Osvetljenost stene (tema 39) — opis projekta

Projektno delo pri fiziki, Gimnazija Bežigrad

Profesor: prof. Peter Gabrovec
Avtor: Anton Luka Šijanec, 1. a

23. April 2020

Povzetek

Ta dokument opisuje izvedbo in pripravo na merjenje osvetljenosti stene s senzorjem BH1750 in procesorjem ESP8266. Opis vključuje postopek programiranja in sestavljanja senzorja, njegovo uporabo, probleme takega merjenja in nekaj osnovnih rezultatov meritve.

Kazalo vsebine

1 Uvod **1**

2 Seznami **1**
2.1 Komponente za izdelavo senzorja . . . 1
2.2 Merjene količine . . . 2

3 Ideja in izvedba **2**
3.1 Shematski prikaz povezav . . . 2
3.2 I^2C . . . 3
3.3 Programiranje . . . 3
3.3.1 Nalaganje programov . . . 3
3.3.2 Kontroliranje . . . 3
3.3.3 Shranjevanje podatkov . . . 3
3.3.4 Čas . . . 4

4 Osvetljenost **5**

5 Obdelava podatkov **5**

1 Uvod

Za ta projekt sem se odločil, ker mi je omogočil še največ samostojnega dela in sem ga izdelal na način, ki mi je bil v zabavo. Za merjenje osvetljenosti sem se odločil sam izdelati uporabniku še najbolj prijazen senzor kot celoto, primeren za dolgotrajno merjenje v kakršnihkoli pogojih brez prisotnosti človeka, npr. za uporabo v oddaljenih krajih za odčitavanje vremenskih pogojev, z uporabo dostopnih delov, ki se jih da relativno hitro in po smešno nizkih cenah dobiti iz Kitajske.

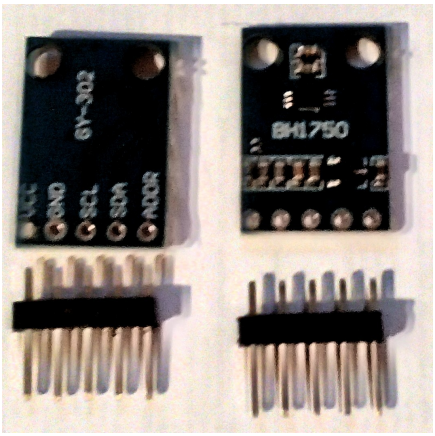
Za popestritev sem projektu z namenom izdelave korelacij med svetlobo in drugimi vremenskimi dejavniki dodal še Boschov senzor temperature, vlage in pritiska.

2 Seznami

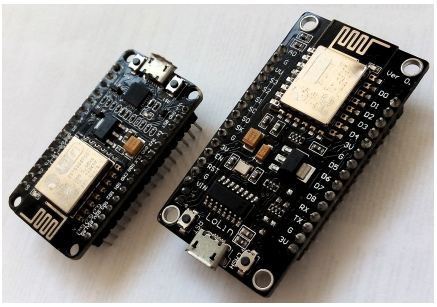
2.1 Komponente za izdelavo senzorja

Ker za izdelavo tiskanih vezij nisem imel časa, volje in denarja, sem kupil že-natisnjena vezja z že-prispajkanimi komponentami iz spletne trgovine Aliexpress.

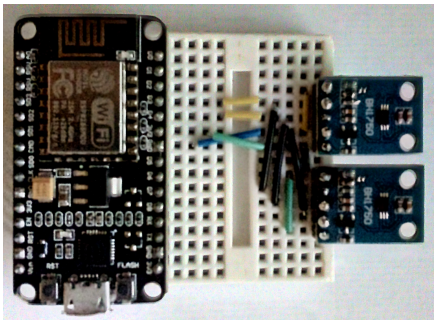
- NodeMCU 1.0 z ESP-12E modulom (ESP8266 procesor in 4MiB flash shrambe)
- 2x GY-302 z BH1750 modulom
- 2x GY-BME/BMP280 z BME280 modulom



Slika 1: BH1750



Slika 2: NodeMCU 1.0 in NodeMCU 0.9



Slika 3: Procesor s svetlobnima senzorjema

2.2 Merjene količine

- osvetljenost (E) v luxih
- čas (t) v sekundah
- dodatno sem meril še: temperaturo (T) v stopinjah C, vlago zraka v odstotkih in zračni pritisk v hPa

Odvisna spremenljivka je osvetljenost; neodvisna pa čas.

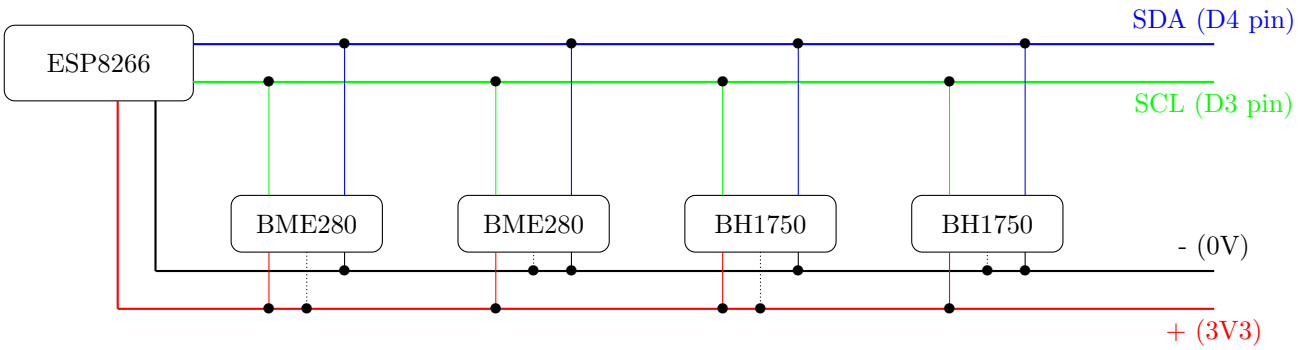
3 Ideja in izvedba

Za bolj natančne meritve sem temperaturo, vlago in pritisk meril z dvema merilnikoma, prav tako sem osvetljenost meril z dvema merilnikoma osvetljenosti. Kasneje se je sicer izkazalo, da so odstopanja izredno majhna in da so vsi podatki v skladu s specifikacijami proizvajalca in dva merilnika sploh ne bi bila potrebna.

Procesor ESP8266 sem izbral zaradi integriranega WiFi 2,4 GHz sevalnika, ki mi je omogočil enostavno upravljanje s senzorjem na daljavo brez potrebe po kablih ali dodatnih sevalnikih. Ta čip ima med drugim tudi dobro prepoznavnost in veliko odprtokodnih programov že napisanih, predvsem zaradi odprtokodnih programov in SDK-jev, ki jih ponuja proizvajalec Espressif (Šangkaj).

Prav ta dva modela merilnikov sem si izbral zato, ker imata že napisane in odprte programske knjižnice za Arduino platformo ter ker omogočata povezavo po I^2C .

3.1 Shematski prikaz povezav



3.2 I^2C

Kot je razvidno iz skice povezav, so vse štiri naprave v procesor priklopljene s samo dvema žicama na samo dva pina. Kot sem omenil v uvodu, sem si izbral take module, ki uporabljajo digitalni protokol I^2C , ki omogoča komunikacijo po principu *master—slave* (gospodar—suženj). Procesor je torej deloval kot *master*, merilniki pa kot *slaves*. Četudi so vse naprave prižgane, na kablih ne bo ničesar, dokler procesor ne pošlje ukaza neki napravi.

Za serijsko komunikacijo se uporablja *clock* in *data* modulacija za digitalni prenos podatkov. Žica, ki je določena kot *clock* oziroma ura, bo med prenosom podatkov neprestano menjavala digitalno stanje med visoko napetostjo (3V3) in nizko (0V). Vsakič, ko bo stanje ure 1 (visoka napetost), bo prejemnik (v tem primeru vsi štirje merilniki) prebral stanje na podatkovni žici, spet bodisi 1 (visoka napetost) ali 0 (nizka napetost). Tako lahko uporabljamo teoretično neomejeno število podatkovnih žic in samo eno žico za uro. Prav tako je ta način komunikacije zelo uporaben z vidika kompatibilnosti, saj lahko naprave delujejo na spremenljivih in neodvisnih frekvencah.

Pa pojdimo nazaj k I^2C . Vsaka naprava ima določen 7 bitni naslov, ki je enak za vsako enoto nekega modela modula. Torej bo vsak BME280 vedno imel naslov 0x76 (binarno 1110110). Da procesor pridobi podatek o npr. osvetljenosti, bo po žici najprej poslal naslov naprave, s katero hoče komunicirati, nato pa določen ukaz, ki je opisan v specifikacijah (angleško *datasheetu*) te naprave. Za naš senzor osvetljenosti (BH1750) bo torej poslal naslov 0x23 (0100011) in ukaz 00010001. Ker *slave* naprave ponavadi nimajo generatorja pulza, ki bi generiral uro, mora za to poskrbeti *master*. Pulz ure na SCL (*clock*) žici mora biti v nekem veljavnem območju, v katerem merilnik deluje, drugače lahko pride do napačnih ali napačno sprejetih podatkov. Po zaznavanju tega ukaza bo naprava odgovorila s podatkom o osvetljenosti. V našem primeru je to 16 bitna številka, torej naprava sporoča podatke v območju 0 – 65535 luxov. Nato lahko gostitelj znova vpraša neko drugo napravo.

Če pozorno berete, ste opazili, da sem omenil, da ima vsak model naprave le en določen naslov. V našem primeru bi torej na eni SDA (podatkovni) žici imeli lahko le eno enoto vsakega modela merilnika. Na srečo proizvajalci ponavadi vključijo nek uporabni način primitivne spremembe naslova. Na shemi povezav vidite črtkane povezave do vsakega merilnika, vsaka enota posameznega modela pa ima obraten signal od druge enote istega modela. Torej bo en BH1750 imel vase vpeljan konstantno visok signal, drug BH1750 pa konstantno nizek signal. BH1750 bo glede na ta vhodni signal (imenujemo ga ADDR signal) nastavil svoj I^2C naslov tako, da bo vsak bit naslova obrnil (0100011 — 0x23 postane 1011100 — 0x5C), BME280 pa bo naslov povečal za 1 (1110110 — 0x76 postane 1110111 — 0x77).

3.3 Programiranje

Programe sem pisal v okolju Arduino, saj je eno izmed največkrat uporabljenih odprtokodnih in brezplačnih orodij za programiranje mikročipov in je dobro podprto s strani proizvajalcev mikročipov, ki brezplačno ponujajo odprte programske knjižnice za njihove produkte.

Program sem objavil na `https://github.com/sijanec/sijaneciot` in je bil zgrajen na mojem obstoječem ogrodju za ESP8266 mikrokontrolerje.

3.3.1 Nalaganje programov

ESP8266 omogoča žično povezavo prek UART protokola za programiranje. Ker večina današnjih računalnikov serijskih portov več nima, sem kupil NodeMCU 1.0, ki ima že vgrajen CP2102 čip. CP2102 je UART oddajnik/sprejemnik, ki omogoča povezavo prek USB. Tako sem torej lahko enostavno nalagal programe prek USB povezave. Kmalu sem sicer v program dodal brezžično posodabljanje programa (*over-the-air upgrade*), kar je še dodatno olajšalo razvijanje programske opreme za senzor in bi v scenariju neke masovne proizvodnje omogočalo hkratno posodabljanje veliko naprav prek Interneta.

3.3.2 Kontroliranje

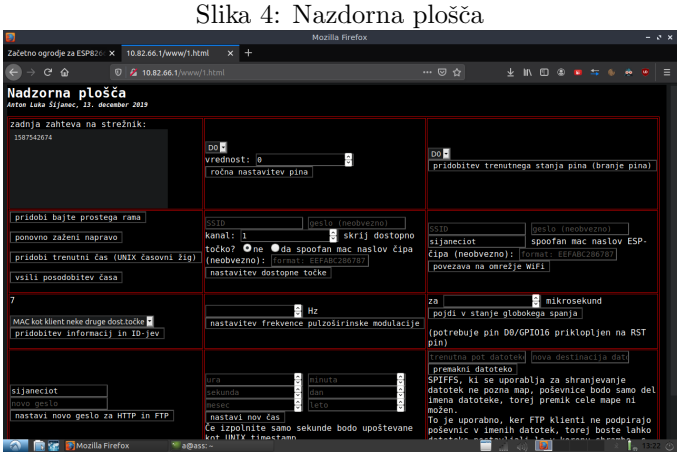
Ko mi je uspelo komunicirati z merilniki, sem se lotil izdelave načina za nastavljanje in dostop do osnovnih funkcij senzorja, kot so npr. avtomatsko pošiljanje podatkov o vremenu, sinhronizacija časa, in tako dalje. Odločil sem se uporabiti aplikacijski protokol HTTP, ki se uporablja za prenos spletnih strani in omogoča preprosto povezavo z uporabo brskalnika. HTTP uporablja transportni protokol TCP, ki je na ESP8266 dobro podprt. Preko HTTPja sem omogočil tudi oddaljeno posodobitev in prejemanje tabele z rezulati meritev.

3.3.3 Shranjevanje podatkov

SPIFFS Ker sem imel 4MiB *flash* shrambe in sem za program porabil zgolj približno 300kB prostora, za začasni prostor za brezžično posodobitev 1MB in za delovni spomin 128kB, sem v 2MB shrambe alociral particijo SPIFFS.

SPIFFS je pogosto uporabljen datotečni sistem, ki ponuja podobne ukaze, kot kateri koli drugi. Namenjen je za mikročipe, ker porabi zelo malo prostora za tabele z datotekami, saj ne shranjuje metapodatkov, kot so datumi sprememb, in ne dovoli uporabe direktorijev oziroma map. Tako so vse datoteke direktno na korenu shrambe, kar je

Slika 4: Nazdorna plošča



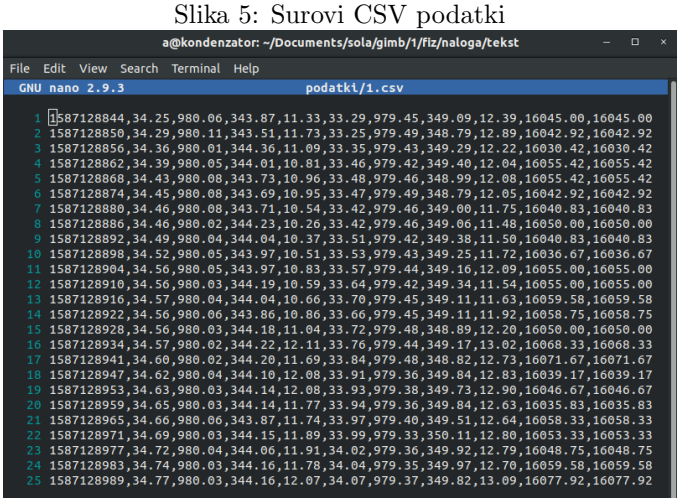
povsem zadostno, saj datotek ponavadi ni veliko. Za kompatibilnost s HTTP protokolom in FTP protokolom (tudi FTP sem uporabljal), sem datoteke poimenoval, kot da bi bile v mapah; vse datoteke za spletno stran so se začele z /www/, vsi podatki senzorjev so se začeli z /www/var/ in konfiguracijske datoteke so se začele z /403/. Tako se končnemu uporabniku prikaže neka iluzija map in podmap.

CSV CSV oziroma *Comma seperated values* format datotek je zelo preprost način shranjevanja podatkov v tabele, in se ga da odpreti v večini statističnih programov. Ima povsem enako strukturo kot celice v Excel preglednici; vrstice so ločene z znakoma 0x0D0A (začetek vrste; nova vrsta), posamezne celice pa z vejico (0x2C).

Vsak izmerek je bil v svoji vrstici. V prvem stolpcu je bil UNIX časovni žig (32-bitno podpisano celo število sekund od 1. januarja 1970 UTC), nato pa so bili po vrsti podatki o temperaturi, pritisku, vlagi obeh senzorjev in nato še podatka o svetlobi.

Podatke sem meril z pet sekundnim intervalom, saj sem imel to možnost. Sicer je to povsem nepotrebno, ker se osvetljenost stene v časovnem razmaku pičlih petih sekund ne spremeni prav nič. Ker sem imel 160MHz procesor, merilnik osvetljenosti, ki osvetljenost izmeri v 120 milisekundah, in **dva megabajta** shrambe, sem si to lahko privoščil.

Slika 5: Surovi CSV podatki



3.3.4 Čas

Računalniki za shranjevanje časa ponavadi uporabljajo neko zunanjo uro, ponavadi je to zelo natančen kvarčni kristal. Jaz take sofisticirane priprave nisem imel, zato sem uporabil števec ciklov procesorja. Ob vsakem ciklu procesorske ure se bo nek notranji števec povečal za 1. To številko lahko preberemo, vendar ni povsem natančna, ker procesorska ura ne teče vedno točno na predpisanih 80MHz ali 160MHz (možne nastavitve ESP8266). Prav tako se ta števec z *resetom* naprave ponastavi in je velik le 32 bitov, torej lahko pri 160 obratih na sekundo najdlje zdrži 26.843545 sekund, preden začne spet šteti od 0. V 26 sekundah bomo sicer ta obrat zaznali in zabeležili, torej to ni problem.

Zato sem uro sklenil posodobiti iz Interneta vsake toliko časa. V Sloveniji ima natančno uro Arnes, vendar omogoča dostop do ure samo po standardnem NTP aplikacijskem protokolu, ki uporablja transportni protokol UDP. UDP je na ESP8266 programju slabo podprt, zato sem sklenil uporabljati TCP.

Vsak spletni strežnik (HTTP) namreč pošlje uro ob vsakem odgovoru. To sem uporabil za sinhronizacijo ure. Vsakih 30 sekund, ko ima senzor dostop do Interneta, se poveže na Arnesov datotečni strežnik in prebere uro. Program tudi avtomatsko preverja *drift* ure (koliko zaostaja) in se na to prilagodi. Konec koncev ne potrebujemo silno natančne ure, vendar je to v pogojih, ko Internetnega dostopa ni po več mesecev, ključno.

4 Osvetljenost

Sprva sem mislil osvetljenost meriti z navadnim *LDR — light dependent resistorjem* oziroma upornikom, ki manjša upornost glede na osvetljenost. Ampak upornost v Ω ni direktno pretvorljiva v luxe, prej bi bila potrebna neka kalibracija. Ker nisem imel nobenega drugega merilca osvetljenosti, s katerim bi opravil kalibracijo, sem kupil omenjeni že-kalibrirani merilec, ki mi poda vrednost osvetljenosti direktno v luxih.

Na začetku sem merilec uprl direktno v sonce. Direktna sončna svetloba je glede na pozicijo na Zemlji od 60 tisoč do 100 tisoč luxov, zato je moj merilnik vedno kazal najvišjo vrednost, ki jo lahko zanesljivo izmeri; 32727 luxov. Merilnik sem zato obrnil proti steni, da se je svetloba najprej odbila od stene in nato v merilnik; tako sem meril odbito svetlobo.

Slika 6: Okolica in položaj senzorja

5 Obdelava podatkov

Podatkov se je nabralo kar veliko. Vsak zapis je uporabil 80 bajtov, to je z 20 tisoč zapisi na dan že kar nevarno blizu 2MB meje, zato sem si vsak večer shranil zapise na računalnik in jih pobrisal iz senzorja.

Risanje v Excelu odpade, ker ta podpira maksimalno 255 podatkov na XY grafu. Za testiranje sem uporabljal Excelu alternativni program Calc iz zbirke odprtokodnih programov LibreOffice. Za hitrejše risanje sem najprej uporabil odprtokodni programski statistični jezik R, vendar ga nisem obvladal tako dobro, da bi mi uspelo lepo narisati grafe. Premium alternativ, kot so MATLAB, pa ne podpiram zaradi njihovih visokih cen. Na koncu sem se odločil grafe risati z LaTeX paketom `pgfplots`. CSV datoteke je interpretiral `gnuplot`.

Zapise, ki so imeli časovni žig, nižji od milijona, sem izpustil, saj so nastali takoj po ponovnem zagonu naprave, preden se je sinhroniziral čas in niso imeli pomena. To sem naredil z ukazom `awk -F, '$1>=1000000' weather.csv > clean2.csv`.

Neodvisna količina, torej osvetljenost, se je spreminjala v razponu od 0 (ponoči), do okoli 20 tisoč (jasno vreme) in je imela natančnost četrtinke luxa. Merjena je bila z dvema merilnikoma in oba sta kazala zelo podobno vrednost (večinoma enako), torej napak ni bilo.

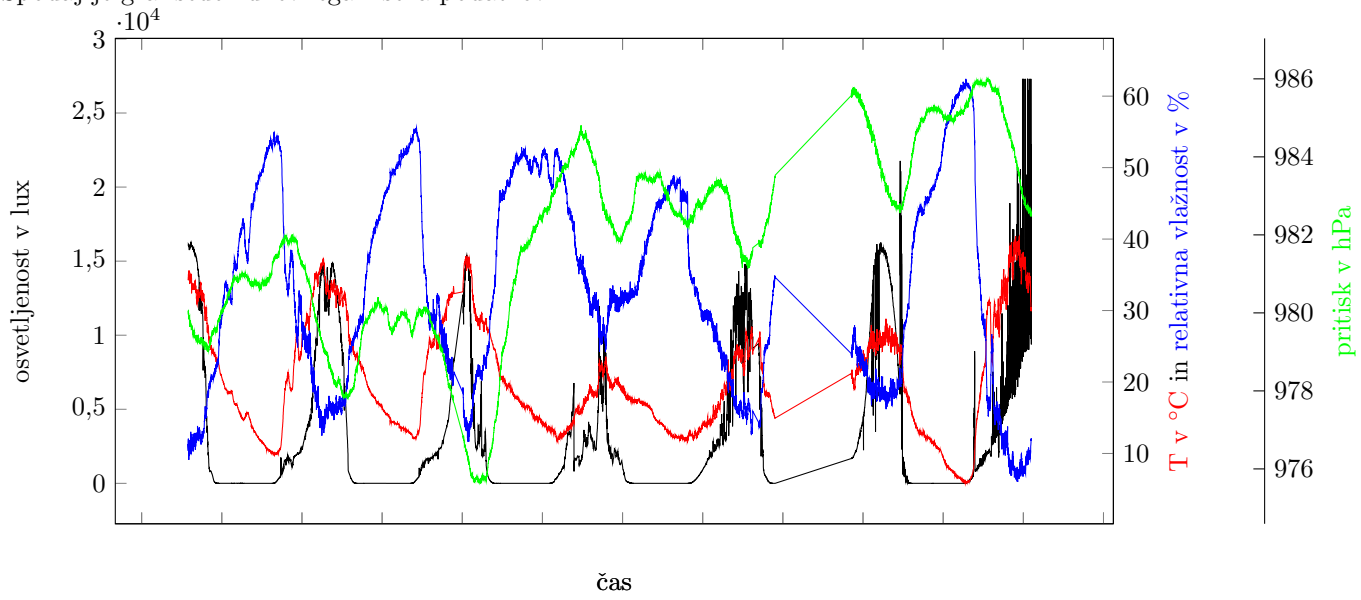
Osvetljenost sem in bom narisal na grafih skupaj z temperaturo, vlago in pritiskom glede na čas, saj bi naloga, če ne bi nečesa primerjal, ne imela pomena.

Absolutna napaka meritve je bila v skrajnem primeru pol luxa v optimalnih pogojih, ponavadi pa sta oba senzorja kazala enako vrednost na četrtinko luxa natančno. Povprečna napaka izmerjene temperature je okoli 1 °C.

Za hitrejše generiranje sem vseeno v grafe vključil zgolj vsak 8. zapis in s tem čas med meritvami povečal na 40 sekund.

Prečiščena tabela rezultatov je dostopna na `http://razor.arnes.si/~asija3/files/sola/gimb/1/fiz/naloga/tekst/clean.csv`.

Spodaj je graf sedemdnevnega izseka podatkov.



Opomba — dodano 23. aprila: Sedmi dan (23. aprila) je bila izvedena meritev pod napačnimi pogoji. Aluminijasta folija, ki sem jo uporabljal kot zaščito pred soncem, da svetloba ni toliko spreminjala merilnika, je zaradi vetra ponoči deloma odpadla, zategadelj so sončni žarki direktno sijali na senzor temperatur, ki je zato pokazal napačne zapise. Prav tako se je svetloba iz aluminijaste folije odbijala direktno v senzor svetlobe, zato se pojavijo meritve tudi v območju $3 \cdot 10^4$lx. Seveda zato sedmega dne ne smemo upoštevati kot dneva s pravilnimi podatki o temperaturi in o osvetljenosti. Podobne artefakte najdemo tudi 21. aprila, le da je bil razlog zanje oblačen dan.