot največji porabnik. Po meritvah sva prišla do sledečih maksimalnih tokov:
  - LED indikator porabi 1.5 mA;
  - ATMega328p med delovanjem porabi največ 2 mA, med spanjem pa manj kot 10 µA;
  - Števec v povprečju porabi okrog 50 µA;
  - RF modul med prenašanjem porabi največ 15 mA, med nedelovanjem pa okrog 0.7 mA.

  Če to seštejemo in dodamo 25% tolerance, dobimo sledeč minimalni izhodni tok regulatorja:
  $$(1.5\ mA + 2\ mA + 50\ \mu A + 15\ mA) \times 1.25\ \cong\ 23\ mA$$
  To je konična obremenitev, ki je prisotna le v času pošiljanja podatkov preko RF modula, to pomeni maksimalno 1 sekundo. Med intervali spanja, ko mikrokontroler čaka na naslednjo meritev, je tok iz regulatorja sledeč:
  $$0\ mA\ (LED) + 10\ \mu A\ (Atmega328p) + 50\ \mu A\ (\check{s}tevec) + 0\ \mu A\ (RF\ modul) = 60\ \mu A$$
  To pomeni, da je med spanjem poraba v vezju zelo majhna.
- *Tok, ki teče skozi regulator neodvisno od obremenitve, mora biti zelo majhen (ang. quiescent current, I_Q).*
  Regulator, ki sva ga izbrala, ima I_Q okoli 20 µA, kar je zelo malo.
- *Regulator mora biti na voljo v SMD izvedbi.*
  Te zahteve ni bilo težko izpolniti, saj je veliko regulatorjev na voljo le v SMD tehniki.
- *Zaželena je vgrajena nadtokovna zaščita.*
- *Prioriteto imajo napajalniki manjših dimenzij.*
  To vključuje baterijo in njeno držalo.
- *Regulator mora biti hitro dobavljiv.*
- *Regulator mora biti učinkovit.*

Po določitvi pogojev in raziskovanju obstoječih rešitev sva prišla do treh različnih rešitev. Za primerjavo sva uporabljala načrtovalec napajalnih vezjih Webench Power Designer podjetja Texas Instruments, ki je brezplačno spletno orodje za načrtovanje in primerjavo napajalnih vezij. Poleg okvirnega načrtovanja vezja omogoča tudi simulacijo in primerjavo z drugimi regulatorji. Izračuna tudi mnoge parametre, ki so naju zanimali, kot je recimo učinkovitost pretvorbe, frekvenca integriranega vezja, vrednost posameznih komponent, kot je npr. induktivnost tuljave, upornost uporov ipd. Izračuni in sheme so seveda le informativne in jih je potrebno preveriti v praksi oziroma jih primerjati s podatki v dokumentaciji čipa, a nama je to orodje močno olajšalo izbiranje pravega regulatorja.

- *9V baterija in »stepdown« regulator LM2596S-5.*
  Ta izbira je sestavljena iz 9 V baterije in stikalnega regulatorja, ki bi napetost 9 V pretvoril na napetost 5 V. Je tudi dimenzijsko največja in najmanj primerna opcija, saj je 9 V baterija zelo velika. Glede na orodje Webench Power Designer je učinkovitost takega regulatorja le 38.9 %. Pozitivno je to, da ima 3 A maksimalnega izhodnega toka, nadtokovno in termalno zaščito, je hitro in lokalno dobavljiv ter je na voljo v SMD izvedbi.
  Nizka učinkovitost in ne preveč velika kapaciteta 9 V baterije (v povprečju okoli 550 mAh) bi pomenila, da bi se baterija relativno hitro izpraznila, zato to ni najbolj elegantna in zanesljiva rešitev.
- *Dve vzporedno vezani 3 V gumbni bateriji CR2032 in »stepup« regulator TPS61032-5.*

Ta izbira je sestavljena iz dveh vzporedno vezanih CR2032 baterij in regulatorja, ki pretvori 2 do 3 V vhodne napetosti na 5 V izhodne napetosti. Je dimenzijsko najmanjša opcija, a ima problem z največjim maksimalnim izhodnim tokom baterije. Učinkovitost glede na Webench orodje je kar 93% in zadosti vsem prej zastavljenim pogojem, razen prvemu. Glede na dokumentacijo o CR2032 gumbni bateriji sva ugotovila, da je maksimalni konični izhodni tok ene baterije 15 mA. Če sta dve vezani vzporedno, je to 30 mA, kar še vedno ni dovolj, kajti glede na simulacijo regulator potrebuje za 23 mA koničnega izhodnega toka okoli 50 mA vhodnega toka.
Torej, čeprav je ta izbira dimenzijsko najbolj prikladna, je najmanj zanesljiva.

- *Dve zaporedno vezani AAA 1.5V bateriji in »stepup« regulator TPS61032-5.*
  Ta možnost je podobna prejšnji, le da sva namesto gumbnih baterij zaporedno vezala dve AAA bateriji, kar ustvari okoli 3 V vhodne napetosti za regulator. V nasprotju z gumbnimi baterijami imajo AAA baterije glede na dokumentacijo veliko večji konični tok in kapaciteto, 860 do 1200 mAh za alkalne baterije. To pomeni, da zadostijo vsem kriterijem in so dokaj zanesljiva in prikladna rešitev.

2.2.3 BUJENJE MIKROKRMILNIKA

Mikrokrmilnik ATMega328p v polnem delovanju za delovanje porabi okoli 2 mA. Ker pa v polnem delovanju deluje tudi veliko sistemov, ki jih ne potrebujeva kot so analogno-digitalni pretvornik, serijska komunikacija in Timer/Counter 2, sva jih v registru PRR (Power Reduction Register) izklopila. Ker sva si zadala cilj, da bi se podatki osveževali na približno 11 minut, sva celoten mikrokrmilnik dala v stanje Power-down, v katerem se izključi vse, razen oscilatorja procesorja, Watchdog sistema in prekinitvenega sistema. V tem načinu mikrokrmilnik porabi le 0,7 mikro Ampera. Ker pa ga je iz tega načina mogoče prebuditi le z prekinitvijo, sva najprej želela uporabiti vgrajen »watchdog« časovnik za bujenje, a šteje le do 8 sekund, kar je premalo. Zato sva morala uporabiti zunanji števec.

Pri izbiri števca sva imela na voljo veliko različnih tipov. Sprva sva želela izbrati RTC (Real Time Clock), ki bi omogočal »time of the day« alarme, kar pomeni, da vsak dan ob določenem uri postavi izhod na visoko stanje in zbudi mikrokrmilnik. Pri teh je bila velikokrat težava napetost, saj jih velika večina uporablja 3.3 V in ne 5 V kot najin modul. Poleg tega pa jih velika večina uporablja I^2C protokol, s katerim pa nimava izkušenj. Pri izbiri sva morala biti tudi seveda pozorna tudi na porabo energije. Nisva želela, da bi števec porabljal več energije kot mikrokrmilnik, ko ni v načinu spanja. Zato, sva izbrala števec MC14536B, ki ga konfiguriramo z vhodi A, B, C ter D, ki jih postavimo na visoko oz. nizko logično stanje glede na pravilnostno tabelo. Tako določimo čas zakasnitve. Konfigurirala sva ga tako, da na vsakih 11 minut postavi INT0 na visoko stanje, kar zbudi mikrokrmilnik. Ko je buden, prebere vrednosti senzorja za temperaturo in vlažnost zraka in ju pošlje centralnemu modulu preko RF modula, nato gre nazaj v spanje za nadaljnjih 11 minut. S tem pristopom prihraniva veliko energije, kar poveča čas delovanja senzorja.

2.3 OPIS DELOVANJA CENTRALNEGA MODULA

Ker sva si zadala cilj zbirati podatke z vseh modulov in jih shranjevati na mikroračunalniku Raspberry Pi, sva morala uporabiti RF sprejemnik. Ker pa knjižnica »Virtual wire«, ki je potrebna za kodiranje in pošiljanje podatkov preko RF modulov, ne deluje na mikroračunalniku Raspberry Pi, sva morala dodati vmesni člen – mikrokontroler Arduino Mega2560, ki prej omenjeno knjižnico podpira.

Arduino tako sporočilo oz. poslano kodo sprejme, nato pa jo pošlje računalniku Raspberry Pi preko serijskega vmesnika.

Ta kodo razdeli na dele ki so ločeni z » ; «. Prvi del je ID tipa, na osnovi tega računalnik Raspberry Pi ve, katere tipe vrednosti (int, float ali string) pričakovati. Drugi del je ID modula, na osnovi katere Raspberry Pi izbere SQL bazo za shranjevanje podatkov. Nato sledijo izmerjene vrednosti - v primeru temperaturnega modula sledi temperatura ter relativna vlažnost zraka v procentih. Temperatura se shrani kot spremenljivka tipa float, vlažnost pa kot spremenljivka tipa integer.

Te vrednosti v SQL bazi so nato na voljo za prikaz na spletni strani, ki je prav tako gostovana na mikroračunalniku.

Za dodajanje novih tipov senzorjev kot na primer modul za merjenje hitrosti vetra ter senzor dežja, je treba dodati kratek del kode v program, ki sprejema kode, dodati novo podatkovno bazo ter tabelo, ter posodobiti spletno stran, kar ni preveč zapleteno in zamudno.

2.4 ZNAČILNOSTI PROGRAMSKEGA DELA SISTEMA

2.4.1 SISTEM KOD

Za prenos podatkov sva si izmislila v prejšnji točki omenjen sistem kod. Ta nama omogoča prenašanje informacij, kot so ID modula ter podatki, ki jih modul meri, v kratkem sporočilu, kar zmanjša možnost korupcije med prenosom ter čas prenosa.

Vsakemu modulu pripadata dve dvomestni identifikacijski števili. Prvo število predstavlja tip modula, drugo število pa je unikaten ID, s katerim specifično določimo, kateri modul pošilja kodo.

Če je v paru identifikacijskih števil prvo število »01«, to predstavlja senzor temperature, če je to število enako »02«, pa lahko predstavlja npr. modul za merjenje trdih delcev v zraku ipd. Če imamo več modulov istega tipa (z istim prvim številom), jih med sabo ločimo po drugem številu, ki je unikatno za vsak modul posameznega tipa. To pomeni, da lahko sistem podpira 99 različnih tipov modulov in 100 modulov vsakega tipa.

Vsak modul, ki podatke oddaja, pošilja kode, ki se začnejo z '!', da se lažje razbere, kje se koda začne, v primeru, da jih pride več hkrati. Nato sledita ID številki ter podatki, ki jih želi poslati. Pošlje jih v šestnajstiškem zapisu, zato jih je potrebno pretvoriti v decimalni zapis. Za temperaturni modul:

!ID-tip; ID-modul; Temperatura; Vlaga;

Oziroma konkretno:

!01; 00; 21.5; 69;

Torej modul s prvo identifikacijsko številko 01 (temperaturni modul) in z drugo identifikacijsko številko 00 (prvi modul tega tipa v sistemu), pošilja podatek, da je izmeril temperaturo 21.5^{o}C ter 69 % vlažnostjo.

Tako je sistem fleksibilen in razširljiv na nove tipe modulov.

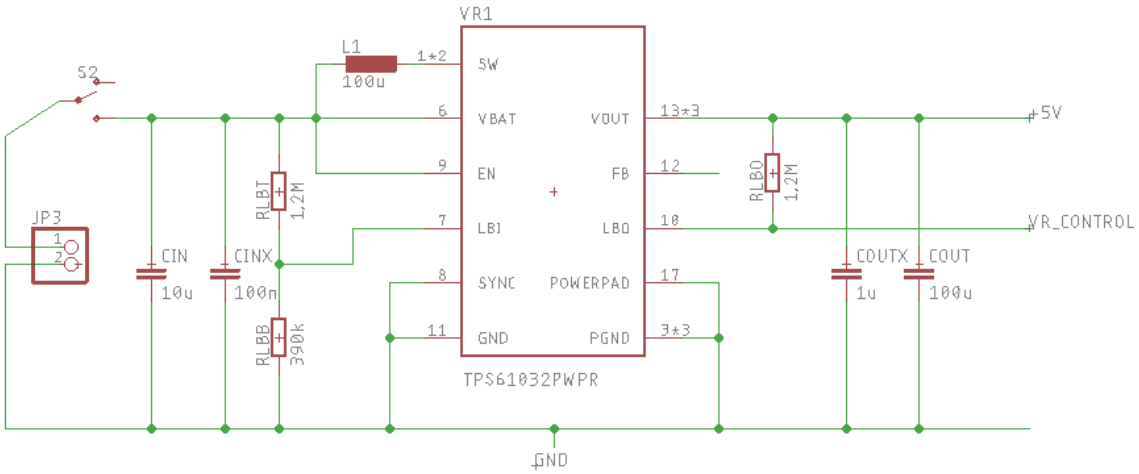
2.4.2 KONFIGURACIJA BREZŽIČNEGA MODULA

Kot je opisano že v sistemu kod, je pri nalaganju programa potrebno vsak modul konfigurirati z dvema ID številkama.

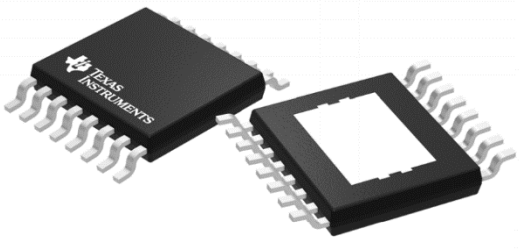
Vsi tipi modulov delujejo po istem principu: večino časa so v načinu spanja in tako varčujejo z energijo. Nato se ob preteku določenega časa prebudijo iz načina spanja, s pomočjo senzorjev izmerijo željeno količino, jih kodirajo po prej omenjenem sistemu, pošljejo preko RF oddajnika in se vrnejo nazaj v način spanja.

3 OPIS KOMPONENT

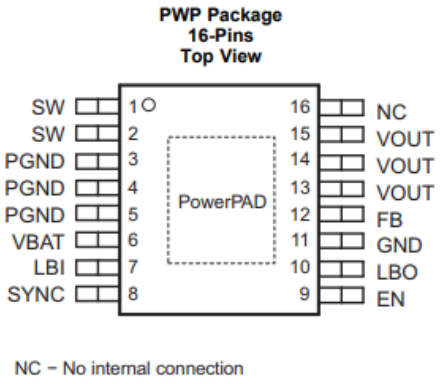
3.1 REGULATOR NAPETOSTI TPS61032PWP



Slika 6: Shema vezja regulatorja napetosti.



Slika 7: Regulator TPS61032 PWPR - 3D model. Vir: http://www.ti.com/lit/ds/symlink/tps61032.pdf



Slika 8: TPS61032 PWPR regulator – imena nogic. Vir: http://www.ti.com/lit/ds/symlink/tps61032.pdf

TPS61032PWP je »boost« stikalni regulator, ki je namenjen za sisteme, ki jih napaja NiCd ali NiMH baterija z dvema ali tremi celicami, Li-Ion baterijo z eno celico ali, kot v najinem primeru, alkalne baterije z dvema ali tremi celicami.

Vhodna napetost je med 1.8 in 5.5 V, izhodna napetost je 5 V, valovanje izhodne napetosti pa je največ ±3 % izhodne napetosti, kar pomeni ±150 mV. Maksimalni izhodni tok je ob zadostni termalni regulaciji 1 A, stikalni regulator pa obratuje pri frekvenci 600 kHz.

Tipičen tok, ki teče skozi regulator neodvisno od obremenitve (ang. quiescent current, I_Q), je 20 µA.

Med pozitivnim polom baterije in regulacijskim vezjem je stikalo, s katerim se vezje vklopi in izklopi. To je hkrati tudi način resetiranja mikrokontrolerja.

3.1.1 Izbira vrednosti tuljave

Stikalni regulatorji zaradi svojega principa delovanja potrebujejo tuljavo, ki shrani energijo med posameznimi pulzi regulatorja.

V dokumentaciji je za izračun potrebne induktivnosti navedena naslednja enačba:

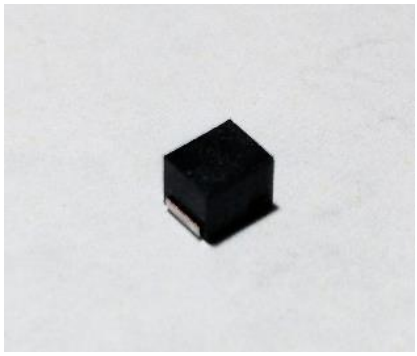
$$L = \frac{Ubat \times (Uizh - Ubat)}{\Delta Il \times f \times Uizh}$$

Pri tem je ΔIl tok, ki ga povzroča popačenje napetosti v tuljavi (ang. ripple current), f pa frekvenca regulatorja. Glede na simulacijo v orodju Webench je $\Delta Il = 19.23\ mA$ in $f = 600\ kHz$. Predpostavili bomo nominalno napetost baterije 2.8 V.

$$L = \frac{2.8\ V \times (5\ V - 2.8\ V)}{0.01923\ A \times 600000\ Hz \times 5\ V}$$

$$L = 106.77\ \mu H$$

Kot najbližjo standardno vrednost sva vzela 100 µH, tuljava pa ima nazivni tok 100 mA, kar je glede na simulacijo v orodju Webench dovolj.



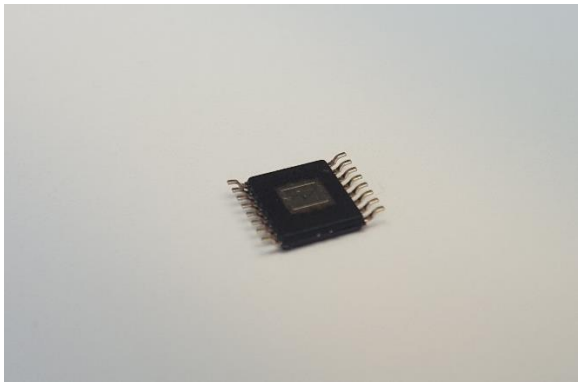
Slika 9: Tuljava za stikalni regulator.

3.1.2 Funkcija varčevanja z energijo

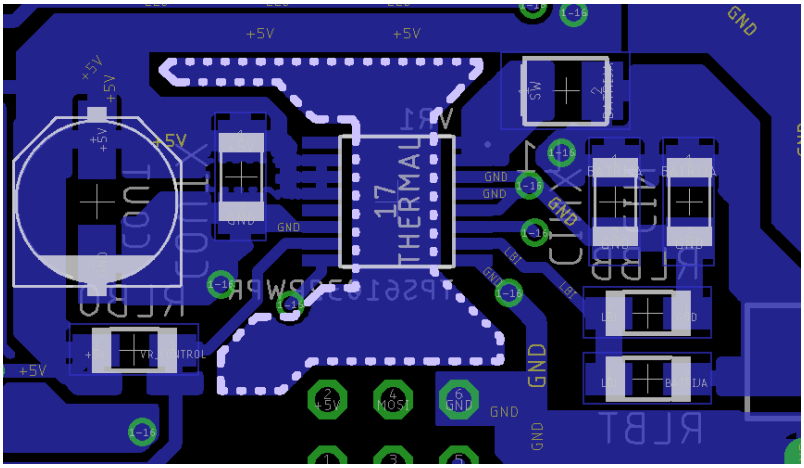
Z nogico SYNC lahko izbiramo med različnimi načini delovanja. S tem ko je priključena na maso, je izbran način varčevanja z energijo. Ta način poveča učinkovitost pretvornika pri manjših obremenitvah, kar velja tudi za to vezje. Regulator v tem načinu delovanja odda le nekaj pulzov in se izključi, ko napetost na izhodu pride do neke mejne vrednosti. Potem počaka, da napetost zaradi bremenskega toka pade pod določeno mejno vrednost in šele takrat spet odda nekaj pulzov. Tako se pri nižjih tokovih izboljša učinkovitost same pretvorbe.

3.1.3 Hlajenje čipa

Čip je TSSOP PowerPad (PWP) izvedbe, kar pomeni, da ima na spodnji strani kontaktno površino za lažje hlajenje. V dokumentaciji je navedena maksimalna toplotna disipacija okoli 1 W. Ker bo čip zagotavljal tok, ki je v najslabšem primeru 2.3 % maksimalnega toka regulatorja, dodatno hlajenje ni bilo potrebno. Vseeno sva ob načrtovanju vezja predvidela poligon, ki se bo stikal s kontaktno površino za hlajenje in tako omogočal večjo toplotno stabilnost regulatorja (ang. thermal plane).



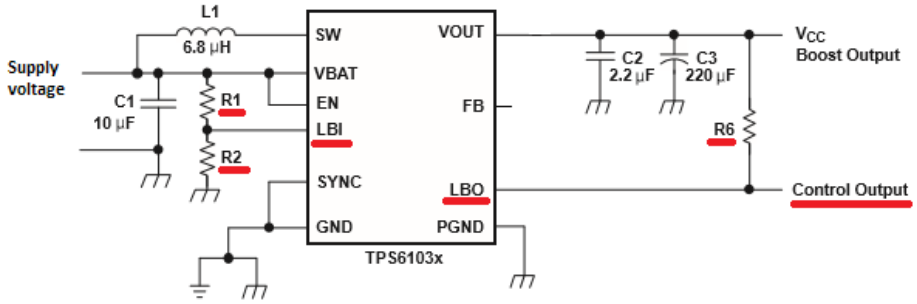
Slika 10: Spodnja stran regulatorja, kjer je vidna kontaktna površina za hlajenje.



Slika 11: Del vezja z regulatorjem v programu Autodesk Eagle. Izbran je poligon, predviden za hlajenje čipa.

3.1.4 LBO in LBI kontakti ter kontroliranje napetosti vira

Regulator ima implementirano tudi vezje za kontroliranje napetosti vira, v tem primeru napetosti dveh zaporedno vezanih AAA baterij. Ko vhodna napetost pade pod nastavljeno vrednost, se generira opozorilo, ki jo posreduje mikrokontrolerju. Za to se uporabljata kontakta LBO in LBI.



Slika 12: TPS61032 PWPR regulator – Primer kontroliranja napetosti vira. Vir: http://www.ti.com/lit/ds/symlink/tps61032.pdf

Opozorilo se generira, ko napetost med GND in LBI kontaktom pade pod 500 mV. Z delilnikom napetosti (na zgornji sliki je to R_1 in R_2) lahko nastavimo, pri kateri napetosti bo regulator generiral opozorilo nizke napetosti baterije. Z uporabo vrednosti R_1 = 1.2 MΩ in R_2 = 390 kΩ pri predvideni minimalni vhodni napetosti 2 V (ko napetost dveh zaporedno vezanih AAA baterij pade iz 3 V nominalne napetosti pod 2 V, kar sva določila kot minimalno napetost baterij) bo napetost U_{R2} = 490 mV, kar generira opozorilo. Ob vhodni napetosti 3 V je U_{R2} = 736 mV, kar je več kot 500 mV, kar ne povzroči generacije opozorila.

Opozorilo se generira na LBO nogici regulatorja. Ko opozorilo ni aktivno, ima kontakt LBO visoko impedanco, kar povzroči, da je napetost, na zgornji sliki označena kot »control output«, enaka izhodni napetosti regulatorja, na zgornji sliki označeni kot »V_{CC}«. Ko se generira opozorilo, se na LBO kontaktu pojavi nizka impedanca in izhod se postavi v nizko stanje (ang. drain), kar sklene R_6 z GND. To povzroči, da »control output« napetost pade iz V_{CC} na 0 V.

LBO kontakt je povezan z mikrokontrolerjem, ki lahko tako preverja, kdaj napetost baterije pade pod 2 V. Ker je minimalna vhodna napetost za delovanje regulatorja okoli 1.6 V (funkcija »undervoltage lockout« izklopi regulator, ko vhodna napetost pade pod 1.6 V), mikrokrmilnik pa uporabnika opozori že, ko je napetost vira 2 V, ima uporabnik še dovolj časa, da vidi opozorilo in menja baterije v modulu, preden modul preneha delovati in meriti temperaturo.

Zgornja slika je le primer, naveden v dokumentaciji regulatorja in ni shema, ki sva jo dejansko uporabila.

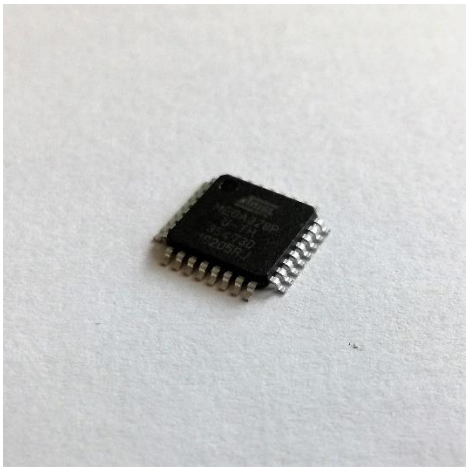
3.1.5 Ostale funkcije regulatorja

Regulator ima implementiran izklop ob prenizki vhodni baterijski napetosti. Meja je okoli 1.6 V. Ko pade napetost pod to mejo, se regulator postavi v »shutdown mode«. V tem načinu delovanja regulator preneha preklapljati s 600 kHz, celotno integrirano krmilno vezje se izklopi in breme odklopi od vhodne napetosti.

Obstaja še ena funkcija regulatorja, in sicer mehek zagon (ang. soft start). Preprosto povedano, to pomeni, da ko se regulator priključi na napetost ali pa preklopi iz »shutdown mode« načina v običajen način delovanja, se izhodni tok omeji na 40 % nominalne vrednosti, da se izhodni kondenzatorji regulatorja ne napolnijo s kratkostičnim tokom. Ko so polni, se omejitev izhodnega toka izključi.

Regulator naj bi, glede na dokumentacijo, imel zmanjšane elektromagnetne vplive na ostale komponente, vendar je to tudi odvisno od razporeditve elementov na plošči in širine ter dolžine povezav. Pomembna je še ena funkcija regulatorja, in sicer izklop zaradi previsoke temperature regulatorja.

3.2 ATMEGA328P MIKROKONTROLER



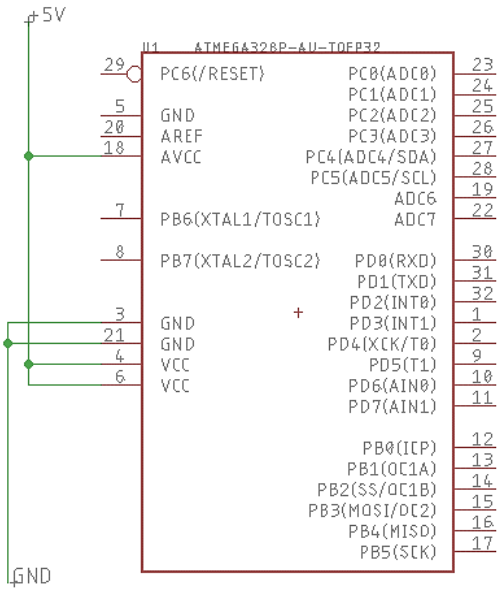
Slika 13: ATMega328p v TQFP izvedbi.

ATMega328p je v najinem vezju nadomestil razvojno ploščico Arduino Nano, katere ključni del je prav tako ta mikrokontroler. S uporabo mikrokontrolerja izvedbe TQFP namesto celotne razvojne ploščice sva zmanjšala velikost ploščice.

Sam čip je 8-bitni AVR mikrokontroler TQFP izvedbe podjetja Atmel oz. Microchip za novejše serije. Ima 32kB FLASH pomnilnika, 1kB EEPROM pomnilnika, 2kB SRAM pomnilnika, 23 splošnih vhodov in

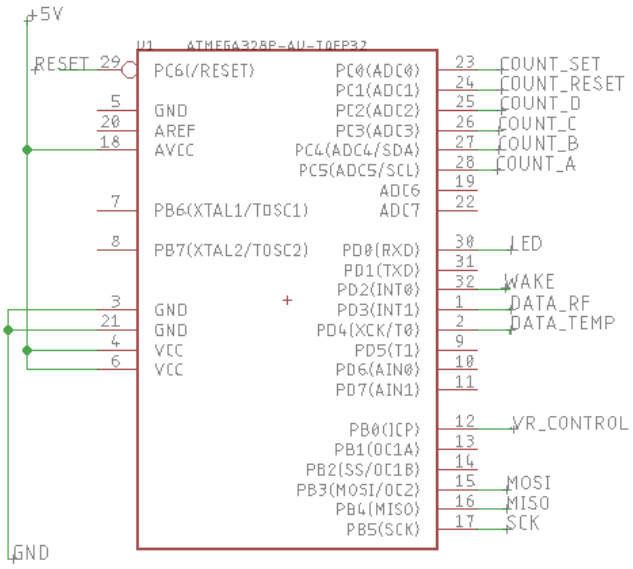
izhodov, en 16-bitni in dva 8-bitna timerja s komparatorji, zunanje in notranje prekinitve, watchdog timer, vmesnik za serijsko povezavo, 6-kanalni 10-bitni ADC pretvornik in 5 različnih načinov spanja. Deluje lahko med 1.8 V in 5.5 V.

Nanj lahko priključimo zunanji oscilator frekvence do 16 MHz ali pa uporabimo vgrajen oscilator. Izbrala sva drugo možnost, saj sva tako zmanjšala število komponent. Izbrala sva najvišjo možno frekvenco vgrajenega oscilatorja, to je 8 MHz. To je pomembno pri nastavljanju prekinitev in »timerjev«.



Slika 14: ATMega328p v najosnovnejši vezavi.

Na zgornji sliki je osnovna vezava, iz katere sva izhajala. Postopoma so bile med razvijanjem vezja dodane še druge povezave. Na spodnji sliki je dejanska in končna shema ATMega328p mikrokrmilnika.

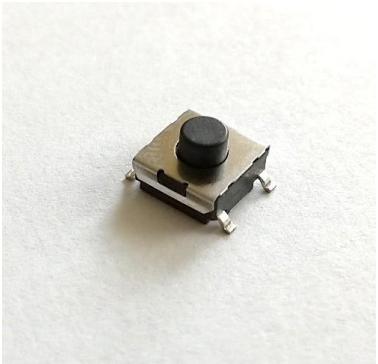


Slika 15: ATMega328p v dejanski vezavi.

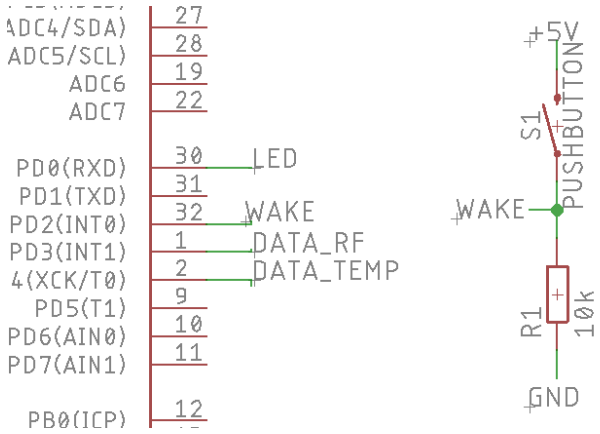
3.3 TIPKA

Zaradi lažjega testiranja sva dodala tipko, povezano na INT0, tako kot izhod števca MC14536B. Tako ni več potrebno čakati 11 minut na bujenje mikrokontrolerja, ampak se ga lahko zbudi s pritiskom na tipko.

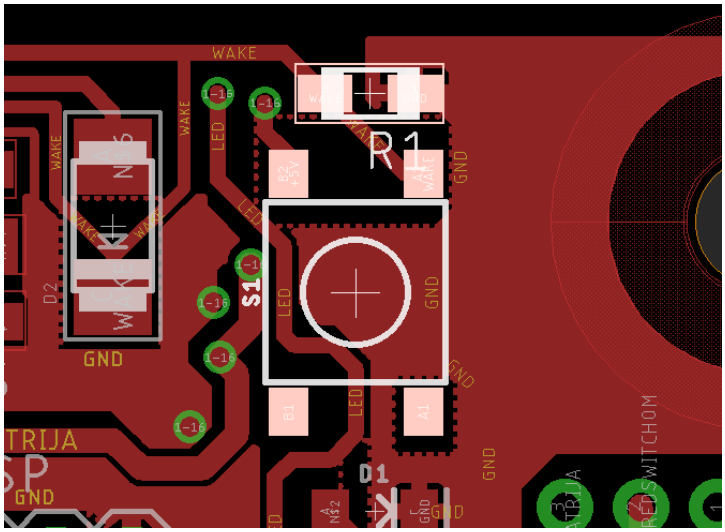
Na INT0 je na shemi priključen tudi »pulldown« upor, ki sva ga kasneje izpustila, ker je povzročal težave pri delovanju števca MC14536B.



Slika 16: Tipka za bujenje mikrokrmilnika.

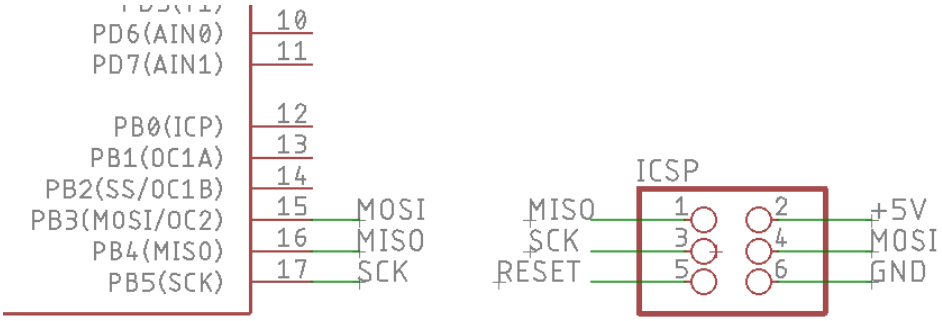


Slika 17: Tipka za bujenje mikrokrmilnika – shema vezja.



Slika 18: Tipka za bujenje mikrokrmilnika - izvedba na PCB ploščici. Tipka in upor sta označena.

3.4 ICSP KONEKTOR

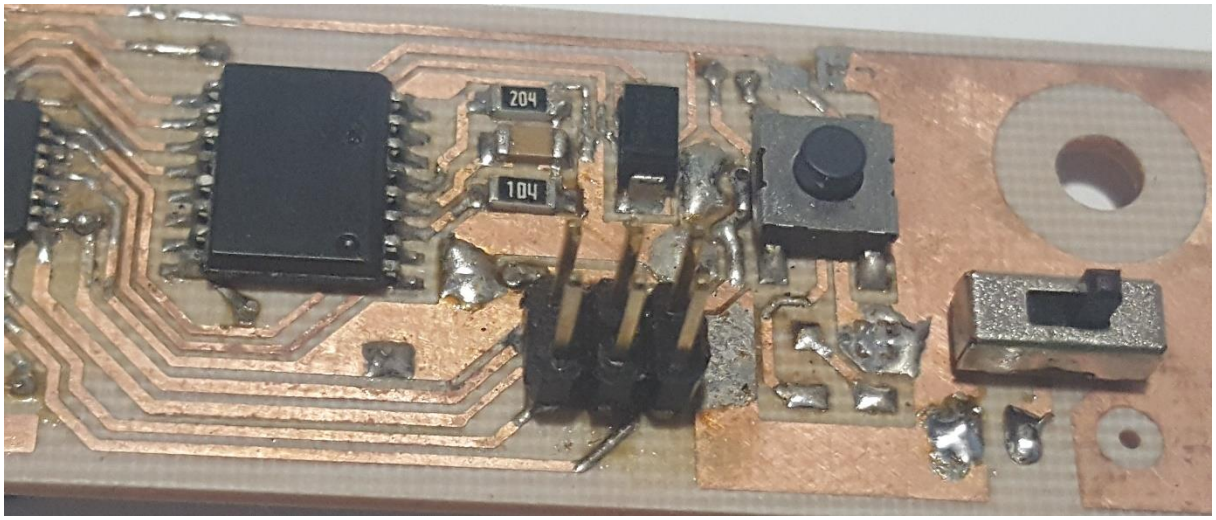


Slika 19: Shema ICSP konektorja in ATMega328p.

Ker bo ATMega328p v SMD izvedbi, ga s programatorjem za THT čipe ne bova mogla sprogramirati. Zato sva se zgledovala po prototipnih ploščicah Arduino in na vezje dodala ICSP konektor. Ima šest

kontaktov, ki so vse, kar potrebujemo za programiranje mikrokrmilnika – MISO, +5V, SCK, MOSI, RESET ter GND kontakti. To nama seveda omogoča tudi kakršno koli popravljanje kode po potrebi brez fizičnih posegov v vezje.

Konektor je sestavljen iz dveh vrstic po tri moške letvice.

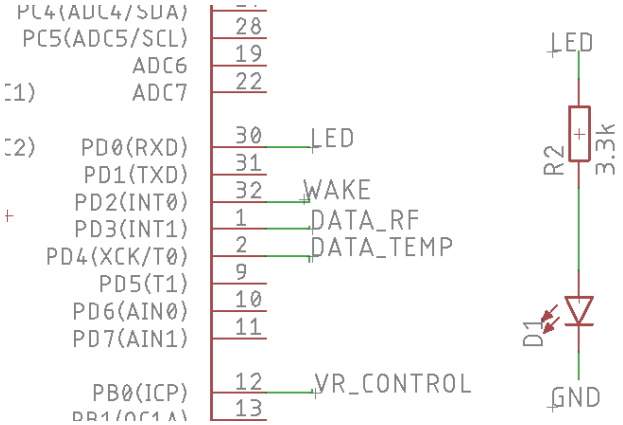


Slika 20: ICSP konektor.

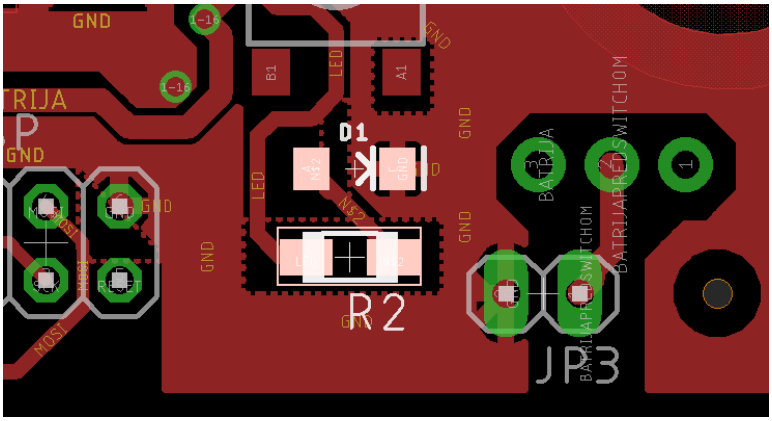
3.5 LED INDIKATOR

LED indikator je preprost, sestavljen je le iz rdeče LED diode in predupora, oba SMD izvedbe.

Indikator je uporabljen iz praktičnih razlogov. Z utripanjem diode je lažje vidno, kdaj mikrokrmilnik pošilja podatke preko oddajnika.

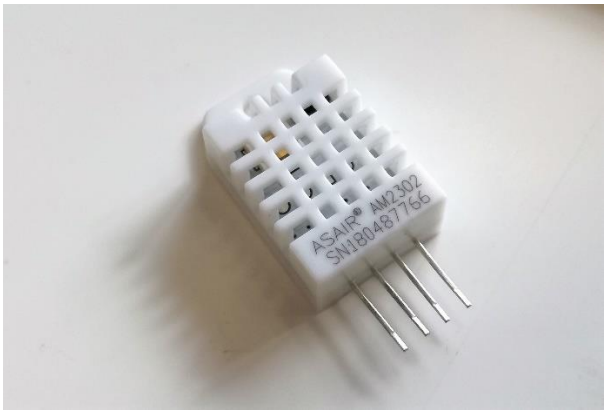


Slika 21: LED dioda - shema vezja.



Slika 22: LED indikator in predupor - izvedba PCB ploščice.

3.6 SENZOR DHT22



Slika 23: Senzor DHT22.

Z temperaturnim modulom sva želela meriti temperaturo in vlago. DHT11 in DHT22 sta ena redkih senzorjev, ki lahko hkrati meri oboje. Imava predhodne izkušnje z uporabo DHT11, a sva se zaradi pogreškov (temperatura odstopa za ±2°C, vlažnost pa ±5%) odločila za natančnejšo različico - DHT22. Ta ima odstopanje ±0,5°C ter ±2-5% odstopanje vlage. Temperaturno območje senzorja je od -40°C do 80°C, kar pomeni, da je primeren za skoraj katerokoli okolje.

Med branjem porabi 2.5 mA, meri pa lahko na vsake 2 sekundi.

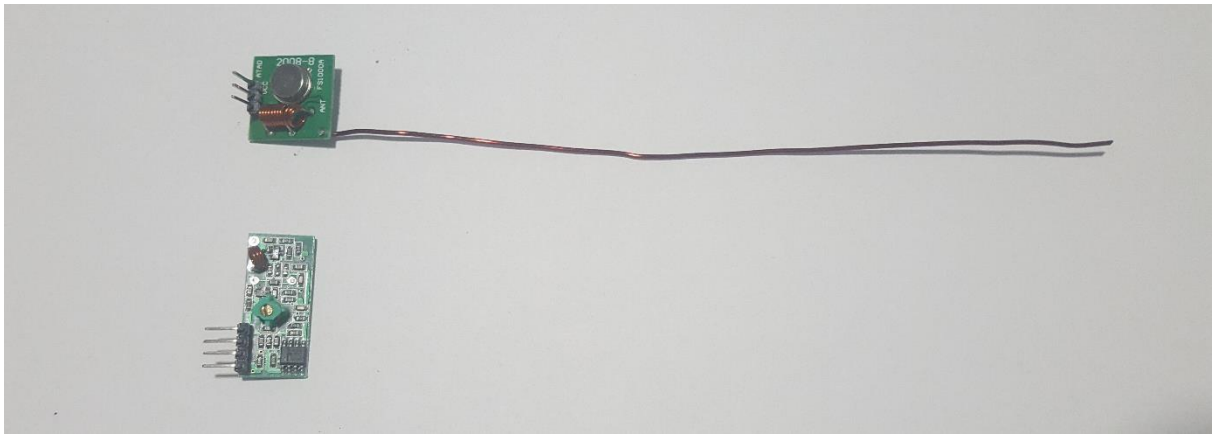
Senzor ima 4 kontakte (V_{CC}, DATA, NC, GND). V shemi vezja ga predstavlja letvica s štirimi kontakti. Tako lahko na vezje senzor priključimo na dva načina: lahko ga neposredno prispajkamo na ploščo ali priključimo s pomočjo letvic.

3.7 RADIO FREKVENČNI ODDAJNIK IN SPREJEMNIK

433 MHz, 868 MHz ter 2,4 GHz so frekvence, ki spadajo v ISM (Industrial, Scientific, Medical) radio frekvenčni pas, ki je po večini sveta na voljo brez licence. Moduli s temi frekvencami so zato pogosti. Izbrala sva generične 433 MHz oddajnike in sprejemnike. Na temperaturnem modulu je le oddajnik, na centralni postaji pa sprejemnik. Za pošiljanje podatkov uporabljava knjižnico »Virtual wire«, ki nama močno olajša delo.

Za pošiljanje podatkov na razdaljo več kot 1 meter sva morala pritrditi še dodatno anteno, ki je razdaljo povečala na vsaj 20 metrov, a se kljub temu občasno pojavijo problemi s pošiljanjem podatkov, predvsem skozi stene.

Sprejemnik ima 4 nogice, od katerih ena ni priključena, oddajnik pa 3 nogice.

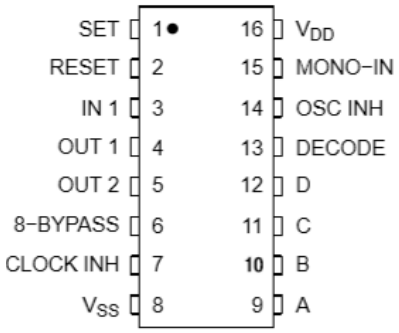


Slika 24: Oddajnik in sprejemnik

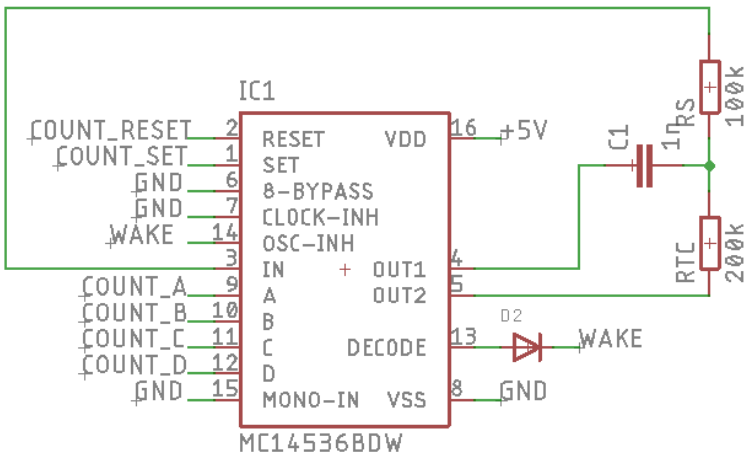
3.8 MC14536B DEKADNI ŠTEVEC

MC14536B timer je programirljiv, 24-stopenjski binarni števec. Stopnjo lahko izberemo binarno s pomočjo A, B, C in D nogic čipa. Vgrajen ima monostabilni multivibrator s pulzirajočim izhodom, katerega frekvenco lahko izberemo s kombinacijo vrednosti kondenzatorja in uporov. Z nastavljivo frekvenco in stopnjami lahko dosežemo širok nabor časovnih intervalov med posameznimi pulzi.

Kot je bilo omenjeno že pri opisu delovanja, ATMega328p potrebuje zunanji pulz oz. prekinitev, da ga zbudi iz načina spanja. Ta pulz priskrbi MC14536B. Želela sva 10 do 15 minutni časovni interval bujenja.



Slika 25: MC14536B dekadni števec.



Slika 26: MC14536B dekadni števec - shema.

Deluje med 3 V in 18 V vhodne napetosti. Ima 24 »flip-flop« stopenj, kar pomeni, da lahko šteje od 2^0 do 2^{24} . Frekvenco štetja nastavljamo s kombinacijo uporov R_S, R_{TC} in kondenzatorja C_1.

Na nogici DECODE OUT (nogica št. 13) se stanje postavi na logično visoko stanje, ko je dosežen željeni časovni interval. Ta nogica se smatra kot izhod števca. Dodana je tudi polprevodniška dioda. Ker je to izhod števca, ki je neposredno povezan na mikrokrmilnik skupaj z prej opisano tipko za bujenje, ta dioda preprečuje, da bi tok stekel od +5V preko tipke v nogico DECODE OUT tega števca. Dokumentacija namreč ne prikaže jasno, če je ta izhod med logičnim stanjem 0 visoko impedančen ali nizko impedančen, zato sva za vsak slučaj dodala polprevodniško diodo. Imenuje se ES1G, je SMA izvedbe z nazivnim tokom 1 A in maksimalnim koničnim tokom 30 A. Na njej napetost pade za 0.9 V, kar je še vedno dovolj, da mikrokrmilnik prepozna logično stanje 1.

Z logičnim stanjem 1 na nogici CLOCK INHIBIT se ustavi notranji oscilator števca. Povezan je na DECODE OUT, kar povzroči, da števec sam sebe ustavi, ko doseže željen časovni interval.

Na nogicah A, B, C, D in 8-BYPASS lahko binarno nastavljamo stopnjo multivibratorja. Če je recimo izbrana stopnja 22, bo multivibrator preklopil 2^{22}-krat, to je 4194304-krat.

TRUTH TABLES

Input					Stage Selected for Decode Out
8−Bypass	D	C	B	A	
0	0	0	0	0	9
0	0	0	0	1	10
0	0	0	1	0	11
0	0	0	1	1	12
0	0	1	0	0	13
0	0	1	0	1	14
0	0	1	1	0	15
0	0	1	1	1	16
0	1	0	0	0	17
0	1	0	0	1	18
0	1	0	1	0	19
0	1	0	1	1	20
0	1	1	0	0	21
0	1	1	0	1	22
0	1	1	1	0	23
0	1	1	1	1	24

Input					Stage Selected for Decode Out
8−Bypass	D	C	B	A	
1	0	0	0	0	1
1	0	0	0	1	2
1	0	0	1	0	3
1	0	0	1	1	4
1	0	1	0	0	5
1	0	1	0	1	6
1	0	1	1	0	7
1	0	1	1	1	8
1	1	0	0	0	9
1	1	0	0	1	10
1	1	0	1	0	11
1	1	0	1	1	12
1	1	1	0	0	13
1	1	1	0	1	14
1	1	1	1	0	15
1	1	1	1	1	16

Slika 27: Tabela izbire stopnje s pomočjo A, B, C, D in 8--BYPASS kontaktov. Vir: https://www.onsemi.com/pub/Collateral/MC14536B-D.PDF

Z logičnim stanjem 1 na nogici SET onemogočimo delovanje števca, dokler ne nastavimo A, B, C, D vhodov. Po nastavitvi damo SET na logično nizko stanje, da omogočimo delovanje.

S pulzom 5 V na nogico RESET začne števec s štetjem. Ko prešteje do nastavljene vrednosti, se ustavi, dokler mu na RESET nogico spet ne damo pulza visokega logičnega stanja.

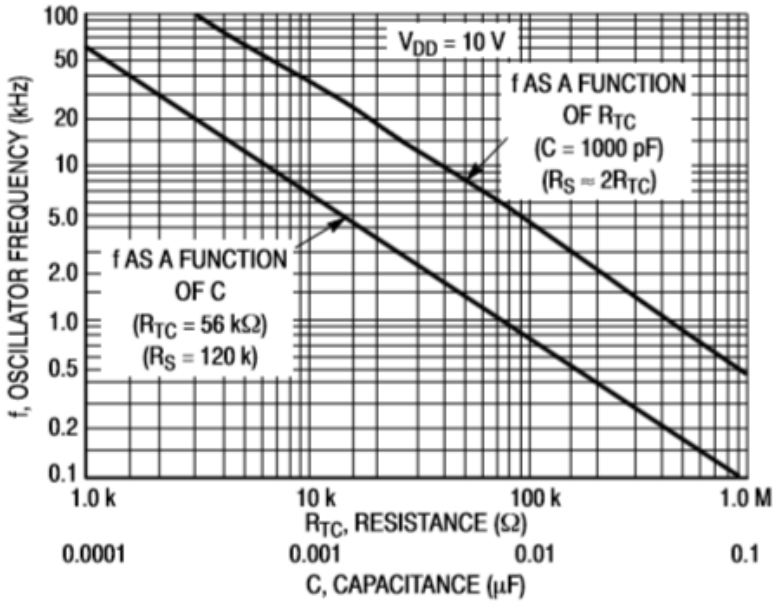


Figure 3. RC Oscillator Frequency as a Function of R_{TC} and C

Slika 28: Frekvenca RC oscilatorja števca v odvisnosti od vrednosti R_{TC} in C_1. Vir: https://www.onsemi.com/pub/Collateral/MC14536B-D.PDF

Na prejšnji sliki je graf, ki prikazuje odvisnost frekvence oscilatorja od vrednosti uporov in kondenzatorja. S pomočjo tega grafa lahko izberemo pravilne vrednosti uporov, da dobimo željeno frekvenco štetja. Ob kondenzatorju velikosti 1000 pF oz. 1 nF (torej zgornja črta) lahko iz grafa

odčitamo, da bomo pri upornosti R_{TC} približno 100kΩ dobili frekvenco okoli 5 kHz. Če s to frekvenco uporabimo pri 22. stopnji, bomo dobili časovni interval okoli 14 minut.

$$t = \frac{1}{f} \times 2^n = \frac{1}{5 \times 10^3 Hz} \times 2^{22} \approx 839\ s \approx 14\ min$$

V teoriji to pomeni, da se bo, ko bomo s mikrokontrolerjem sprožili začetek štetja, po približno 14 minutah na nogici 13 DECODE OUT vrednost postavila iz logične nule oz. 0V na logično eno oz. 5 V.

Želela sva časovni interval med 10 in 15 minutami, saj ne vidiva potrebe po zelo natančnih zakasnitvah.

S tem, ko se DECODE OUT postavi na 5 V, se na mikrokontrolerju sproži prekinitev, ki ga zbudi iz načina spanja. Ta potem izmeri vrednost tipala, jo pošlje centralnemu modulu preko RF modula, resetira števec, ga aktivira, da začne s štetjem, na koncu pa se postavi nazaj v spanje. Ta postopek traja približno eno sekundo, vmes pa regulator zagotavlja okoli 18.5 mA toka. Potem števec teoretično prešteje do 2^{22} pri frekvenci 5 kHz, kar traja približno 14 minut. Takrat iz regulatorja teče le okoli 60 µA. **Tako se poraba energije v vezju močno zmanjša, kar omogoča daljše delovanje vezja brez menjave baterije.**

Izračunala sva frekvenco 5 kHz, a ko sva ta del vezja testirala na prototipni ploščici, ta pri stopnji 22 ni dala ustrezne zakasnitve. Zato sva eksperimentirala z izbiro stopnje in na koncu izbrala 18. stopnjo. S pomočjo osciloskopa sva ugotovila, da je dejanska frekvenca oscilatorja le 1.562 kHz. To bi lahko bila posledica pogreškov kondenzatorja in uporov za nastavitev frekvence, parazitna kapacitivnost povezav na PCB ploščici ali pa nenatančen opis delovanja v dokumentaciji in tabeli.

4 IZDELAVA VEZJA

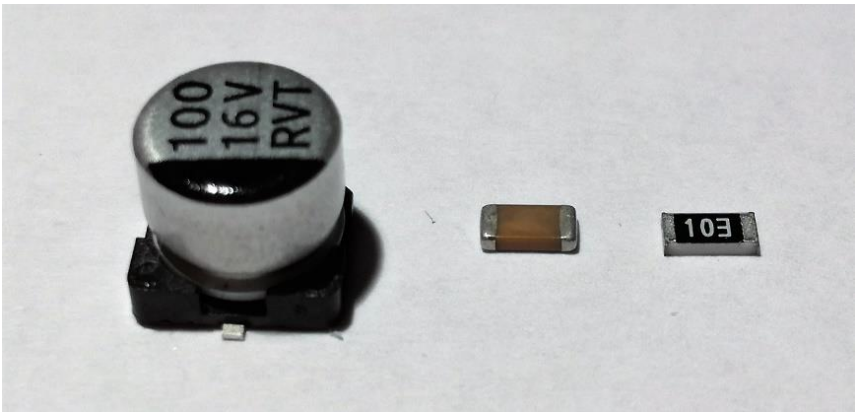
S pomočjo programa Autodesk Eagle sva narisala vezje ter v šoli izdelala PCB ploščico.

4.1 SPLOŠNO

Želela sva uporabiti čim več SMD komponent. Pri tem je bilo potrebno paziti na pravilno izbiro izvedbe (ang. package). Dodala sva še prostor in luknjo za vijak, kar omogoča pritrditev na npr. ohišje. Uporabila sva ploščico z dvema plastema bakra, kar je olajšalo izdelavo PCB ploščice in jo hkrati zmanjšalo.

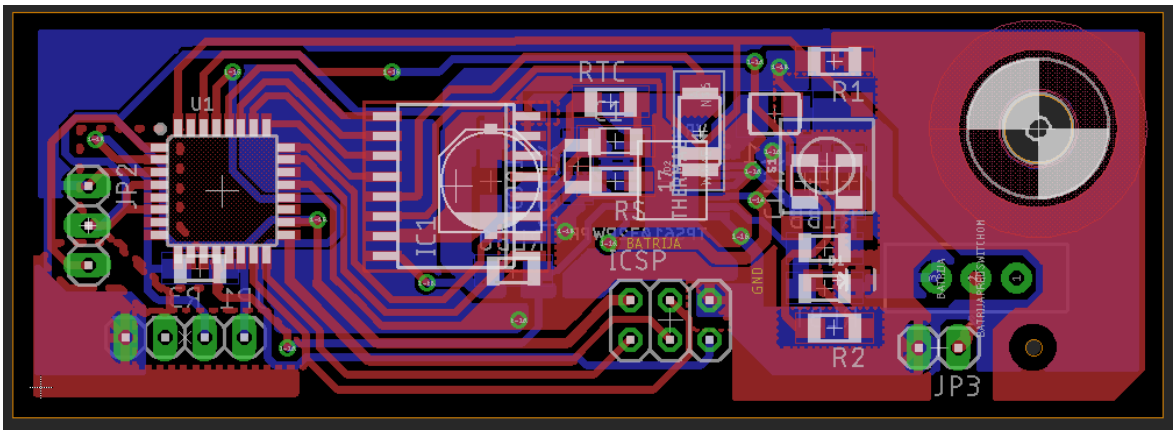
GND povezave so bile, če je bilo le mogoče, povezane s pomočjo za to predvidenega poligona (ang. ground plane). To, poleg ostalih ukrepov, navedenih v dokumentaciji napetostnega regulatorja, zmanjša elektromagnetne vplive tega na preostalo vezje.

Vsi upori in kondenzatorji so tehnologije SMD, in sicer izvedbe 3216, kar pomeni, da je element dolžine 3.2 mm in širine 1.6 mm. Izjema je kondenzator C_{OUT} pri regulatorju, ki je elektrolitski SMD kondenzator izvedbe »V-chip« tipa D, kar pomeni, da je širina in dolžina enaka 6.6 mm.

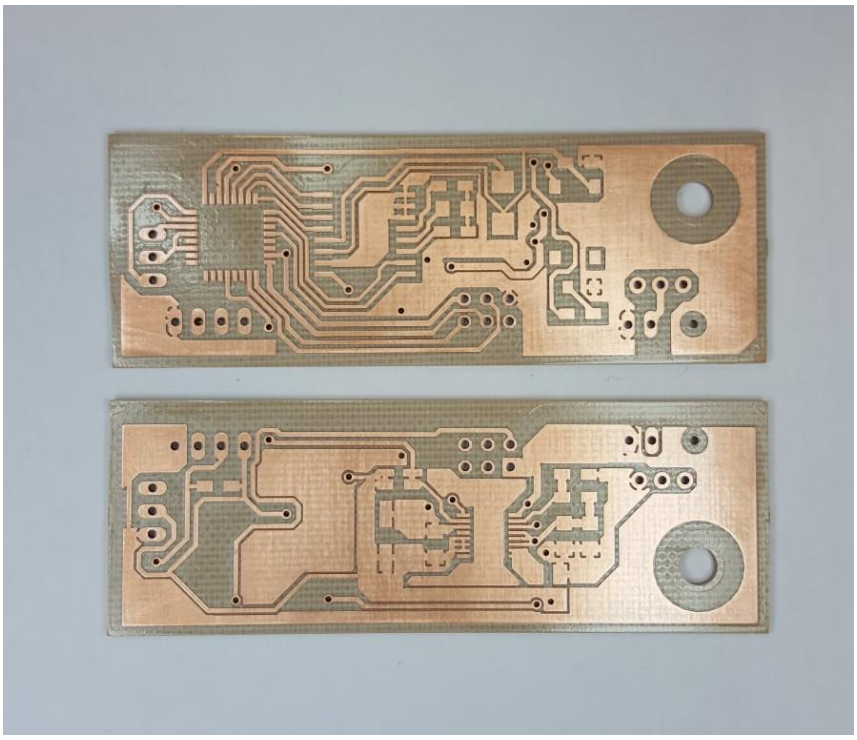


Slika 29: Od leve proti desni: "v-chip" elektrolitski kondenzator, 3216 kondenzator, 3216 upor.

Vezje je, kot že prej omenjeno, dvoslojno. Na spodnji sliki sta prikazana oba sloja hkrati. Med sabo sta povezana na vsakem kontaktu THT elementov (to je stikalo in letvice) ter na povezavah, ki se imenujejo »via« in so predvidene za povezovanje obeh slojev.



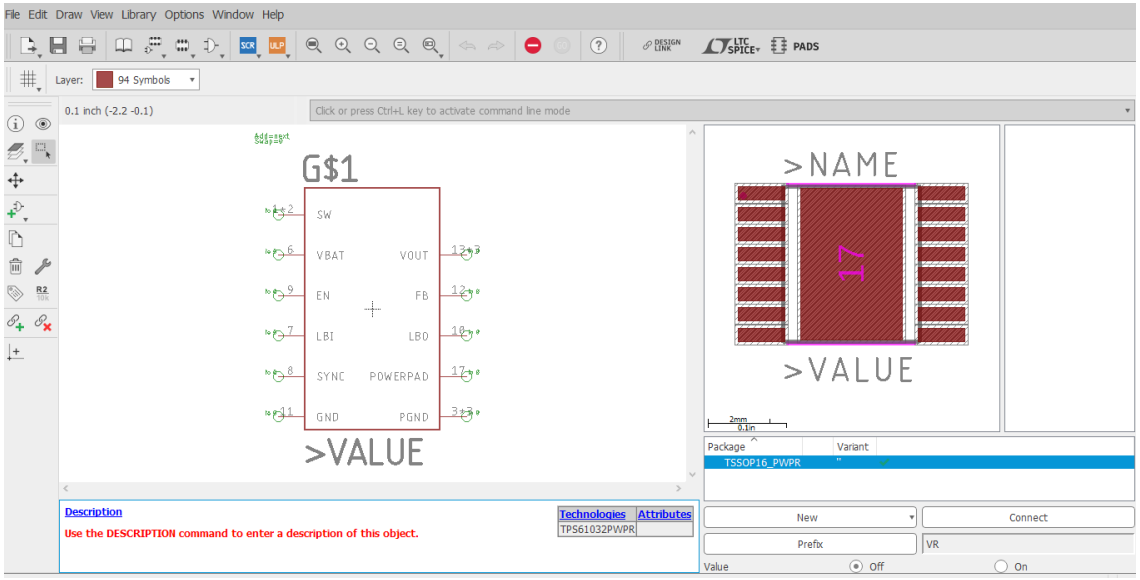
Slika 30: Izvedba PCB ploščice - oba sloja.



Slika 31: Izdelana PCB ploščica. Zgoraj je prikazan zgornji, spodaj pa spodnji sloj vezja.

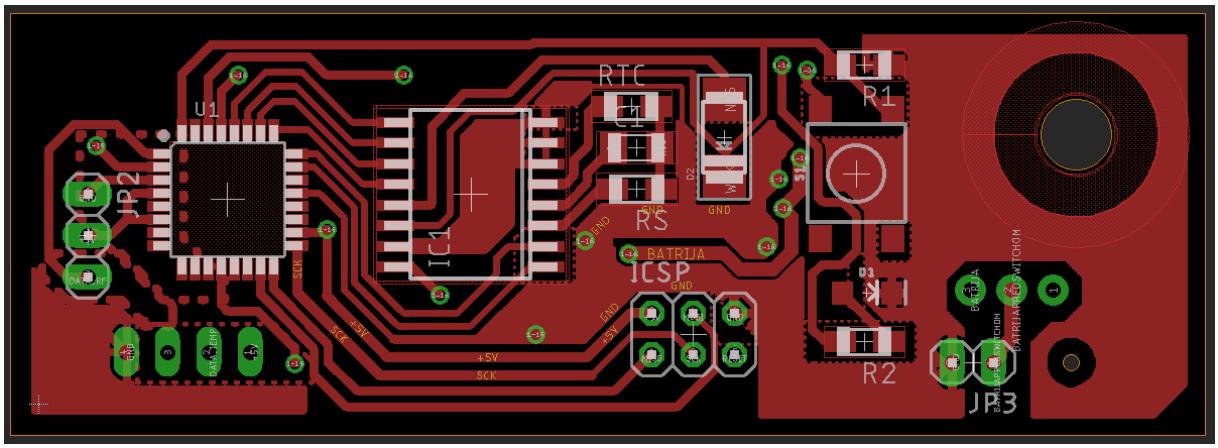
4.2 IZDELAVA NOVIH KOMPONENT

Ker modela regulatorja napetosti ni v prednaloženih knjižnicah programa Eagle, sva ga morala ustvariti sama. Ob tem sva si pomagala z dokumentacijo čipa.



Slika 32: Izdelava nove komponente - regulatorja napetosti - v programu Eagle.

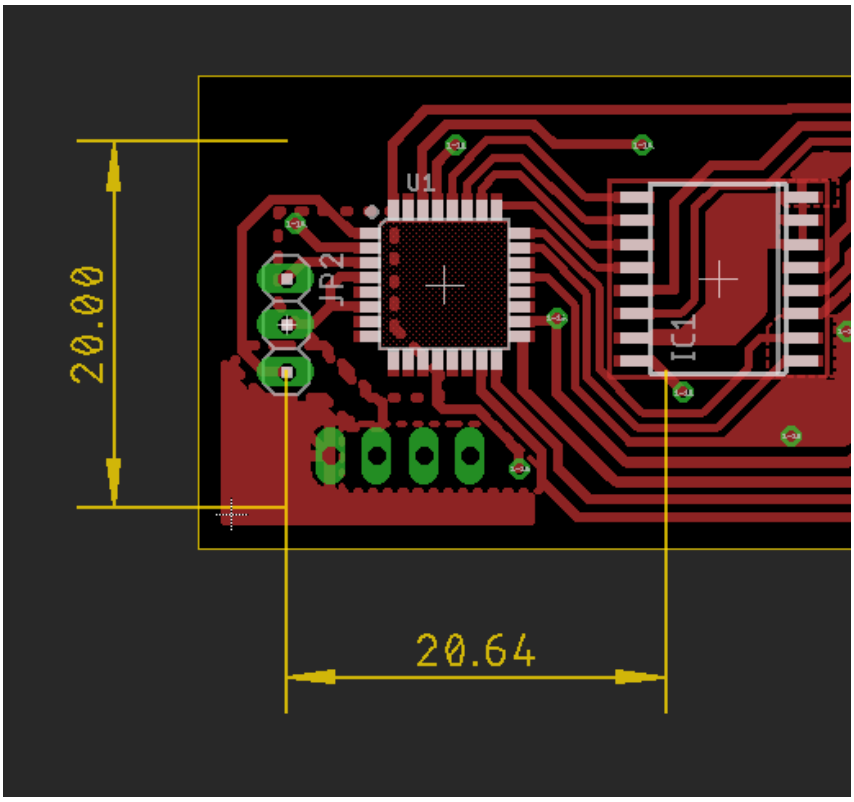
4.3 OPIS ZGORNJEGA SLOJA



Slika 33: Izvedba PCB ploščice - zgornji sloj.

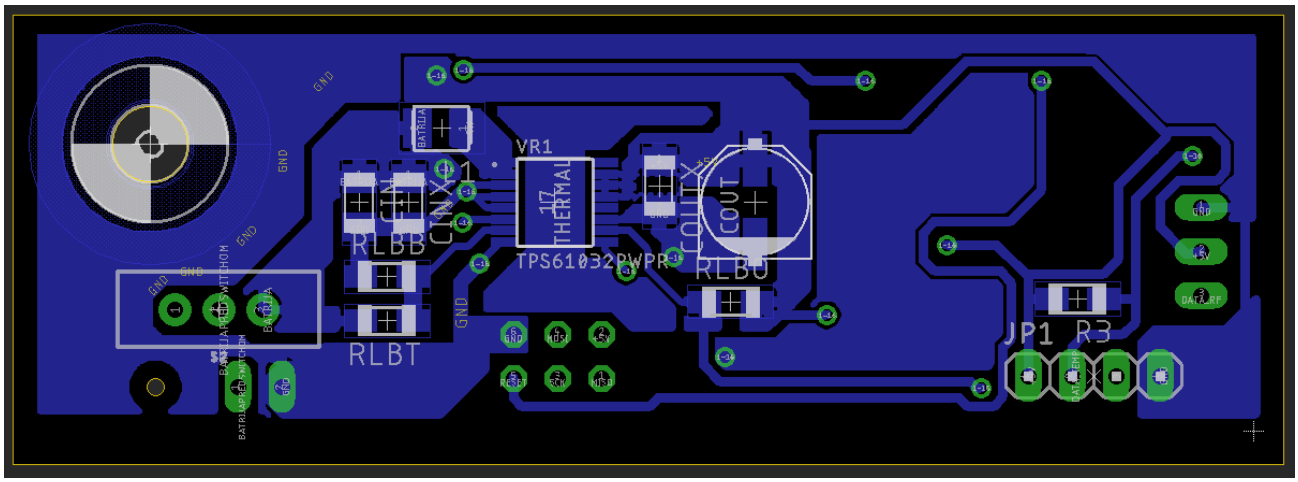
Na zgornjem sloju se od leve proti desni nahajajo ženske letvice za priklop RF oddajnika (oznaka JP2), mikrokontroler ATMega328p (oznaka U1), dekadni števec MC14536B (oznaka IC1), upora R_S, R_{TC} in kondenzator C_1, ki nastavijo frekvenco števca, polprevodniška dioda za zaščito števca, tipka za zbujanje mikrokrmilnika, LED dioda, njen predupor R_2 ter kontakti za priklop baterije (oznaka JP3).

RF modul je širok in dolg 20 mm. Vezje je bilo načrtovano tako, da višina komponent pod modulom ne bi onemogočala dejanske priključitve modula. ICSP konektor je postavljen tako, da RF modul ni nad njim.



Slika 34: Postavitev RF oddajnika na PCB ploščici.

4.4 OPIS SPODNJEGA SLOJA

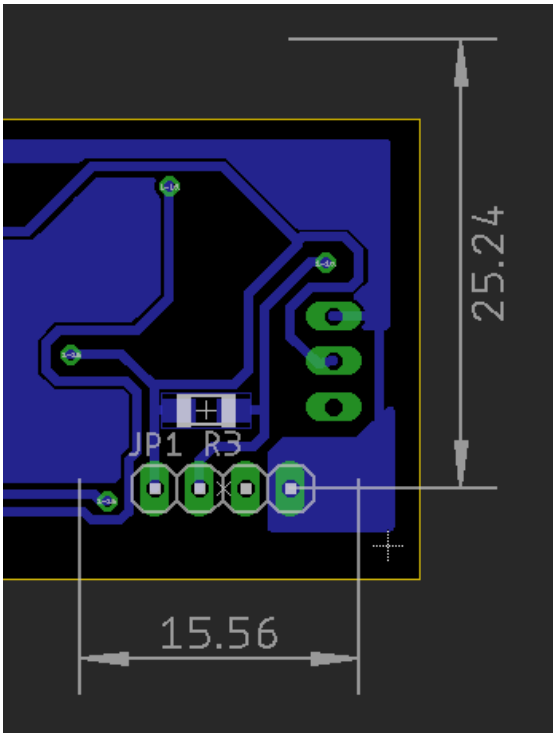


Slika 35: Izvedba PCB ploščice - spodnji sloj.

Na zgornji sliki je spodnji sloj PCB ploščice. Slika je vertikalno zrcaljena, da so oznake komponent berljive.

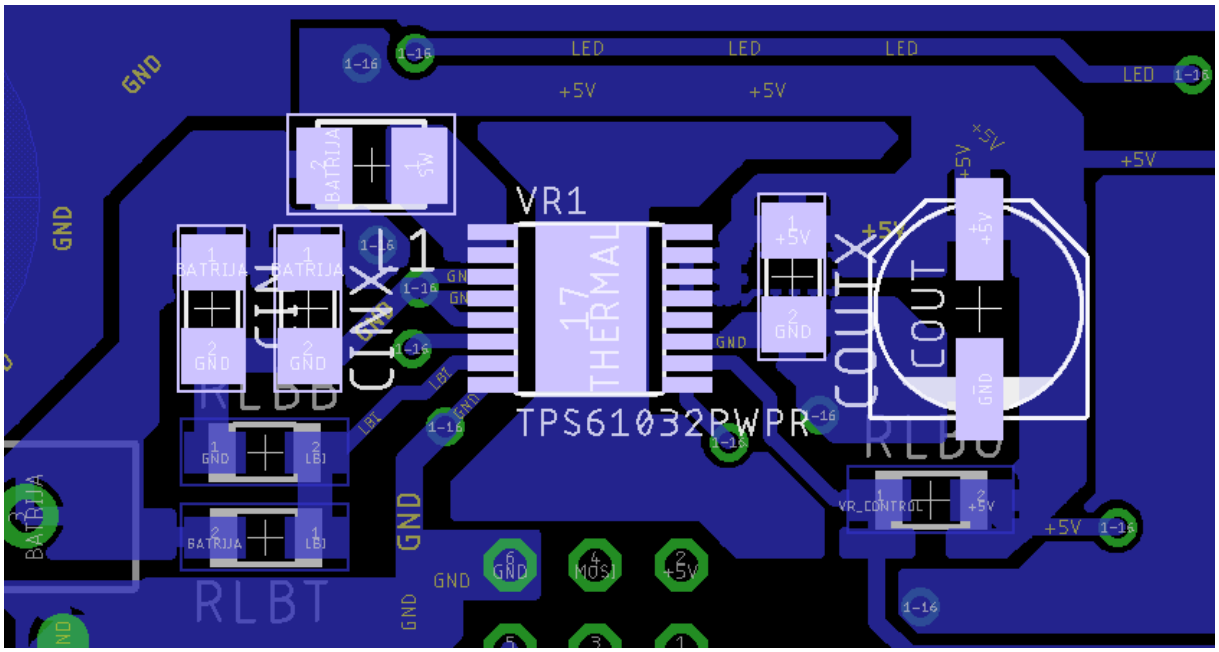
Na tej strani PCB ploščice se od leve proti desni nahajajo stikalo za vklop in izklop napajanja, regulacijsko vezje ter ženske letvice za priključitev DHT22 senzorja. Pod regulacijsko vezje po vrstnem redu od leve proti desni spadata kondenzatorja C_{IN} in C_{INX}, upora R_{LBT} in R_{LBB}, regulator TPS61032PWP, upor R_{LBO} ter kondenzatorja C_{OUTX} in C_{OUT}.

Tudi tukaj je bilo potrebno paziti, da prispajkane komponente niso motile priključitve DHT22 senzorja.



Slika 36: postavitev DHT22 senzorja na spodnjem sloju PCB ploščice.

V dokumentaciji regulatorja so opisane tudi smernice za razporejanje in povezovanje komponent stikalnega regulatorja, po katerih sva se ravnala. Ta razporeditev je zelo pomembna, saj so prisotne visoke frekvence preklapljanja (600 kHz). Nepravilne povezave in razporeditev lahko povzročijo veliko parazitno kapacitivnost in induktivnost, poveča pa se tudi elektromagnetni vpliv na ostalo vezje (ang. EMI).



Slika 37: Vezje regulatorja in označeni pomembnejši elementi.

Vezje sva začela razporejati s postavitvijo regulatorja. Naslednji pomembnejši del je kondenzator na izhodu, na vezju označen kot C_{OUTX}. Postavljen mora biti čim bližje regulatorju, povezave pa čim širše in krajše, za čim manjšo parazitno induktivnost.

Naslednja zelo pomembna komponenta je induktor, na vezju označen kot L_1. Z regulatorjem mora biti povezana s čim krajšo povezavo, ker se lahko drugače pojavi EMI (ang. electromagnetic interference).

Zadnja pomembna komponenta je vhodni kondenzator, na sliki označen kot C_{IN}, ki je prav tako postavljen čim bližje regulatorju, skupaj z kondenzatorjem C_{INX}.

Na koncu sva na vezje postavila ostale upore, dodala poligon za hlajenje čipa (označen kot THERMAL), vse GND pa poskušala povezati s čim večjimi poligoni, za zmanjšanje elektromagnetnega vpliva regulatorja na ostale komponente.

C_{OUT} (elektrolitski kondenzator) je tako imenovan »bulk« kondenzator, ki je lahko povezan z relativno dolgimi povezavami.

5 PODROBNEJŠI OPIS PROGRAMOV

5.1 TEMPERATURNI MODUL

Za delovanje programa na temperaturnem modulu so potrebne tri knjižnice:

- *AVR sleep*, ki omogoča spanje mikrokrmilnika za varčevanje z energijo;
- *Virtual Wire*, ki omogoča pošiljanje podatkov preko RF modulov;
- *DHT* za uporabo senzorja DHT22.

```
void setup(void)
{
  EICRA = (1 << ISC01) | (1 << ISC00); //Nastavi INT 0, da se vključi pri pozitivnem prehodu
  EIMSK = (1 << INT0); // Omogoči INT 0
  ADCSRA = 0; // Izključi analogno-digitalni pretvornik
  PRR = _BV(PRADC) | _BV(PRUSART0) | _BV(PRSPI) | _BV(PRTIM2); //Izključi Power Reduction ADC, USART0, Serial Peripheral Interface, Timer/Counter2
  for (byte i = 0; i < 20; i++) {
    pinMode(i, INPUT); //Vse pine nastavi kot vhode
    digitalWrite(i, HIGH); //na vse pine nastavi pullup
  }
  //Nastavi vse potrebne pine
  pinMode(2, INPUT); //INT 0 pin
  pinMode(SEND, OUTPUT); //LED indikator za pošiljanje
  pinMode(ERR, OUTPUT); //LED indikator za napako
  pinMode(RESET, OUTPUT); //Pin za ponastavitev timerja
  pinMode(SET, OUTPUT); //Pin za vklop timerja
  digitalWrite(SET, HIGH); //Nastavi na 5V dokler niso A,B,C,D vhodi nastavljeni
  //Pini za nastavitev dekad timerja
  pinMode(A, OUTPUT);
  pinMode(B, OUTPUT);
  pinMode(C, OUTPUT);
  pinMode(D, OUTPUT);
  //Nastavitev za 11 minut
  digitalWrite(A, HIGH);
  digitalWrite(B, LOW);
  digitalWrite(C, LOW);
  digitalWrite(D, HIGH);
  digitalWrite(SET, LOW);//Vključi timer
  dht.begin(); //Inicializira DHT22
  vw_set_tx_pin(TRANSMIT_PIN);
  vw_setup(2000); //Inicializira oddajnik
}
```

Slika 38: Začetna nastavitev.

Nato, kot je razvidno v zgornji sliki, v funkciji *setup* nastavimo vse potrebne registre ter konfiguriramo vhode in izhode. Najprej v register EICRA (External Interrupt Control Register A) postavimo bita ISC01 in ISC00 na vrednost 1. S tem nastavimo prekinitveni program (v nadaljevanju prekinitev) št. 0, da se sproži na pozitivnem prehodu stanja signala. To se zgodi, ko števec pride do željene vrednosti in svoj izhod postavi na logično 1.

Nato v register EIMSK (External Interrupt Mask Register) bit INT0 postavimo na 1 in s tem omogočimo oz. vklopimo prekinitev 0. Omogočiti je treba tudi globalni prekinitveni sistem.

Potem v register PRR (Power Reduction Register) postavimo enice v bite za izklop delov mikrokrmilnika, ki jih ne potrebujeva. To so analogno-digitalni pretvornik, serijska komunikacija in števec 2.

Vse nogice nastavimo na vhode, z notranjim »pullup« uporom. Nato definiramo nogico št. 2, ki predstavlja prekinitev 0 (oznaka INT 0), kot vhod. Ostale nogice, kot recimo izhod za LED diodo, nastavimo kot izhod. Tu nastavimo vrednost A, B, C, ter D nogic tako, da dobimo željeno stopnjo za števec MC14536B. Na koncu še inicializiramo senzor DHT22 ter RF oddajnik.

```
                              //ID-tip      ID-modul
static char header[9] = {'!', '0', '1', ';', '0', '0', ';'}; //ID tipa in modula ki je na začetku vsakega sporočila
char masterBuffer[100] = {'!', '0', '1', ';', '0', '0', ';'}; //Array z sporočilom
```

Slika 39: Header – polje znakov.

Ker se mora vsako sporočilo brezžičnega modula začeti z ID številko tipa ter ID številko modula, je vse to shranjeno v polje znakov z imenom »header«. To se vedno kopira na začetek polja, imenovanega »masterBuffer«, kamor se shranjuje sporočilo, preden je poslano.

```
void sendMessage()
{
  digitalWrite(SEND, HIGH); //LED indikator da se pošiljajo podatki
  vw_send((uint8_t *)masterBuffer, endFinder());
  vw_wait_tx(); // Počakaj da se pošlje celo sporočilo
  digitalWrite(SEND, LOW);
  for (int i = 0; masterBuffer[i] != 0; i++) //Kopiranje headerja
    masterBuffer[i] = header[i];
}
```

Slika 40: Funkcija sendMessage.

```
void appendF(float f) //Na konec trenutnega sporočila doda še float
{
  char buffer[10];
  dtostrf(f, 2, 1, buffer); //Pretvori float v char array
  int start = endFinder();
  int i = 0;
  while (buffer[i] != 0) //Prekopira buffer v sporočilo
  {
    masterBuffer[start + i] = buffer[i];
    i++;
  }
  masterBuffer[start + i] = ';'; //Na koncu doda še podpičje
  masterBuffer[start + i + 1] = 0;
}

void appendI(int n) // Na konec trenutnega sporočila doda še število
{
  char buffer[10];
  itoa(n, buffer, 10); //Pretvori int v char array
  int start = endFinder();
  int i = 0;
  while (buffer[i] != 0)
  {
    masterBuffer[start + i] = buffer[i]; //Prekopira buffer v sporočilo
    i++;
  }
  masterBuffer[start + i] = ';'; //Na koncu doda še podpičje
  masterBuffer[start + i + 1] = 0;
}
```

Slika 41: Funkciji appendF, ter appendI.

```
int endFinder() //Ugotovi na katerem mestu se konča trenutno sporočilo
{
  int i = 0;
  while (masterBuffer[i] != 0)
    i++;
  return i;
}
```

Slika 42: Funkcija endFinder.

Da bi bil sistem kod čim bolj univerzalen, sva ustvarila 4 funkcije, ki pomagajo tvoriti ter poslati sporočilo. Funkcija *sendMessage* pošlje sporočilo shranjeno v polju »masterBuffer«, pri tem pa prižge LED indikator. Funkciji *appendF* ter *appendI* na konec trenutnega sporočila dodata še spremenljivko tipa float oz. integer, kamor se shranijo izmerjeni podatki. To je v primeru temperaturnega modula uporabljeno za podatka o temperaturi in vlažnosti zraka.

To storita tako, da najprej spremenljivko tipa float s pomočjo funkcije *dtostrf* oz. spremenljivko tipa int s pomočjo funkcije *itoa* (obe funkciji sta del programskega jezika C++) spremenita v polje znakov, nato pa te znake ki so shranjeni v polju »buffer« prekopirata v »masterBuffer«. Ker pa nikoli ne vemo, kako dolgo je polje »masterBuffer«, sva ustvarila funkcijo *endFinfer*, ki išče znak 0 oz. »null character« na koncu polja, ter vrne zadnje mesto v polju.

```
ISR(INT0_vect) //Funkcija za Interrupt 0
{
  cli(); //Onemogoči globalni sistem interruptov
}
```

Slika 43: ISR funkcija.

ISR (Interrupt Service Routine) funkcija se izvede, ko se sproži prekinitev 0. Tu služi le za bujenje čipa, zato je edina stvar v funkciji izključitev globalnega prekinitvenega sistema, kar prepreči ponoven vklop prekinitve, dokler se ne opravi meritev temperature, vlage ter prenos podatkov preko modula. Za tem se globalni sistem ponovno vključi.

```
void loop(void)
{
  float h = dht.readHumidity(); //Prebere vlago
  float t = dht.readTemperature(); //Prebere temperaturo
  if (isnan(h) || isnan(t)) { //Če branje ni uspelo, se vrni na začetek
    digitalWrite(ERR, HIGH); //LED indikator za napako
    delay(500);
    digitalWrite(ERR, LOW);
    delay(500);
    return;
  }
  appendF(t); //Na konec trenutnega sporočila doda temperaturo
  appendI(round(h)); //Na konec trenutnega sporočila doda vlago
  sendMessage(); //Pošlje sporočilo
  digitalWrite(RESET, HIGH); //Resetira timer
  delay(500);
  digitalWrite(RESET, LOW);
  delay(500); //Počaka pol sekunde, da se interrupt ne sproži pomotoma ob ponoastavitvi
  sleep_enable(); //Omogoči spanje
  set_sleep_mode(SLEEP_MODE_PWR_DOWN); //Nastavi način spanja
  adcsra = ADCSRA; //Shrani ADC Control in Status Register A
  ADCSRA = 0; //Izklopi ADC
  EIMSK |= _BV(INT0); //Omogoči INT0
  mcucr1 = MCUCR | _BV(BODS) | _BV(BODSE); //Izključi brown-out detector
  mcucr2 = mcucr1 & ~_BV(BODSE);
  MCUCR = mcucr1;
  MCUCR = mcucr2;
  sei(); //Omogoči globalni sistem interruptov
  sleep_cpu(); //Gre v spanje
  sleep_disable(); //Tu se zbudi z spanja ob interruptu
  ADCSRA = adcsra; //Obnovi ADCSRA
}
```

Slika 44: Delovanje programa.

V funkciji *loop* (torej v glavni funkciji) funkciji *dht.*readHumidity in *dht.readTemperature* prebereta trenutne vrednosti temperature in vlage, za tem pa program preveri, če sta bili meritvi uspešni in ju v tem primeru s funkcijama *appendI* ter *appendF* dodata k sporočilu.

Poleg temperature je tudi vlaga z strani senzorja podana kot float, a zaradi relativno velikega pogreška program zanemari decimalno mesto ter ga zaokroži na celo število. Nato se sporočilo pošlje, hkrati pa se prižge LED indikator.

Naslednji korak je ponastavitev števca, da ponovno začne s štetjem. To doseževa s 500 ms dolgim pulzom na RESET nogico števca MC14536B.

Nato omogočimo spanje čipa ter nastavimo način spanja na »Power down«, ki porabi najmanj energije. Zatem shranimo ADC Control in Status Register A in ju izklopimo. Izključimo tudi funkcijo *brown-out detection* za še manjšo porabo energije. Z registrom EIMSK in funkcijo *sei* nato vklopimo INT 0 ter globalni prekinitveni sistem. Na koncu damo mikrokrmilnik v spanje.

Ko se zbudi zaradi prekinitve na INT0, se program začne pri funkciji *sleep_disable.* Nato takoj obnovi register ADCSRA ter se vrne na začetek.

5.2 CENTRALNA POSTAJA

5.2.1 Arduino Mega2560

Arduino Mega2560 je razvojna ploščica, osnovana na mikrokrmilniku ATMega2560 podjetja Atmel oz. Microchip. Ima 256 kB FLASH spomina, 8 kB SRAM ter 4 kB EEPROM. Ta je preko USB kabla priključen na Raspberry Pi, ima zelo preprost program.

```
void setup()
{
  delay(1000);
  Serial.begin(115200);
  vw_set_rx_pin(REC);
  vw_setup(2000); //Inicializacija sprejemnika
  vw_rx_start();
  pinMode(LED, OUTPUT);
}
```

Slika 45: Začetna nastavitev Arduino Mega2560.

V funkciji *setup* vzpostavi serijsko povezavo z Raspberry Pijem in inicializira RF sprejemnik.

```
void loop()
{
  uint8_t buf[VW_MAX_MESSAGE_LEN];
  uint8_t buflen = VW_MAX_MESSAGE_LEN;
  String msg = "";
  if (vw_get_message(buf, &buflen)) //Ne blokira
  {
    int i;
    digitalWrite(LED, HIGH); //LED indikator

    for (i = 0; i < buflen; i++) //Prekopira sporočilo v String msg
    {
      msg += (char)buf[i];
    }
    Serial.println(msg); //Pošlje sporočilo Raspberry Piju
    digitalWrite(LED, LOW);
  }
}
```

Slika 46: Glavna funkcija Arduino Mega2560.

V funkciji *loop* se ustvari začasen pomnilnik z imenom »buf« ter string »msg«. Ko sporočilo prispe, se shrani v »buf«, ter nato prekopira v »msg«. Nato je »msg« posredovan Raspberry Piju preko serijskega vmesnika.

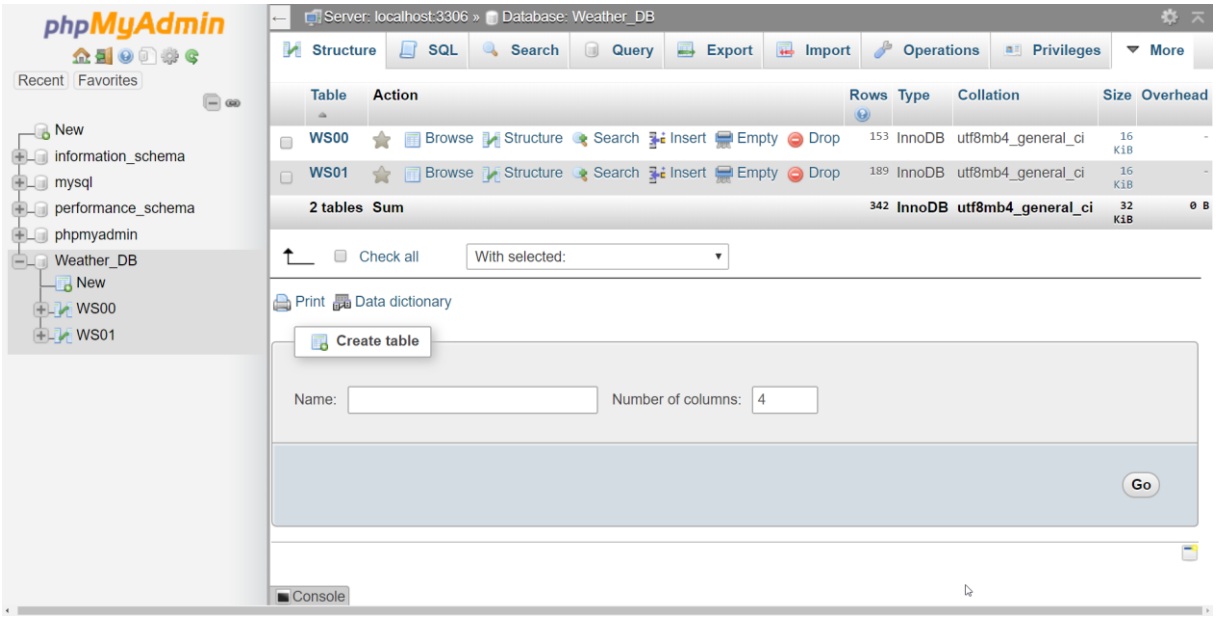
5.2.2 Raspberry Pi

Raspberry Pi, čeprav je velik le 85 x 56 mm, ima vse funkcije ki bij jih pričakovali na navadnem računalniku. Na njegovi micro SD kartici, ki služi kot edini spomin na njem, je naložena Raspbian distribucija operacijskega sistema Linux, prilagojena za Raspberry Pi. Ima vgrajen WiFi in Bluetoooth modul, poleg tega pa ima tudi Ethernet priključek, ki omogoča hitrosti prenosa do 300 mb/s. Ker je računalnik, omogoča izvajanje mnogo programov neodvisno en od drugega, in to v realnem času, zato je primeren za gostovanje spletnih strani, do katerih lahko dostopa več uporabnikov hkrati brez opaznih upočasnitev.

Program za Raspberry Pi je napisan v programskem jeziku Python, ki je največkrat uporabljen za programiranje na njem.

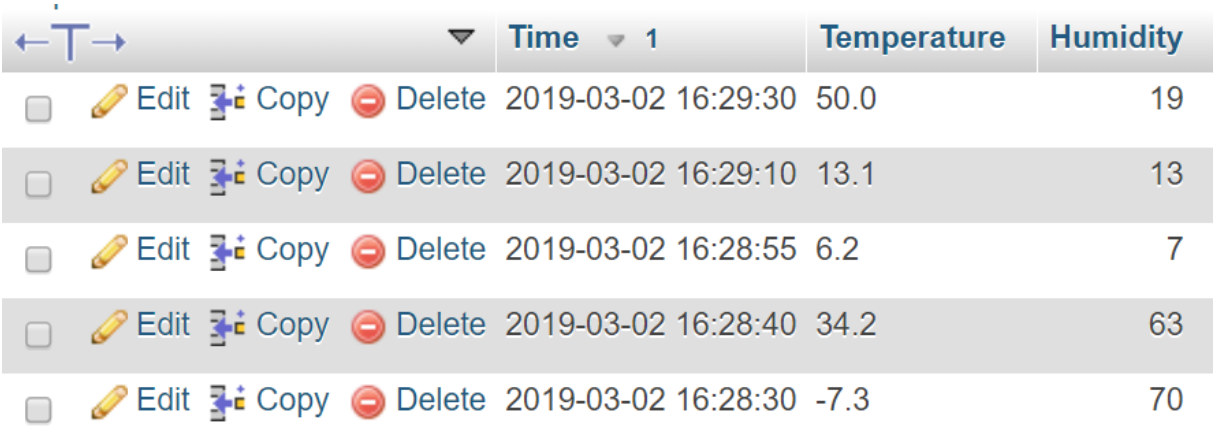
5.2.2.1 SQL podatkovna baza

Čeprav bi podatke lahko shranjevala kot besedilo v datoteke, sva želela to narediti bolj prilagodljivo. Zato sva se odločila za SQL bazo. SQL (ang. *Structured Query Language* oz. *strukturirani povpraševalni jezik za delo)* je poseben programski jezik, ki se uporablja za ustvarjanje ter nadziranje podatkovnih baz. Za upravljalni sistem baze, sva uporabila odprtokodni sistem *MySQL*. Za lažjo uporabo sva uporabila program *phpMyAdmin,* ki namesto ukaznega terminala ponuja grafično upravljanje baz.



Slika 47: SQL baza Weather_DB.

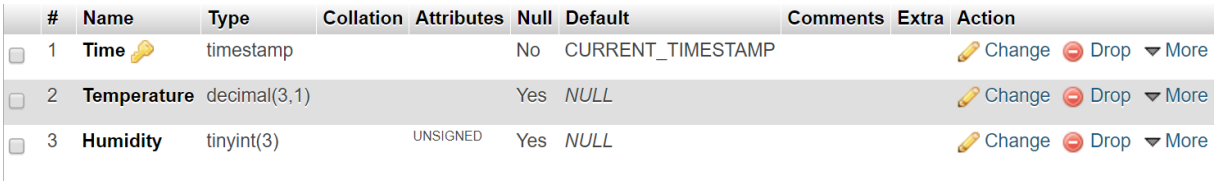
Ustvarila sva podatkovno bazo z imenom *Weather_DB* oz. vremenska podatkovna baza, ki predstavlja tip modula, v tem primeru temperaturni modul. Ta ima dve »Tabeli« z imenom WS00 ter WS01 kar je kratko za *Weather station* (temperaturni modul).



	Time 1	Temperature	Humidity
Edit Copy Delete	2019-03-02 16:29:30	50.0	19
Edit Copy Delete	2019-03-02 16:29:10	13.1	13
Edit Copy Delete	2019-03-02 16:28:55	6.2	7
Edit Copy Delete	2019-03-02 16:28:40	34.2	63
Edit Copy Delete	2019-03-02 16:28:30	-7.3	70

Slika 48: Stolpci v podatkovni bazi – Slika je simbolična.

Vsaka tabela se nato deli na »stolpce«, v tem primeru na tri. Prvi je čas, kamor se zabeleži čas ob vpisu v bazo, drugi je temperatura ter tretji vlaga, kamor se vpišeta vrednosti s temperaturnega modula.



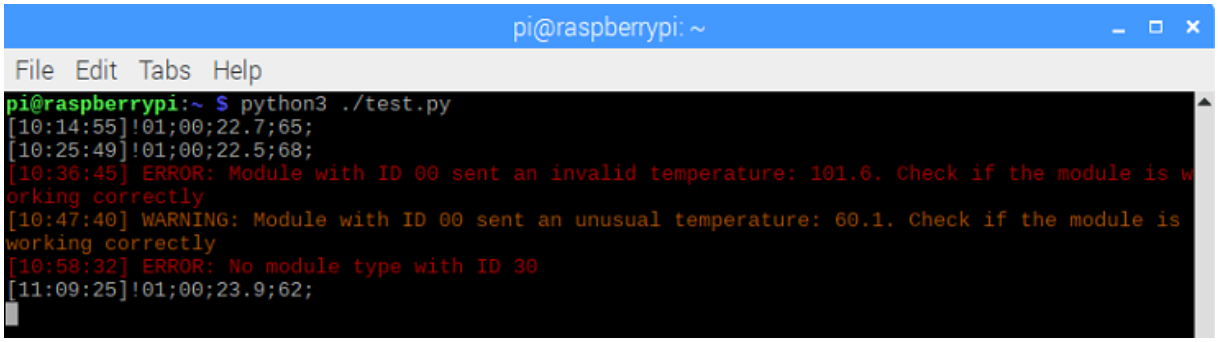
#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
1	Time	timestamp			No	CURRENT_TIMESTAMP			Change Drop More
2	Temperature	decimal(3,1)			Yes	*NULL*			Change Drop More
3	Humidity	tinyint(3)		UNSIGNED	Yes	*NULL*			Change Drop More

Slika 49: Podatkovni tipi stolpcev.

Vsak stolpec ima svoj podatkovni tip. Ker se bo v roku nekaj let nabralo veliko podatkov, sva zaradi varčevanja s prostorom določila le minimalno potrebne podatkovne tipe. Podatkovni tip stolpca čas je *timestamp,* ki shranjuje čas od leta do sekunde. V tem primeru je nastavljen tako, da ne more shraniti ničelne (null) vrednosti. Če modul posreduje neveljaven podatek temperature ali vlage, pa se lahko zapiše ničelna vrednost. Podatkovni tip stolpca temperatura je *decimal,* ki sicer lahko shranjuje vrednosti do 65 števk z do 30 decimalnimi mesti, a sva ga omejila na 3 števke z eno decimalko, saj temperatura ne bo nikoli manj kot -99 stopinj ali več kot 99 stopinj Celzija. Podatkovni tip stolpca vlaga je *unsigned tynyint, ki lahko shranjuje vrednosti od 0 do 255.*

5.2.2.2 Serijski vmesnik

Za sprejemanje podatkov preko serijskega vmesnika, sva v programskem jeziku Python 3 sestavila majhen program, ki sprejme vse podatke, ki pridejo preko serijskega vmesnika ter jih vnese v pravilno SQL bazo. Da se program ne sesuje, ko dobi neveljaven podatek, sva dodala več preverjanj ustreznosti poslanih kod, ter izpisovanje napak v konzolni terminal.



Slika 50: Terminal – Slika je simbolična.

Za sprejemanje podatkov preko serijskega vmesnika sva uporabila knjižnico *serial*, za vnašanje podatkov v SQL bazo knjižnico *MySQldb,* za lepši ter preglednejši izpis podatkov pa knjižnici *time* za izpisovanje časa ter *termcolor* za barvanje sporočil napak z rdečo.

```
def printF (msg,sev):
    if sev == 0: #Serverity 0-notification
        col = "white"
        name = ""
    elif sev == 1: #Warning
        col = "yellow"
        name = "WARNING: "
    elif sev == 2: #Error
        col = "red"
        name = "ERROR: "
    else:
        raise Exception("Invalid severity level: " + sev)

    print(colored("["+time.strftime("%H:%M:%S")+"]"+name+msg, col))
```

Slika 51: Funkcija printF.

Za lažje barvanje sporočil izpisanih v terminalu, sva ustvarila funkcijo z imenom *printF,* ki sprejme dva parametra: *msg* oz. sporočilo ter *sev* (angl. severity oz. nujnost). Nujnost je lahko treh stopenj: 0 – obvestilo bele barve, 1 – opozorilo rumene barve ter 2 - napaka rdeče barve.

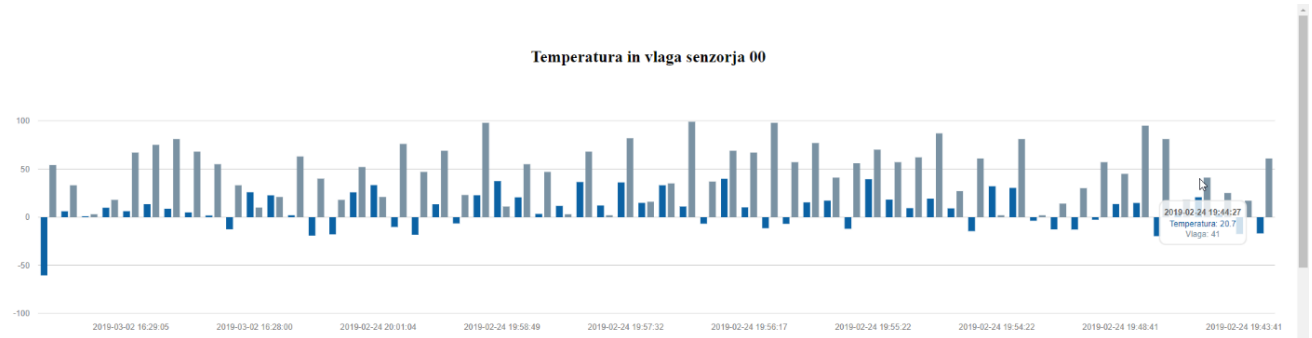
```
while True:
    data = arduino.flush()
    data = arduino.readline().decode("utf-8")
    print("["+time.strftime("%H:%M:%S")+"]"+data,  end="")
    p = data.split(";")
    if p[0] == "!01": #Vremenska postaja /Tip, ID, Temp, Vlaga
        con = mdb.connect('localhost', 'root', 'HelloWorld!', 'Weather_DB'); #IP, uporabnisko ime, geslo, ime DB
        with con:
            cursor = con.cursor()
            cursor.execute("SHOW TABLES")
            clear = False
            for x in cursor:
                if ''.join(x) == "WS"+p[1]:
                    if clear == False:
                        clear = True
            if clear == False:
                printF("No module with ID " +p[1], 2)
            if float(p[2]) < -100.0 or float(p[2]) > 100.0:
                    printF("Module with ID " +p[1] + " sent an invalid temperature: "+ p[2] + ". Check if the module is working correctly", 2)
                    clear = False
            if int(p[3]) < 0 or int(p[3]) > 100:
                    printF("Module with ID " +p[1] + " sent an invalid humidity: "+ p[3] + ". Check if the module is working correctly", 2)
                    clear = False
            if clear == True:
                if float(p[2]) < -25.0 or float(p[2]) > 45.0:
                    printF("Module with ID " +p[1] + " sent an unusual temperature: "+ p[2] + ". Check if the module is working correctly", 1)
                cursor.execute("INSERT INTO " + "WS" +p[1]+ " VALUES(%s,%s,%s)",(None,p[2], p[3])) #ID postaje, cas, temp, vlaga
                con.commit()
        cursor.close()
    else:
        printF("No module type with ID "+p[0],2)
```

Slika 52: Program za Raspberry Pi.

Glavni del programa se izvaja v *while true* zanki, kar pomeni da se zanka ponavlja v neskončnost. Najprej z funkcijo *flush()* program počaka, da vsa prihajajoča sporočila prispejo do konca. Nato sporočilo prebere ter ga kodira v UTF 8. Prejeto sporočilo nato skupaj z časom izpiše v terminal. Sporočilo se nato razdeli na dele, ločene s podpičjem. Nato se preveri, kateri tip modula pošilja sporočilo, če pa tak tip ne obstaja, se na terminalu izpiše napaka. Če je sporočilo s temperaturnega modula, se program poveže z vremensko podatkovno bazo. Nato preveri s katerega temperaturnega modula je sporočilo ter se poveže z primerno tabelo. Nato preveri, če je temperatura mogoča, torej če ni manjša od -100 ^{o}C ali večja od 100 ^{o}C. V tem primeru se podatki ne shranijo in v terminal se izpiše napaka. Isto se zgodi tudi v primeru, da je vrednost vlažnosti manjša od 0 % ali večja od 100%. V primeru da je vrednost temperature nenavadna, torej če je manjša od -25 stopinj ali večja od 45 stopinj Celzija, se izpiše opozorilo v terminal, da je potrebno preveriti, če modul deluje pravilno. Če so podatki v normalnih mejah, se podatki vpišejo v bazo, program zapre dostop do baze ter čaka novo sporočilo.

5.2.2.3 Spletna stran

Za preprost prikaz podatkov sva naredila spletno stran, napisano v programskem jeziku PHP 7. Spletna stran je lokalno gostovana na Raspberry Pi na osnovni odprtokodnega HTTP spletnega strežnika Apache 2. Sestavljena je iz dveh delov, to sta graf ter tabela z vrednostmi. Sicer oba predstavljata podatke, a graf služi za opazovanje celote ter trendov, tabela pa za natančen vpogled. Prikazujeta zadnjih 60 meritev, kar obsega približno 11 ur delovanja. Tabelo je razvrščena tako, da je na vrhu vedno najnovejši podatek. Graf ki je nared s pomočjo *morris.js* je interaktiven, saj ko pomaknemo miško na stolpec, izpiše točne vrednosti. Spletna stran je statična in ne dinamična, kar pomeni da če želimo nove podatke, je potrebno stran osvežiti ročno.



Čas	Temperatura (°C)	Vlaga (%)
2019-03-02 16:29:35	-60.5	54
2019-03-02 16:29:25	6.0	33
2019-03-02 16:29:20	0.8	3
2019-03-02 16:29:15	9.8	18
2019-03-02 16:29:05	6.1	67
2019-03-02 16:29:00	13.5	75
2019-03-02 16:28:50	8.6	81
2019-03-02 16:28:45	5.0	68
2019-03-02 16:28:35	1.7	55
2019-03-02 16:28:05	-12.7	33
2019-03-02 16:28:00	25.9	10
2019-03-02 16:27:50	22.6	21
2019-03-02 16:27:40	2.0	63
2019-02-24 21:26:01	-19.1	40

Slika 53: Spletna stran – Slika je simbolična.

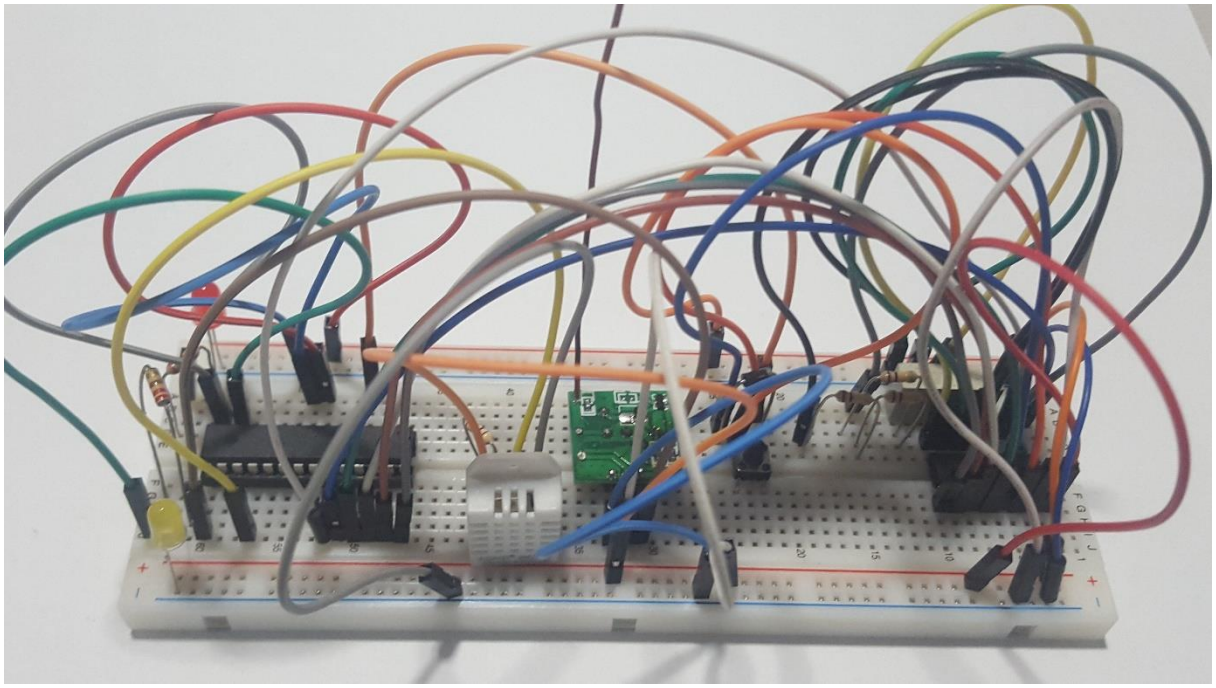
6 TESTIRANJE IN PREDSTAVITEV REZULTATOV

6.2 POTEK TESTIRANJA

Testiranje sva razdelila na pet delov.

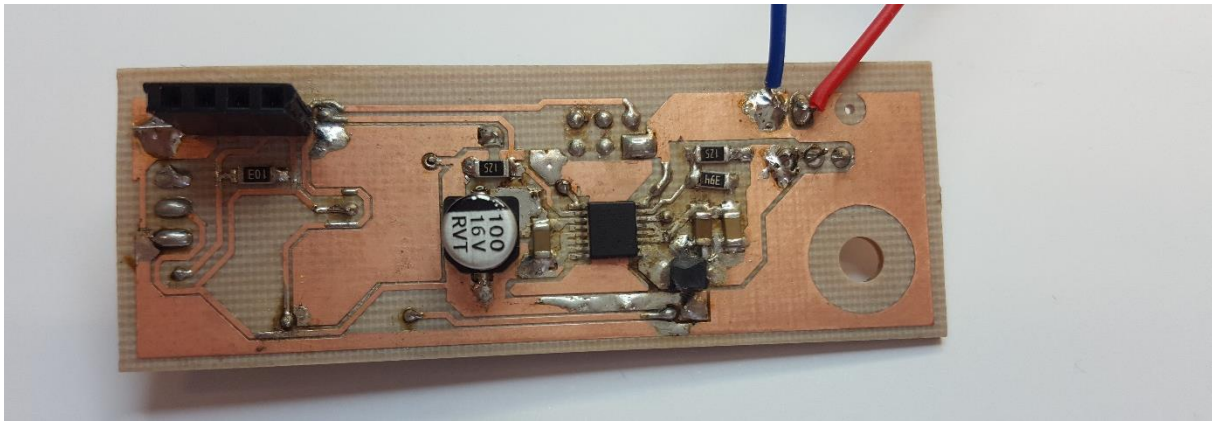
Prvi del je zajemal testiranje posameznih segmentov temperaturnega modula na prototipni ploščici (ang. breadboard). Najprej sva preizkusila mikrokrmilnik ATMega328p in programiranje s pomočjo ICSP kontaktov. Nato sva ločeno preizkusila dekadni števec in tipko za bujenje. Uporabila sva števec CD4536, ki je identičen MC14536, le da je namesto SO16 v DIP-16 izvedbi. Takrat sva preizkusila še program za varčevanje z energijo in bujenje mikrokrmilnika. Za tem sva ločeno preizkusila še RF modul in led indikator. Regulatorja tu nisva preizkusila, ker obstaja le v TSSOP izvedbi, ki je na prototipni ploščici nisva mogla preizkusiti.

V drugem delu sva na prototipni ploščici prej omenjene segmente uporabila skupaj. Na ATMega328p sva priključila ostale komponente temperaturnega modula. Iz istega razloga kot prej tu nisva preizkusila regulatorja. V tej fazi sva lahko izmerila tok, ki bi tekel iz regulatorja, če bi bil prisoten. Med spanjem porabi okoli 100 µA, kar je malo več od pričakovanega.



Slika 54: Vezje na prototipni ploščici.

V tretjem delu sva preizkusila delovanje centralnega sistema in komunikacije med moduloma. Takrat sva sestavila še spletno stran in SQL bazo za shranjevanje podatkov.

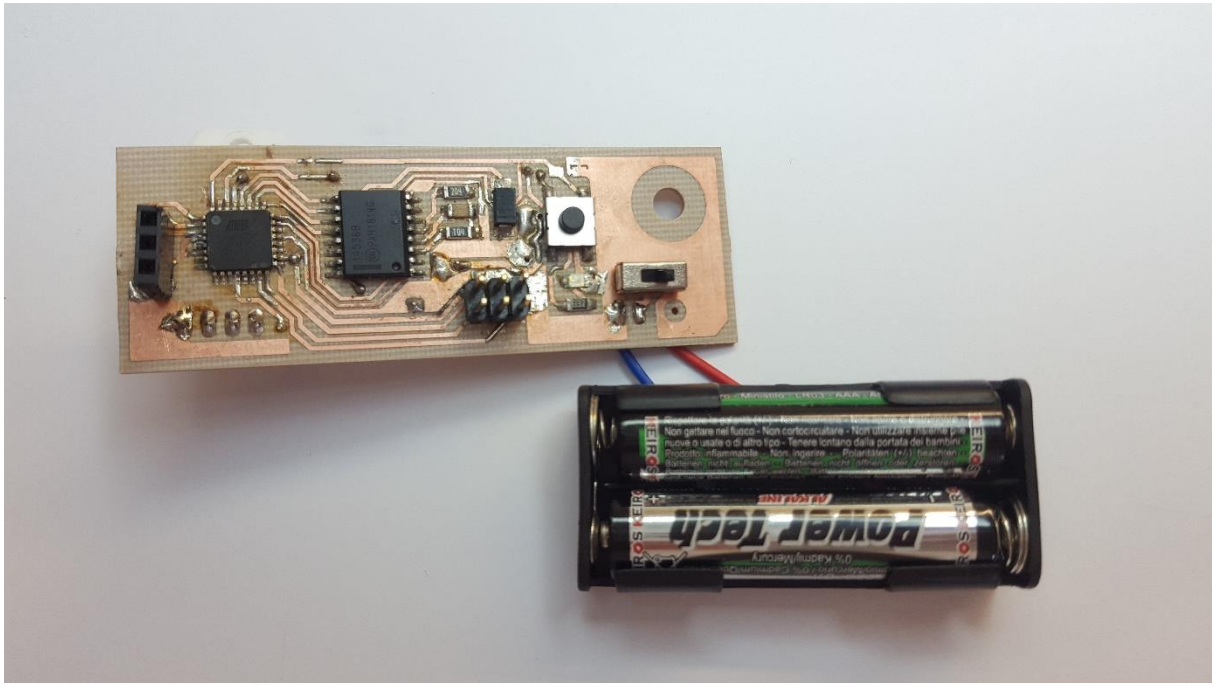


Slika 55: Regulator na PCBju.

Pod četrti del spada preizkus delovanja regulatorja. Preizkus sva opravila tako, da sva na PCB ploščico, opisano pod točko 4, prispajkala le komponente regulatorskega vezja. Ob vhodni napetosti 3 V je bila izhodna napetost enaka 5 V, kar pomeni, da je regulacijsko vezje delovalo.

Zadnji del testiranja je bil preizkus delovanja vseh komponent na PCB ploščici. Na ploščico z regulatorjem iz prejšnjega dela sva prispajkala še vse ostale komponente. Tu so se pojavile težave. Ob priključitvi na napajanje je regulatorju izhodna napetost začela nihati med 1.5 V in 2.5 V. Vhodni kondenzator C_{IN} se je začel močno segrevati. Tudi, ko sva ga zamenjala z novim, težave nisva rešila.

Tako sva se odločila, da bova preizkusila PCB vezje brez regulatorskega vezja. Napajala sva ga s 5 V preko ICSP kontaktov. Ugotovila sva, preostalo vezje deluje kot predvideno.



Slika 56: Preizkus ostalih komponent vezja.

6.3 SKLEPI IN HIPOTEZE

Končni izdelek torej ne deluje v celoti, v celoti pa je delujoča verzija na prototipni ploščici in končna verzija centralnega modula.

Da bi končni izdelek deloval v celoti, bi bilo potrebno najti napako v regulatorju in sestaviti novo vezje.

Potrdiva lahko naslednje hipoteze:

- *vezje bo imelo večino komponent v tehnologiji SMD;*
- *sistem bo omogočal razširitev na več senzorjev oz. modulov, brez da bi bilo ob tem potrebno vse preprogramirati;*
- *računalnik Raspberry Pi bo podatke shranjeval v SQL bazo.*

Zaradi težav z regulacijskim vezjem nisva mogla izmeriti dejanske porabe. Tako ne moreva potrditi zadnje hipoteze:

- *Baterije senzorja ne bo potrebno zamenjati vsaj pol leta.*

Ob kritičnem pogledu na potek najinega raziskovanja in izdelave sva prišla do več sklepov oz. ugotovitev:

- Regulator bi bilo potrebno predhodno preizkusiti po obremenitvijo.
- Med spajkanjem sva ugotovila, da bi komponente morale med sabo imeti več prostora, kar bi olajšalo spajkanje. PCB ploščica bi brez težav lahko bila večja.
- Pred izdelavo končne PCB ploščice bi lahko izdelala module posameznih segmentov vezje. To pomeni razdelitev vezja na regulacijski modul, modul s števcem in tipko, modul s mikokrmilnikom in ICSP kontakti, RF modul ter senzor DHT22. Ti moduli se bi nato priključili na prototipno ploščico oz. kar en v drugega. S tem pristopom bi iskanje napak postalo dosti lažje.
- Koristni bi bili tudi TX in RX konektorji na PCB ploščici, kar bi s pomočjo serijskega povezave olajšalo iskanje napak v programu.
- Za zanesljiv dostop do spletne strani izven domačega omrežja, bi morala spletno stran gostovati na zunanjih strežnikih, z spletno domeno.

7 ZAKLJUČEK

Naloga se je izkazala za bolj kompleksno, kot pa sva pričakovala. Na koncu so se pojavile napake, katerih zaradi pomanjkanja časa nisva uspela odpraviti.

Če bi sistem nadgrajevala, bi najprej zamenjala RF modul. Čeprav je preprost za uporabo v takem sistemu, je premalo zanesljiv, da bi deloval na daljše razdalje, še posebej, če mora signal potovati skozi stene. Prav tako bi lahko v prihodnosti s pomočjo 3D tiskalnika natisnila ohišje. Mogoče bi tudi zamenjala tudi senzor DHT22 zaradi počasnega odziva na spremembe vlage in temperature.

Kljub težavam pa sva z rezultati zadovoljna. Z izjemo regulatorja deluje vezje na prototipni ploščici in na PCB ploščici. Centralni modul v celoti deluje, kot sva si želela. Prav tako deluje tudi brezžični prenos podatkov med moduloma. Celoten sistem je po najinem mnenju ob odpravi težav, povezanih z ovrženo hipotezo, primeren za uporabo v praksi

8 VIRI

Gumbne baterije. Citirano 16. 2. 2019. Dostopno na: https://www.batteries.com/pages/coin-cell-button-cell-battery-guide

Orodje za načrtovanje napajalnih vezij Texas Instruments Webench Power Designer. Uporabljeno dne 16. 2. 2019. Dostopno na: https://webench.ti.com/power-designer

TPS61032PWP regulator napetosti. Citirano 19. 2. 2019. Dostopno na: http://www.ti.com/product/TPS61032

Dokumentacija za TPS61032PWP regulator napetosti. Citirano 19. 2. 2019. Dostopno na: http://www.ti.com/lit/ds/symlink/tps61032.pdf

9 V baterije. Citirano 19. 2. 2019. Dostopno na: https://en.wikipedia.org/wiki/Nine-volt_battery

AAA baterije. Citirano 19. 2. 2019. Dostopno na: https://en.wikipedia.org/wiki/AAA_battery

Dokumentacija za MC14536 dekadni števec. Citirano 19. 2. 2019. Dostopno na: https://www.onsemi.com/pub/Collateral/MC14536B-D.PDF

Dokumentacija za ATMega328p mikrokrmilnik. Citirano 19. 2. 2019. Dostopno na: https://www.microchip.com/wwwproducts/en/ATMega328p

Izdelava novih komponent v programu Eagle. Citirano 20. 2. 2019. Dostopno na: https://www.autodesk.com/products/eagle/blog/library-basics-part-1-creating-first-package-autodesk-eagle/

Smernice za preprečevanje elektromagnetnih motenj v vezju. Citirano 25. 2. 2019. Dostopno na: https://training.ti.com/engineer-it-leverage-tis-online-toolbox-tackle-emi-touching-soldering-iron

Smernice za uporabo stikalnih regulatorjev. Citirano 2. 3. 2019. Dostopno na: http://www.ti.com/lit/an/slva773/slva773.pdf

Programska knjižnica za branje podatkov z senzorja DHT22. Citirano 2. 3. 2019. Dostopno na: https://github.com/adafruit/DHT-sensor-library ter https://github.com/adafruit/Adafruit_Sensor

Programska knjižnica za brezžično pošiljanje podatkov Virutal Wire. Citirano 2. 3. 2019. Dostopno na: https://web.archive.org/web/20170311002258/https://www.pjrc.com/teensy/td_libs_VirtualWire.html

Program za izdelavo grafa na spletni strani moris.js. Citirano 2. 3. 2019. Dostopno na: https://morrisjs.github.io/morris.js/

ZAHVALA

Najprej se bi rada zahvalila mentorju, g. Mateju Kališku, inž. el., za koordinacijo celotne raziskovalne naloge in strokovne nasvete. Zahvala gre tudi g. Janku Holobarju inž. el. za rezkanje tiskanega vezja in nasvete pri načrtovanju. Prav tako gre zahvala g. Gregorju Kramerju, univ. dipl. inž. el., za usmerjanje in pomoč pri reševanju problemov.

IZJAVA

Mentor Matej Kališek v skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom RF nadzorni modul, katere avtorja sta: Kristjan Šoln Gašper Gril

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 6.3.2019

ŠOLSKI CENTER CELJE SREDNJA ŠOLA ZA KEMIJO, ELEKTROTEHNIKO IN RAČUNALNIŠTVO CELJE, Pot na Lavo 22 1

Žig šole

Podpis mentorja

Podpis odgovorne osebe

POJASNILO

V skladu z 2. in 17. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.