IoT stanica za praćenje temperature i vlage unutar kuće

Brodar, Vedran

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Stručni studij

IoT stanica za praćenje temperature i vlage unutar kuće

Završni rad

Vedran Brodar

Osijek, 2021.

SADRŽAJ

1. UVOD

Nagli porast internetskih tehnologija, kao i veliko povećanje broja korisnika te uređaja koji su povezani na različite načine dovelo je do toga da povezujemo te iste uređaje u mrežne infrastrukture. Te infrastrukture omogućavaju raznim vrstama uređaja (npr. razni senzori za vlagu, temperaturu i sl.) komunikaciju i prijenos podataka na različite načine. Neki od primjera tih uređaja su pametne klupe, hladnjaci, perilice i sl. Za primjer, hladnjak koji putem aplikacije može poslati poruku da treba kupiti neki proizvod koji se potrošio ili navodnjavanje staklenika pomoću senzora koji detektira razine vlage u zemlji i stakleniku u kojoj se biljke nalaze. Osim kućanskih aparata to još mogu biti i uređaji poput POS aparata, bankomata, uređaja za magnetsku rezonancu i ostali.

Samim integriranjem uređaja ostvaruju se razne pogodnosti koje značajno poboljšavaju životni standard (npr. kontroliranje termostata, elektroničkih uređaja, osvjetljenja ili nečeg drugog preko mobitela ili drugih pametnih uređaja), štede dragocjeno vrijeme te omogućavaju pojednostavljenje prijenosa raznih vrsta podataka od uređaja na uređaj.

Sam IoT (engl. *Internet of things*) je krenuo naglo rasti upravo zbog kontroliranja gotovo svih aspekata unutar kuće pomoću jednog uređaja kao što je mobitel (tzv. "Pametne kuće"), pristupati svim kamerama u kućanstvu zbog sigurnosnih razloga , isto kao i u raznim ugrađenim sustavima poput bolničkih ili poljoprivrednih sustava.

U ovoj temi govorit će se o tome kako je moguće povezati nekoliko takvih uređaja: Raspberry Pi koji je ugrađeni (engl. *embedded)* sustav baziran na Linuxu, te nekoliko Arduino Nano koji su mikrokontroleri (engl. *microcontrollers*), zajedno sa nekoliko senzora koji omogućavaju komunikaciju među njima te razmjenu podataka. Sam cilj ove teme je omogućiti komunikaciju između navedenih uređaja, vršiti očitavanja temperature i vlage te njihovih omjera u raznim prostorijama u kućanstvu i na temelju tih informacija donositi odluke o tome da li se treba prozračiti određenu prostoriju ili ne.

Ovaj završni rad razrađen je u tri cjeline. U drugom poglavlju objašnjen je pregled područja i slična rješenja. U trećem poglavlju naziva „*O tehnologijama*“ su objašnjene tehnologije i sklopovlja koja su korištena za izradu rada, dok u četvrtom poglavlju naziva „*Projekt vremenske stanice unutar kuće*“ je objašnjena upotreba aplikacija i uređaja.

2. SLIČNE APLIKACIJE

Osim ovog završnog rada postoji i nekolicina sličnih projekata koji se razlikuju po nekim komponentama, tehnologijama ili načinu pristupa problemu. Primjeri takvih projekata su:

- „*IOT meteorološka stanica*“, Knežević Lovro, FER.
- „*Prikupljanje senzorskih podataka putem AWS IoT platforme*“, Mijić Andrija, FER.
- „*IoT aplikacija za pametni vrt*“, Gaćina Luka, Sveučilište u Splitu.
- „*IoT i pametna kuća*“, Obadić Sven, FERIT.

Primjer „*IOT meteorološka stanica“* koristi senzor DHT22 koji prikuplja temperaturu i vlagu kao i u ovom radu, ali koristi SAP Hana mikrokontroler i SAP cloud platformu za prijenos podataka što čini veliku razliku u pristupu naspram ovog rada [1]. Slično je rješenje i u radu „*Prikupljanje senzorskih podataka putem AWS IoT platforme*“ gdje se koristi sličan senzor za radio-frekvencijsku komunikaciju dok se aplikacija izvodi na Android platformi a sva razmjena podataka ide preko raznih Amazonovih servisa [2]. „*IoT aplikacija za pametni vrt*“ koristi ESP8266 NodeMCU razvojnu pločicu sličnu kao što je Arduino u ovom radu, ali se također sav prijenos podataka vrši preko servisa naziva Blynk [3]. „*IoT i pametna kuća*“ teoretski opisuje koja su moguća rješenja za IoT koristeći trenutno dostupne tehnologije [4].

Vidi se sličnost korištenja senzora za temperaturu i vlagu, te razvojnih pločica koje su slične onima koji su korišteni u ovom radu, ključna razlika je u načinu prijenosa podataka koji je ovim radom riješen spremanjem podataka u lokalnu bazu, dok se gore navedeni radovi koriste gotovim servisima čija je namjena prikupljanje, razmjena podataka te uvid u iste.

3. O TEHNOLOGIJAMA

Tehnologije koje se koriste u ovom projektu su slijedeće: Raspberry Pi 4 Model B, Arduino Nano, DHT22 senzor za temperaturu i vlagu [5], NRF24L01 primopredajnik za komunikaciju (engl. *Transceiver)* , Arduino sat koji pokazuje stvarno vrijeme (engl. *Real-time clock – RTC*) DS3231.

3.1. Što je IoT?

IoT je koncept koji podrazumijeva spajanje bilo kojeg uređaja na internet i/ili s drugim uređajima. Možemo to zamisliti kao veliku mrežnu infrastrukturu koja uključuje milijarde uređaja koji prikupljaju i međusobno razmjenjuju podatke.

Sam naziv Internet of things je prvi koristio Kevin Ashton koji je često nazivan izumiteljem IoT-a. On je prvi put koristio taj termin 1999. za opisivanje sustava u kojem je "*Internet spojen na fizički svijet preko sveprisutnih senzora*". Do tog naziva je došao dok je radio eksperimente na Tehnološkom institutu Massachusetts (engl. *Massachusetts Institute of Technology*) gdje je pokušavao unaprijediti sustav korištenjem radio frekvencijske identifikacije (engl. *Radio-frequency identification – RFID*). Smatrao je da ako svi objekti koje koristimo u svakodnevnom životu budu imali identifikatore i mogućnost spajanja na internet, ti će objekti moći međusobno komunicirati jedni s drugima, dok će se njima upravljati preko računala.

Ciscova internet grupa je predvidjela da će do 2020. godine na mreži biti spojeno preko 50 milijardi uređaja koji su međusobno spojeni putem interneta, WIFI-a, Bluetootha itd.

Tri ključna čimbenika koja čine IoT su:

1. Spajanje objekata i živih bića.
2. Senzori.
3. Promjena tipa uređaja koji komuniciraju u mreži [6].

Kod spajanja objekata i živih bića misli se na uređaje kao što su pametni satovi raznih proizvođača koji omogućavaju sportašima pratiti temperaturu, rad srca, prijeđenu kilometražu ili potrošene kalorije. To se ujedno nadovezuje i na stavku pod brojem dva jer za mjerenje te iste temperature ili neke druge informacije koriste razni senzori koji se ujedno mogu spojiti na računala, spremati na "oblak" i sl. Kada se priča o trećem čimbeniku misli se na činjenicu da su do prije desetak godina samo ljudi komunicirali na mreži, dok se danas ta slika mijenja te na mreži postoje različiti uređaji sa svojim jedinstvenim adresama i informacijama koje razmjenjuju na

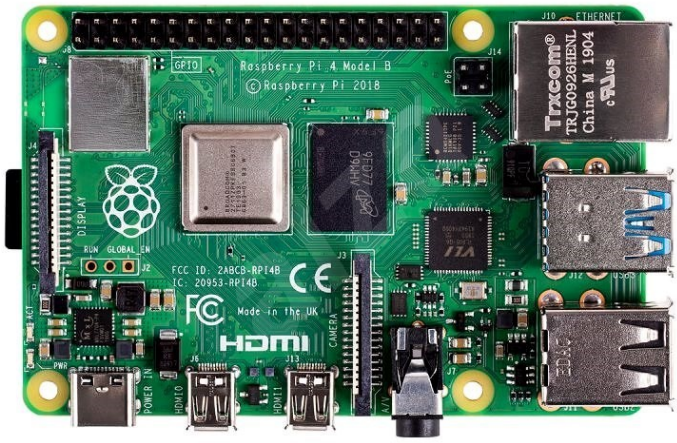
mreži. U budućnosti će biti od ključnog značaja filtriranje važnih informacija od tih milijardi drugih "beskorisnih" informacija koje će putovati mrežom.

3.2. Hardver

Hardver je opipljivi odnosno fizički dio računala. Njega čine sve elektroničke i mehaničke komponente. Tu spadaju komponente poput matične ploče, procesora, grafičke kartice, zvučnika, tipkovnice, miša i sl.

3.2.1. Mala računala

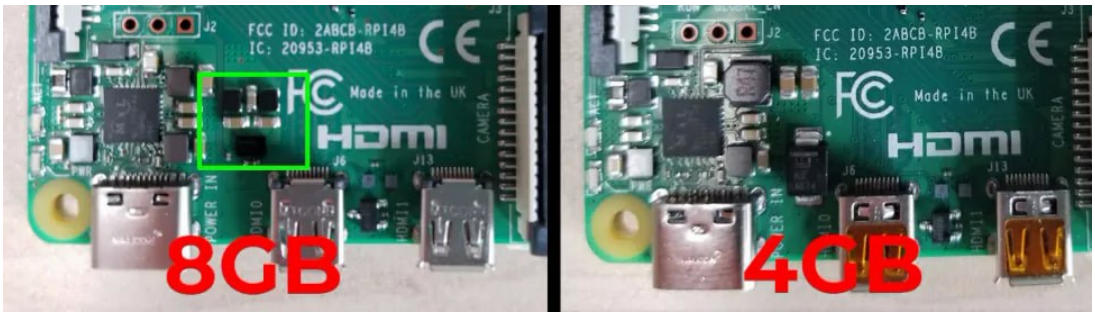
Gore navedeni Raspberry Pi 4 Model B je SBC (engl. *Single Board Computer*) koji je razvijen u svrhe učenja osnova računalnih znanosti. Ključne komponente Raspberry Pi-a su 64-bitni ARM Cortex-A72 četverojezgreni procesor koji radi na 1.5GHz , podupiranje rezolucija do 4K, 8GB RAM-a te 3.0 USB, dok za umrežavanje koristi gigabit Ethernet [7]. Što se periferije tiče može koristiti bilo koji uređaj koji podupire USB, ovisno o podršci i operacijskom sustavu, dok se drugi uređaji mogu na njega spajati preko raznih priključaka i konektora koji se nalaze na Raspberry Pi-u. Za svrhe ovog projekta na njemu je instalirana zadnja inačica Raspberry Pi OS, te se samom Raspberry Pi-u pristupa se preko VNC-a (engl. *Virtual Network Computing*) koji omogućava prikaz radne površine Raspberry Pi-a preko ekrana na računalu pošto se samom Raspberry Pi-u pristupa bez miša, tipkovnice i monitora koji su spojeni na njega (engl. *headless*). Za programiranje dijelova kôda koji se izvršavaju na Raspberry Pi-u koristi se programski jezik Python. Na slici 3.1. prikazan je Raspberry Pi sa svim svojim dijelovima i ulazima.



Slika 3.1. *Raspberry Pi 4 Model B sa svim svojim priključcima i ulazima.*

Raspberry Pi 4 Model B također je izašao u tri varijante ovisno o željenoj količini radne memorije: 1 GB RAM-a, 2 GB RAM-a i 4 GB RAM-a.

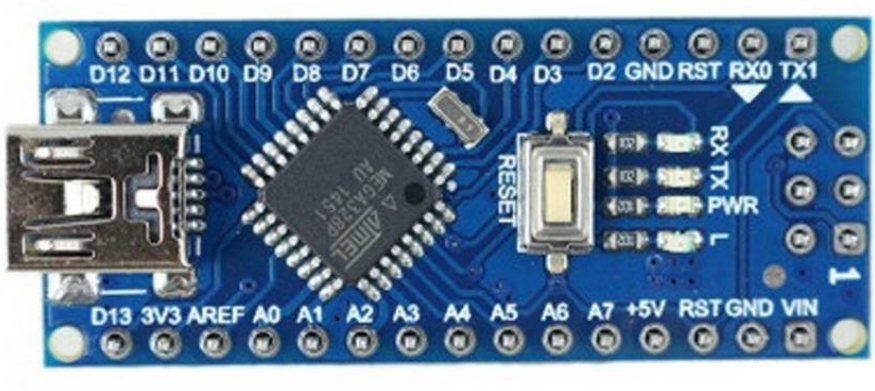
Tijekom 2020. godine izlazi još jedna varijanta istog modela. Različita je od prethodnih modela po količini radne memorije koje ima, a to je 8 GB. S tim povećanjem memorije dolazi i do promjene napajanja i to na napajanje s prekidanjem struje (engl. *switching-mode power supply*). Slikom 3.2. prikazana je razlika između verzije sa 4 GB i 8 GB radne memorije.



Slika 3.2. *Razlika između napajanja na modelima s 4GB i 8GB RAM-a [8].*

3.2.2. Mikrokontroleri

Arduino Nano u kombinaciji sa senzorima DHT222 i NRF24L01 služi kao niskoenergetski odašiljač za prikupljanje informacija o temperaturi i vlazi. Prikupljeni podatci se šalju na Raspberry Pi koji služi kao baza podataka. Sam Arduino Nano je mikrokontroler koji je lako ukomponirati s eksperimentalnom pločicom (engl. *breadboard*) te je baziran na ATmega328 mikrokontroleru s jednim čipom koju je napravila tvrtka Atmel. Arduino Nano i mikrokontroler ATmega328 prikazani su slikom 3.2.Također Nano koristi i modificiranu 8-bitnu jezgru RISC procesora, dimenzija je 45x18mm te koristi napajanje od 5V [9]. Slikom 3.3. prikazan je Arduino Nano sa svim pripadajućim ulazima i izlazima.



Slika 3.3. *Arduino Nano sa svim svojim ulazima i izlazima [9].*

Arduino Nano je korišten kao podređeni mikrokontroler koji prikupljene podatke šalje nadređenom mikrokontroleru Arduinu Uno-u koji je spojen na Raspberry Pi. Osim Arduino Nanoa u radu se koristi još i Arduino Uno koji je prikazan slikom 3.4.

Slika 3.4. *Prikaz Arduino UNO R3 mikrokontrolera s pripadajućim ulazima i izlazima [10].*

Arduino UNO R3 je mikrokontrolerska ploča bazirana na 8bitnom Atmega328P mikrokontroleru. Osim mikrokontrolera sastoji se još od kristalnog oscilatora, serijske komunikacije, regulatora napona i sl. Također na sebi ima i 14 digitalnih input/outputa od kojih se njih 6 može koristiti kao pulsno širinski modulator (engl. *Pulse Width Modulator – PWM*). Osim digitalnih ima i 6 analognih priključaka, USB ulaz, tipku za reset i zaglavlje za serijsko programiranje u krugovima (engl. *In Circuit Serial Programming – ICSP*)[10].

14 digitalnih priključaka može se koristiti preko funkcija kao što su pinMode(), digitalRead(), digitalWrite(). Svaki od tih priključaka radi na 5V te može primiti ili dati struje jakosti do 40mA, također ima interni otpornik od 20-50 KOhma koji je u početku isključen.

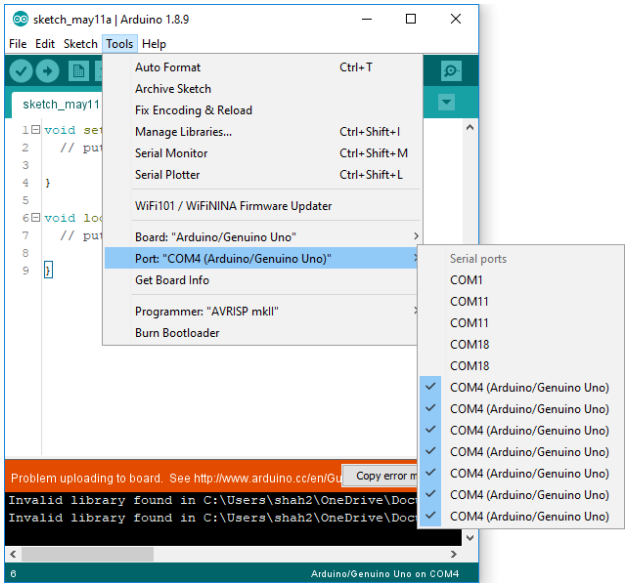
Od tih 14 priključaka neki imaju specifične funkcije kao što su:

- Serijski priključak 0 koji predstavlja prijamnik (engl. *Receiver – Rx*) i 1 koji predstavlja predajnik (engl. *Transceiver – Tx*).
- Vanjski prekidni priključci 2 i 3 koji mogu biti namješteni da izazovu prekid na niskim vrijednostima, promjeni vrijednosti i sl.
- Priključci za pulsno širinsku modulaciju na brojevima 3, 5, 6, 9 i 11.
- SPI priključci 10 (SS), 11 (MOSI), 12 (MISO) i 13 (SCK) koji se koriste za serijsku komunikaciju.
- LED-ica na priključku 13.

Na 6 analognih priključaka svaki prima vrijednost između 0-1024. Mjere između 0 i 5V ali se te granice mogu izmijeniti preko Reference() funkcije.

Osim njih još postoje AREF koji daje reference na voltažu te Reset priključak koji resetira mikrokontroler.

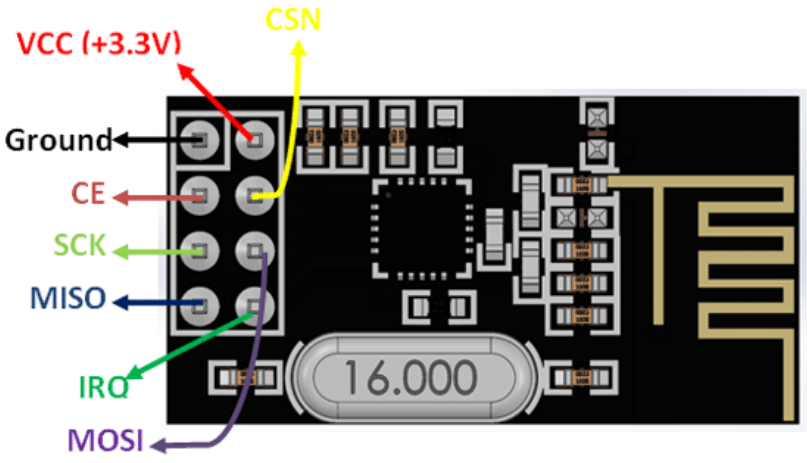
Arduino može komunicirati s računalom, drugim Arduinom kao što je učinjeno ovim projektom ili nekim drugim mikrokontrolerom. Komunikacija se odvija preko digitalnih priključaka 0 (Rx) i 1 (Tx). Kada se Arduino spoji na računalo USB-om stvori se virtualni port preko kojega Atmega16U2 komunicira s računalom preko Arduino IDE-a. Odabir porta i Arduino IDE-a prikazan je slikom 3.5.



Slika 3.5. *Odabir porta preko kojega se odvija komunikacija između računala i Arduina.*

3.2.3. Komunikacija i senzori

nRF24L01 je primopredajnik s jednim čipom koji radi s frekvencijama između 2.4 i 2.5GHz ISM (engl. *Industrial Scientific Medicinal*) frekvencijskog raspona, a prikazan je slikom 3.5. Sam primopredajnik se sastoji od potpuno integriranog sintetizatora frekvencije, pojačala snage, kristalnog oscilatora, demodulatora, modulatora i poboljšanog ShockBurst motora. Izlazna snaga, podešavanje protokola i frekvencijski kanali se lako programiraju putem SPI-a (eng. *Serial Peripheral Interface*). Potrošnja je vrlo mala samo 9mA pri izlaznoj snazi od -6dBm [decibel / miliwatt] i 12.3mA u RX načinu rada [11]. Slikom 3.6. prikazan je primopredajnik sa pripadajućim priključcima.



Slika 3.6. *Prikaz nRF24L01 primopredajnika s pripadajućim priključcima.*

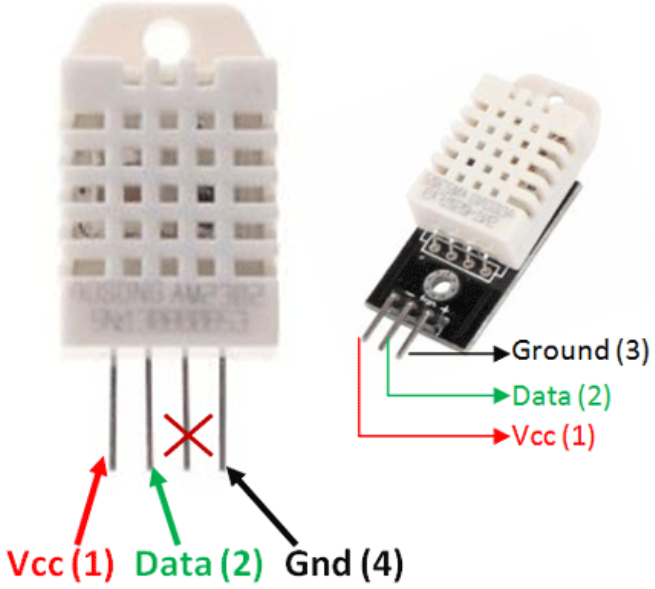
Gore navedeni primopredajnik se sastoji od:

- Uzemljenja (engl. *Ground*).
- Napajanja od 3.3V (*VCC*).
- Ulaza za omogućenje čipa (engl. Chip Enable - *CE*).
- Ulaza za gašenje sinkrone serijske komunikacije(engl. Chip Select Not - *CSN*).
- Serijskog sata (engl. Serial Clock – *SC*).
- Nadređena linija za slanje podataka podređenom uređaju (engl. *Master Out Slave In – MOSI*).
- Podređena linija za slanje podataka nadređenom uređaju(engl. *Master In Slave Out – MISO*).
- Prekida (engl. *Interrupt – IRQ*).

DHT22 je senzor za temperaturu i vlagu malih dimenzija, nasljednik je senzora DHT11. Praktičan je zbog svoje niske potrošnje i velike udaljenosti slanja (20 metara).Senzor DHT22 prikazan je slikom 3.7. Koristi napajanje od 3.5V – 5.5V, te dolazi u dvije varijante kao modul s tri ili kao senzor s četiri priključka. Ako se koristi senzor s četiri priključka jedan priključak se ne koristi tako da se i na tom modelu senzora zapravo koriste samo tri priključka [12].

Može ga se koristiti za :

- Mjerenje temperature i vlage.
- Automatsku kontrolu klime.
- Nadziranje okoliša.
- Kao lokalnu meteorološku stanicu.



Slika 3.7. *Prikaz oba modela senzora DHT22.*

Priključci koje DHT22 koristi su:

- Napajanje (VCC).
- Podatkovni priključak (engl. *Data*).
- Uzemljenje (engl. *Ground*).

Samim time što je nasljednik senzora DHT11 znači da je u određenim pogledima bolji od njega.

Na slici 3.8. prikazana je usporedba specifikacija senzora DHT11 i DHT22.

DHT11	**DHT22**
$5	$9.95
3 to 5V snage i ulazi/izlazi	3 to 5V snage i ulazi/izlazi
2.5mA max.jačina struje tijekom pretvorbe	2.5mA max.jačina struje tijekom pretvorbe
Dobro za očitanja vlažnosti od 20-80% s točnošću od 5%	Dobro za očitanje vlažnosti od 0-100% s točnošću od 2-5%
Dobro za očitanja temperature od 0-50 ºC I točnosti očitanja od +- 2ºC	Dobro za očitanja temperature od -40 to 80ºC I točnosti očitanja od +- 0.5 ºC
Ne više od 1Hz frekvencije uzorkovanja (jednom svake sekunde)	Ne više od 0,5Hz brzina uzorkovanja (jednom u 2 sekunde)
Dimenzije uređaja 15,5 mm x 12 mm x 5,5 mm	Dimenzije uređaja 15,1 mm x 25 mm x 7,7 mm
4 priključka s razmakom od 0,1*	4 priključka s razmakom od 0,1*

Slika 3.8. *Usporedba senzora DHT11 i DHT22.*

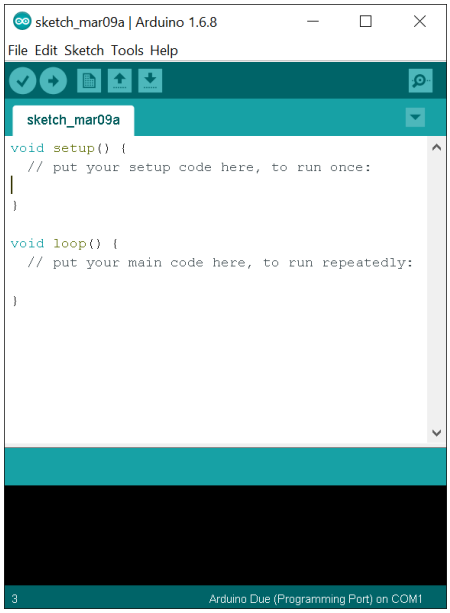
Na slici se također vidi da je DHT22 skuplji od svoga nasljednika, da su isti po potrošnji i naponu, te da su i istih dimenzija, ali isto tako i da DHT22 ima puno veći raspon mjerenja za temperaturu i vlagu s manjim odstupanjima. Odabir senzora dakle ovisi o primjeni, željenoj točnosti te financijama. Za ovaj projekt korišten je senzor DHT22.

3.3. Softver

Softver je neopipljivi dio računala u koji se svrstavaju svi programi i podaci koji se nalaze na računalu uključujući i operacijski sustav. Postoje tri tipa softvera a to su sistemski softver (npr. operacijski sustav, GUI), aplikacijski softver (npr. Internet preglednici, baze podataka) i programski softver (npr. kompajleri, tekstualni editori).

3.3.1. Arduino jezik

Sam Arduino koristi prilagođeni C jezik, a cjelokupnu pretvorbu iz programskog jezika u binarni vrši prevoditelj (engl. *compiler*). U praksi se koristi integrirano razvojno okruženje tj. IDE (engl. *Integrated Development Environment*) koje je dostupno za velik broj različitih operacijskih sustava. Arduino IDE je razvijan u programskim jezicima Java, C i C++, a sam IDE odlikuju karakteristike poput pretraga, zamjena, usklađivanja margina, naglašavanja sintakse programskih jezika, te posjeduje i razne prozore koji prikazuju razne mogućnosti, izbornike, statuse i sl. Postoje i drugi IDE-ovi koji se mogu koristiti za razvijanje programa za Arduino kao što su Atmel Studio ili AVR Studio. Programski jezici koji se najčešće koriste pri radu s Arduinom su C i C++ s obzirom na to da ima jako veliku podršku brojnih korisnika u vidu brojnih biblioteka [13]. Slikom 3.9. prikazan je izgled Arduino IDE-a.



Slika 3.9. *Izgled osnovnog Arduino IDE-a.*

Minimalni Arduino program se sastoji od dva dijela: *void setup()* i *void loop()*. Kod pripremnog (engl. *setup*) dijela programa inicijaliziraju se varijable, pin modovi, počeci korištenja nekih biblioteka i sl.[14]. Ovaj dio se izvršava samo jednom i to prilikom paljenja Arduino pločice. Nakon kreiranja pripremnog dijela, izvršava se petlja (engl. *loop*). Ona izvršava zadane dijelove programa i to u petlji koja se kontinuirano izvršava što omogućava programu razne izmjene na temelju uvjeta [15].

3.3.2. Python

Python je programski jezik visoke razine i opće namjene. Stvorio ga je Guido van Rossum 1990. godine dok je prva javna inačica bila objavljena u veljači 1991. godine. Sam programski jezik ime dobiva po televizijskoj seriji Monty Python što se reflektira po raznim literaturama koje su rađene za sam programski jezik. Python kao jezik je definiran kao objektno orijentirani programski jezik (engl. *Object-oriented programming*). Kod OOP-a za razliku od drugih načina programiranja težište je na akcijama koje se izvršavaju na podatkovnim strukturama. Cilj je projektiranje aplikacija kao skupine objekata koje mogu međusobno izmjenjivati poruke.

Postoje četiri osnovna principa na kojima se temelji OOP:

1. Apstrakcija.
2. Enkapsulacija.
3. Nasljeđivanje.
4. Polimorfizam.

Apstrakcija je proces u kojemu na temelju promatranja pojedinačne vrste iz stvarnog svijeta gradimo model koji ga opisuje – klasu.

Enkapsulacija predstavlja sva stanja i metode unutar objekta, te omogućuje pristup samo javno dostupnim članovima. Implementacija se sakriva pomoću deklaracija privatno(engl. *private*) i zaštićeno (engl. *protected*).

Nasljeđivanjem je omogućeno da se objekt može višestruko koristiti, proširivati i preraditi, time se štedi vrijeme za programiranje i memorija.

Polimorfizam omogućuje definiranje više metoda istog imena, a svaka od njih može primati kao parametre objekte različitih tipova.

Navedenim osnovnim načelima programer može se apstrakcijom modela iz stvarnog svijeta koristiti samo njemu bitne informacije vezane uz model, te isto tako može definirati statička i dinamička svojstva tih objekata i razinu pristupa za svakog člana klase. Osim osnovnih osobina OOP-a koja su ključna za jezik kao što je Python, praktičan je i zbog automatske alokacije memorije. Sličan je jezicima kao što su Ruby, Perl ili Java. Za razliku od ostalih navedenih jezika on je interpreterski jezik što znači da se programi izvršeni na njemu sporije izvršavaju naspram programa izvršenih na kompajlerskim jezicima kao što su C ili C++. Najviše ga se uspoređuje s Javom jer su oboje prevoditeljski jezici. Unatoč manjkavosti u brzini jako je popularan jezik te ima širok spektar primjena kao što su web development (Django i Flask radni okviri), znanstvena i numerička računanja (SciPy, Pandas, IPython), edukaciji i sl. [16].

3.3.3. Baza podataka

Baza podataka korištena pri izradi projekta je MySQL. MySQL je open source sustav za upravljanje bazom podataka. Uz PostgreSQL je najčešći izbor za projekte otvorenog kôda. Optimizirana je kako bi bila brza na štetu nekim funkcionalnostima. Unatoč tome vrlo je stabilna te podupire brojne programske jezike kao što su Java, PHP, Python, Perl i ostali. Baza je relacijskog tipa, te se pokazala kao najbolji način skladištenja i pretrage velikih količina podataka [17].

Temelj je svakog informacijskog sustava ili poslovnog subjekta koji vlastito poslovanje temelji na dostupnosti brzih i kvalitetnih informacija.

Prije bilo kakvog rada s bazom podataka potrebno je dizajnirati odgovarajući izgled baze odnosno napraviti shemu. Ta shema se u daljnjim postupcima prevodi u određen broj tablica koje se koriste za pohranjivanje podataka.

Osnovni element u bazi se naziva entitet. On može predstavljati bilo što: osobu, događaj, objekt ili bilo koji drugi objekt iz stvarnog života o kojem želimo sačuvati informacije.

Drugi važan pojam u bazi podataka je relacija tj. odnos između raznih entiteta koji se na razne načine predstavljaju unutar baze. Prema vrstama relacije možemo ih podijeliti na:

1. Jedan prema jedan.
2. Jedan prema više.
3. Više prema više.

Osim vrstama relacije potrebne su nam dvije dodatne stavke a to su primarni ključ PK i strani ključ FK. Primarni ključ je jedinstveni identifikator koji se u svakoj tablici može ponoviti samo jednom (npr. JMBG, ID), dok se strani ključ može ponavljati više puta i služi za spajanje više tablica.

Jedan prema jedan (1:1) odnos predstavlja odnos u kojoj jednoznačna vrijednost primarnog ključa može biti vezana samo na jedan vanjski ključ (npr. jedan student može imati samo jedan indeks).

Jedan prema više (1:M) je najčešći i predstavlja odnos u kojem jednoznačna vrijednost primarnog ključa može povezivati niti jedan, jedan ili više slogova u tablicama (npr. više studenata može biti iz istog mjesta).

Veza više naprema više (M:M) se rješava dodavanjem treće tablice koja odnos više naprema više rastavlja u odnos jedan naprema više. Primarni ključevi iz svake od te dvije tablice se nalaze u trećoj tablici, a za rezultat se dobiva treća tablica koja zapisuje svaku pojavu ili instancu odnosa.

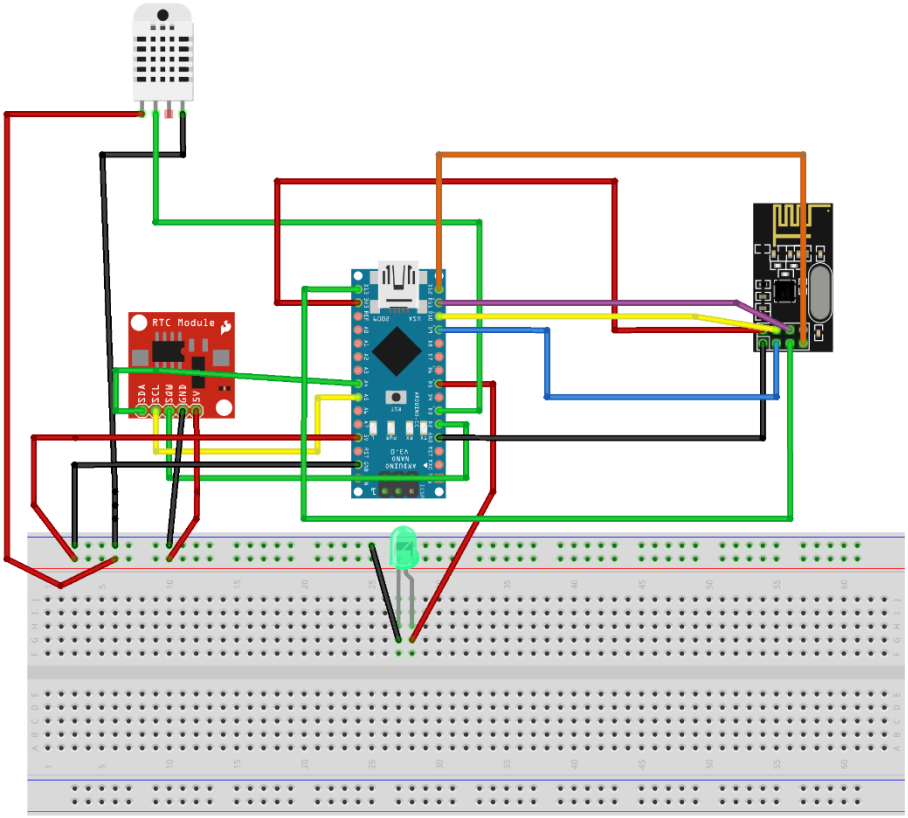
Relacijske baze podataka kao što je MySQL podatke pohranjuju unutar tablica koje se sastoje od stupaca i redova. Stupci se još nazivaju i atributima ili poljima, a služe za skladištenje raznih podataka i informacija o određenim entitetima. Redovi se još nazivaju i slogovi ili zapisi (engl. *record*) te sadrže sve podatke jednog entiteta (u stručnim literaturama još se nazivaju n-torke što je preuzeto iz matematike gdje su baze podataka utemeljene davnih 1970-ih).

4. PROJEKT SENZORSKE STANICE UNUTAR KUĆE

Za ispravan rad cjelokupnog sustava potrebno je spajanje svih komponenti na njihova odgovarajuća mjesta. Arduino Nano kao vanjska stanica je spojen na laptop koji se nalazi u drugoj prostoriji, dok je Arduino Uno kao bazna stanica spojen na Raspberry Pi u drugoj prostoriji. Osim izvora napajanja ključno je da su ispravno spojeni na USB ulaze, također da su senzori spojeni na primjerene GPIO priključke kako na Arduino mikrokontrolerima tako i na Raspberry Pi-u. Arduino stanice preko senzora prikupljaju podatke te ih razmjenjuju, dok Raspberry Pi služi za pohranu i prikaz podataka korisniku na ekranu računala, te na LCD 16x2 zaslonu.

4.1. Arduino Nano (vanjska stanica)

Arduino Nano povezuje se USB kablom na laptop koji mu služi kao izvor napajanja, dok se senzori DHT22, DS3231 RTC i nRF24L01 spajaju na odgovarajuće GPIO priključke na Arduinu. Spajanje je prikazano na slici 4.1.



Slika 4.1. *Shematski prikaz spajanja Arduino Nano-a.*

DHT22 senzor je crvenom žicom spojen preko eksperimentalne pločice na napajanje Arduino Nanoa, dok je crnom žicom spojen na uzemljenje. Isto tako zelenom žicom spojen je podatkovni priključak sa senzora na Arduinov digitalni priključak D3.

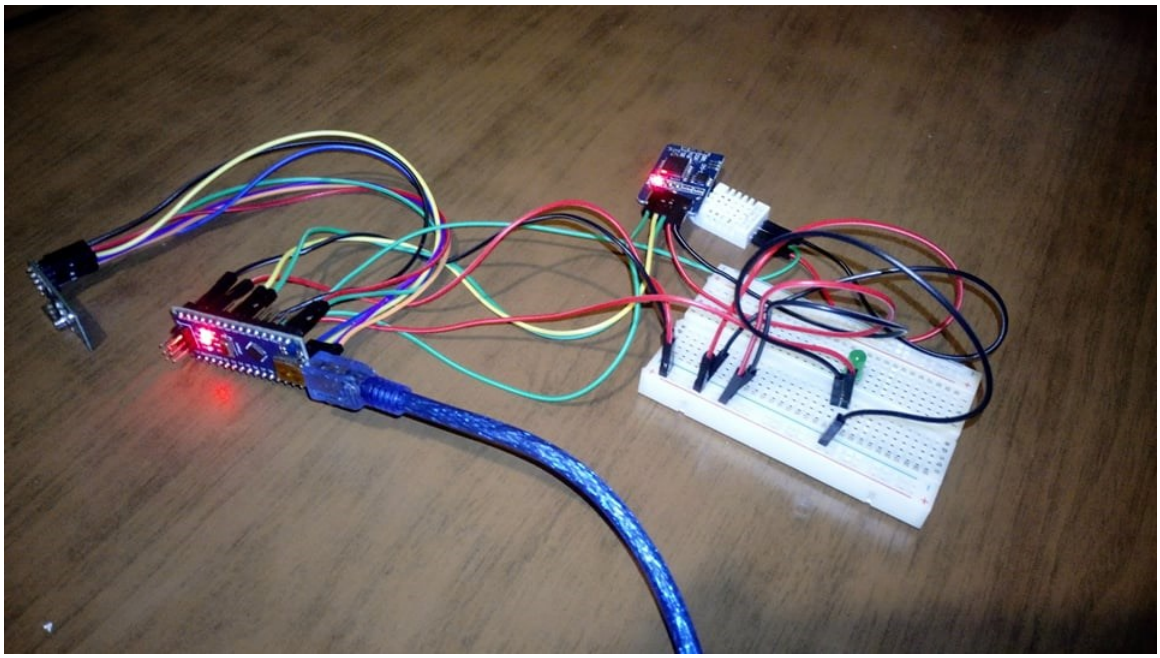
DS3231 RTC modul [18] je također spojen preko eksperimentalne pločice gdje su crvenom i crnom žicom spojeni napajanje i uzemljenje, dok su ostali priključci spojeni na odgovarajuća mjesta:

- Podatkovni priključak (engl. *data line - SDA)* spojen na analogni priključak A4 (zelena žica).
- Vremenski priključak (engl. *clock line - SCL)* spojen na analogni priključak A5 (žuta žica).
- Priključak kvadratnog vala (engl. *Square wave line - SQW)* spojen na digitalni priključak D2 (zelena žica).

nRF24L01 radio frekvencijski modul spojen je s crvenom i crnom žicom izravno na Arduino i to na priključke 3.3V i na uzemljenje. Ostali priključci su spojeni na sljedeći način:

- CSN (engl. *Chip Select Not*) spojen na digitalni priključak D10 (žuta žica).
- MOSI (engl. *Master Out Slave In*) spojen na digitalni priključak D11 (ljubičasta žica).
- CE (engl. *Chip Enable*) spojen na digitalni priključak D9 (plava žica).
- SCK (engl. *Serial Clock*) spojen na digitalni priključak D13 (zelena žica).
- MISO (engl. *Master In Slave Out*) spojen na digitalni priključak D12 (narančasta žica).

Osim njih još je spojena i zelena LED dioda koja služi za prikaz buđenja Arduina iz dubokog sna. Ona je spojena preko eksperimentalne pločice na uzemljenje pomoću crne žice te na digitalni priključak D5. Slikom 4.2. prikazano je kako je spojen vanjski senzor Arduino Nano.



Slika 4.2. *Gotova izvedba Arduino Nano stanice.*

4.1.1. Kôd za Arduino Nano senzorsku stanicu

Nakon što je spojen Arduino Nano i sve ostale komponente kao što je navedeno slikom 4.3. vidi se uključivanje svih ostalih potrebnih biblioteka koje ih podupiru te omogućavaju lakši rad i brže programiranje senzorske stanice.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <DHT.h>
#include <DS3231.h>
#include <DS3232RTC.h>
#include <avr/sleep.h>
```

Slika 4.3. *Učitavanje biblioteka potrebnih za Nano senzorsku stanicu.*

Osim uključivanja biblioteka na početnih nekoliko linija također vidi se i definiranje priključka na koji je spojen, njegov tip „*DHTTYPE DHT22*“ te „*interruptPin*“ koji služi za buđenje Arduina iz dubokog sna. Osim priključaka definirani su još vremenski interval „*time_interval*“ koji predstavlja vremenski razmak u minutama u ovom slučaju jednu minutu, te „*sum_time*“ koji je prazna varijabla u koju se sprema suma vremena.

Osim njih još je definiran i „*rtc(SDA,SCL)*“ koji služi kao funkcija koja kao argumente prima podatak i vrijeme. Varijabla „*pipeOut*“ služi za međusobnu komunikaciju između prijemnika i predajnika. Sve navedene biblioteke i definicije prikazane su slikom 4.4.

```
const uint64_t pipeOut = 0xE8E8F0F0E1LL; //Defining transmition pipe
#define DHTPIN 3
#define DHTTYPE DHT22
#define interruptPin 2 //Pin we are going to use to wake up the Arduino

const int time_interval=1;// Sets the wakeup interval in minutes
int sum_time;
DS3231  rtc(SDA, SCL);
DHT dht(DHTPIN, DHTTYPE);
```

Slika 4.4. *Definiranje priključaka, varijabli i funkcija potrebnih za stanicu.*

Osim gore navedenih definiranih varijabli, definiramo još jednu varijablu „*RF24 radio(9,10)*“ koja kao argumente prima priključke 9 i 10 koji predstavljaju CE i CSN priključke na nRF24L01 senzoru. Zadnja varijabla koja je definirana je struktura „*struct MyData*“ koja se sastoji od dva podatka tipa byte nazvanih „*byte h*“ i „*byte t*“ koji predstavljaju vlagu i temperaturu koja se šalje gore definiranom pipe adresom. U „*void setup()“* djelu su definirane sve funkcije koje se izvršavaju samo jednom i omogućavaju korištenje biblioteka koje su uključene u programski kôd. Na slici 4.5. se vidi početak „*dht*“, „*rtc*“ i „radio“ funkcija, te također postavljanje trenutnog vremena pomoću funkcija „*setDOW()*“, „*setTime()*“ i „*setDate()*“ koje su dio „*rtc*“ biblioteke.

```
void setup()
{
  pinMode(5, OUTPUT); // Pin 5 on Arduino Nano - Green LED

  Serial.begin(9600);
  //rtc.begin(); //Beginning of rtc
  dht.begin();
  rtc.begin();
  radio.begin();
  radio.setAutoAck(false);
  radio.setDataRate(RF24_250KBPS);
  radio.openWritingPipe(pipeOut);

  //Adding rtc functionality

  rtc.setDOW(FRIDAY);     // Set Day-of-Week to SUNDAY
  rtc.setTime(9, 00, 0);     // Set the time to 12:00:00 (24hr format)
  rtc.setDate(24, 6, 2021);   // Set the date to June 24th, 2021
```

Slika 4.5. *„void setup()“ funkcija koja se izvršava samo jednom i to kada se program pokrene.*

Na slici 4.6. redom je prikazano postavljanje dva alarma „*ALARM_1*“ koji je primarni, i „*ALARM_2*“ koji služi kao sekundarni u slučaju nužde. Također se vidi da se pomoću „*squareWave*()“ funkcije ne koristi kvadratni signal nego da je cilj koristiti ga kao prekidač (engl. *interrupt*). Uz pomoć funkcije „*RTC.get*()“ dohvaća se trenutno vrijeme, dok se uz pomoć „*RTC.setAlarm*()“ postavlja vrijeme tako da ono odgovara zadanim minutama i vremenskom intervalu koji je bio definiran na početku programa kao jedna minuta.

```
RTC.setAlarm(ALM1_MATCH_DATE, 0, 0, 1, 0);
RTC.setAlarm(ALM2_MATCH_DATE, 0, 0, 1, 0);
RTC.alarm(ALARM_1);
RTC.alarm(ALARM_2);
RTC.alarmInterrupt(ALARM_1, false);
RTC.alarmInterrupt(ALARM_2, false);
RTC.squareWave(SQWAVE_NONE);
```

```
 time t t; //create a temporary time variable so we can set the time and read the time from the RTC
t=RTC.get();//Gets the current time of the RTC
RTC.setAlarm(ALM1_MATCH_MINUTES , 0, minute(t)+time_interval, 0, 0);// Setting alarm 1 to go off every 5 minutes
// clear the alarm flag
RTC.alarm(ALARM_1);
// configure the INT/SOW pin for "interrupt" operation (disable square wave output)
RTC.squareWave(SQWAVE_NONE);
// enable interrupt output for Alarm 1
RTC.alarmInterrupt(ALARM_1, true);
```

Slika 4.6. *Postavljanje alarma korištenjem „RTC“ biblioteke.*

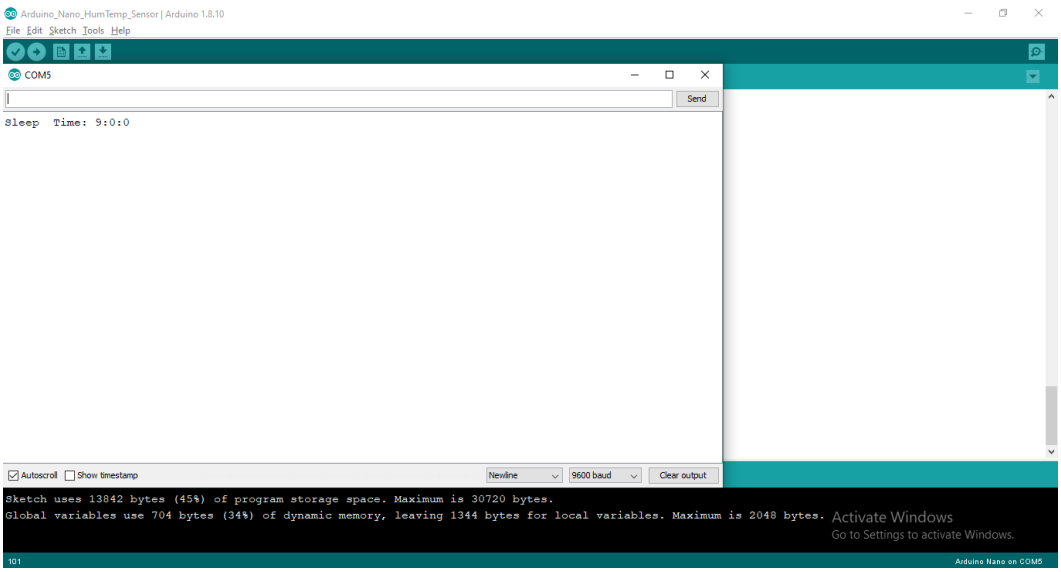
Na slici 4.7. prikazana je „*loop*()“ funkcija koja konstantno poziva funkciju „*Going_To_Sleep*()“ što znači da se kôd izvršava redom liniju po liniju kako je definirana ta funkcija.

```
void loop()
{
  Going_To_Sleep();

}

void Going_To_Sleep(){
    sleep_enable();
    attachInterrupt(0, wakeUp, LOW);
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);
    digitalWrite(5,LOW);
    time_t t;
    t=RTC.get();
    Serial.println("Sleep  Time: "+String(hour(t))+":"+String(minute(t))+":"+String(second(t)));
    delay(1000);
    sleep_cpu();//activating sleep mode
    Serial.println("Woke up!");//next line of code executed after the interrupt
    digitalWrite(5,HIGH);//turning LED on
    data.h = dht.readHumidity();
    data.t = dht.readTemperature();
     if (isnan(data.h) || isnan(data.t)){
      Serial.println(F("Failed to read from DHT sensor!"));
    return;
  }
```

Slika 4.7. *Prikaz „void loop()“ funkcije kojoj je svrha aktivno kontroliranje i upravljanje Arduino mikrokontrolerom u beskonačnoj petlji.*

Kada program završi s posljednjim redom kôda kreće petlju i taj isti kôd otpočetka. U funkciji „*Going_To_Sleep*()“ definirani su omogućavanje spavanja Arduina, paljenje i gašenje LED-ice kada se Arduino probudi ili zaspi, te ispis na ekran vremena kada je Arduino „zaspao“ (Slika 4.8). Osim tih funkcija vrši se i čitanje vrijednosti temperature i vlage, te ispisivanje greške ako ih Arduino nemože dohvatiti.

Slikama 4.8. i 4.9. prikazani su ispis vremena kada je Arduino Nano otišao u duboki san, te prikaz kôda koji služi za ispis temperature i vlage na ekran.



Slika 4.8. *Ispis vremena kada je Arduino Nano otišao u „duboki san“.*

```
//Date and time
 // Send Day-of-Week
Serial.print(rtc.getDOWStr());
Serial.print(" ");

// Send date
Serial.print(rtc.getDateStr());
Serial.print(" -- ");
// Send time
Serial.println(rtc.getTimeStr());

//temp and humidity print
Serial.print("| Room 1 -- Arduino: | \n");
Serial.print(" Temperature:  ");
Serial.print(data.t);
Serial.print(" °C ");
Serial.print("Humidity: ");
Serial.print(data.h);
Serial.print(" % ");
Serial.print("\n");
```

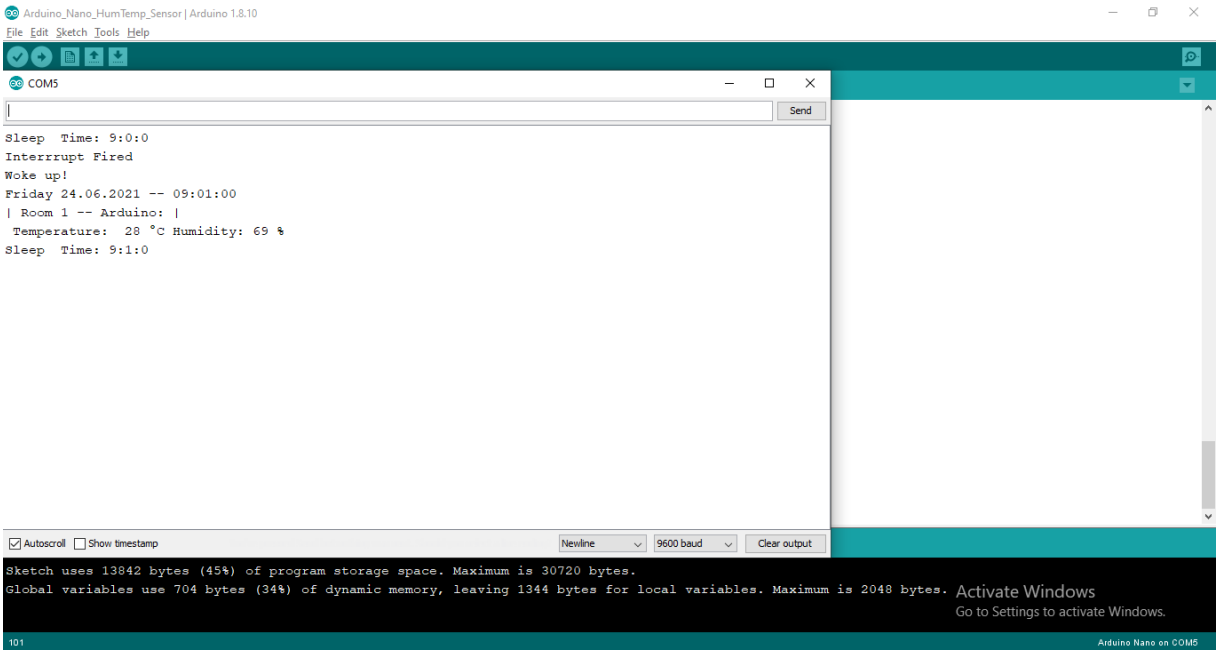
Slika 4.9. *Dohvaćanje i ispis na ekran postavljenog vremena te temperature i vlage.*

Osim gore navedenog ispisa i dohvaćanja podataka te njihovog ispisa istovremeno se također odvija i slanje podataka na Arduino Uno koji se nalazi u drugoj prostoriji, postavljanje novog alarma te se vidi definicija funkcije „*wakeUp*()“ koja služi za okidanje i buđenje Arduina iz dubokog sna (Slika 4.10.). Osim njih još se izvršava i funkcija „radio.*write*()“ koja služi za zapis i prijenos podataka preko „*pipeOut*“ adrese [19].

```
  //Set New Alarm
  sum_time = minute(t)+time_interval;
  if (sum_time >= 59) { //Checking if one minute has passed if it has reset seconds
  sum_time = sum_time - 60;
  }
  else {
  sum_time = sum_time;
  }
  RTC.setAlarm(ALM1_MATCH_MINUTES , 0, sum_time, 0, 0);
  // clear the alarm flag
  RTC.alarm(ALARM_1);
  }

void wakeUp(){
  Serial.println("Interrrupt Fired");//Print message to serial monitor
   sleep_disable();//Disable sleep mode
  detachInterrupt(0); //Removes the interrupt from pin 2;

}
```

Slika 4.10. *Prikaz postavljanja novog alarma i definicija funkcije „wakeUp()“.*

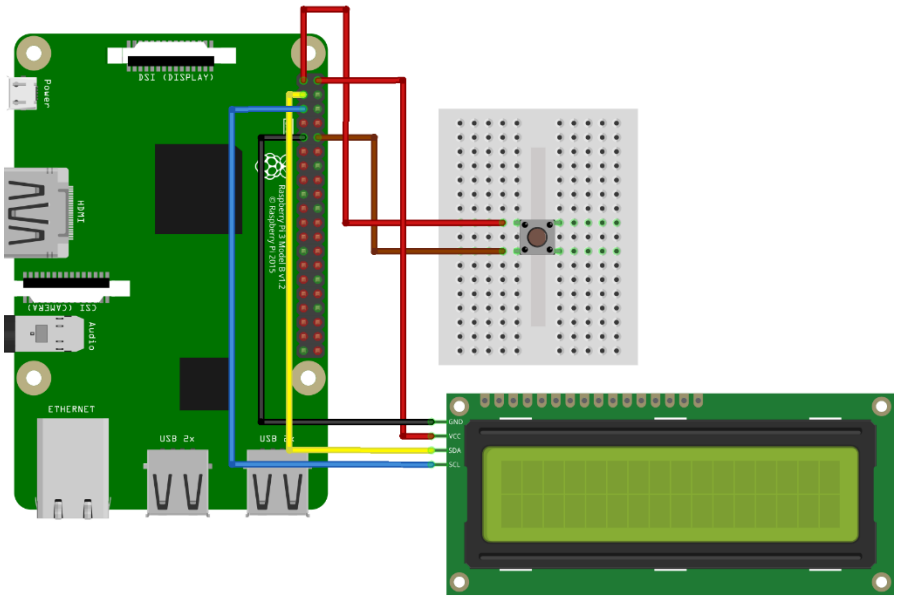
Na kraju svih izvršenih petlji i uvjeta koji su prethodno opisanim programom definirani dobije se ispis svih podataka kao što je navedeno na slici 4.11. Na njoj se vidi vrijeme spavanja Nano-a, kada je prekidač okinuo, njegovo buđenje, ispis podataka te na kraju vrijeme kada se opet vraća u „duboki san“.



Slika 4.11. *Ispis podataka na ekran nakon što se Arduino Nano probudi.*

4.2. Arduino Uno i Raspberry Pi (bazna stanica)

Arduino Uno se spaja na Raspberry Pi koji služi kao bazna stanica [20]. Osim što su spojeni međusobno ključno je da su i ostali senzori pravilno spojeni na pripadajuće priključke prikazane slikom 4.12.



Slika 4.12. *Shematski prikaz spajanja Raspberry Pi-a.*

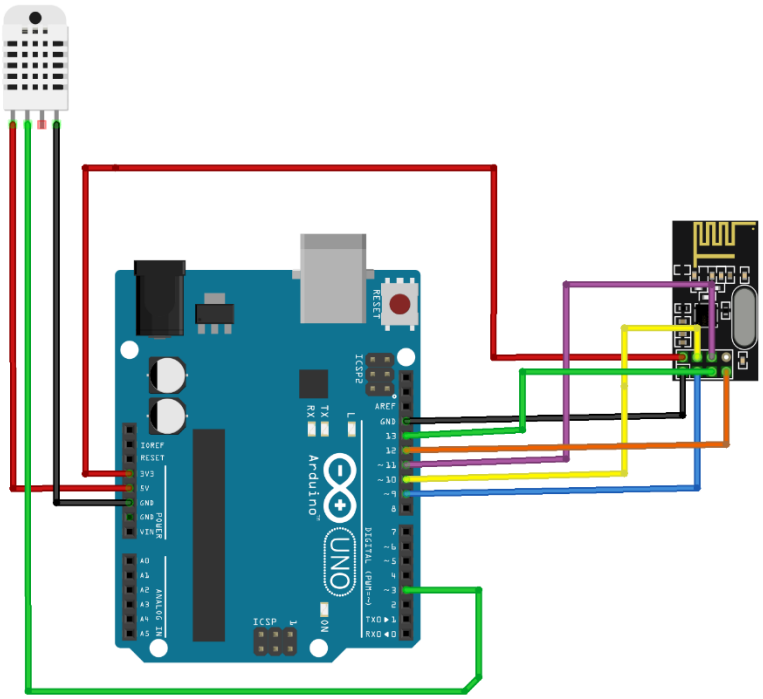
Na Raspberry Pi je spojeno tipkalo (engl. *pushbutton)* i to preko male eksperimentalne pločice. Napajanje je spojeno crvenom žicom na priključak 2 na Raspberry Pi-u dok je smeđom žicom tipkalo spojeno na priključak 10 odnosno GPIO15. Osim tipkala na Raspberry Pi je još spojen i LCD 16x2 ekran. On je spojen na sljedeći način:

- Napajanje spojeno na priključak 1 (crvena žica).
- Uzemljenje spojeno na priključak 9 (crna žica).
- Podatkovni (SDA – data) priključak spojen na priključak 3 (žuta žica).
- Vremenski (SCL - clock) priključak spojen na priključak 5 (plava žica).

Osim međusobnog spajanja na Raspberry Pi se pristupa preko VNC softvera koji omogućava „*headless*“ pristup radnoj površini.

Slično kao i za vanjsku stanicu (Arduino Nano) spojene su komponente i na Arduino Uno. Način spajanja prikazan je slikom 4.13. DHT22 senzor spojen je na sljedeći način:

- Uzemljenje spojeno na GND priključak (crna žica).
- Napajanje spojeno na priključak 5V (crvena žica).
- Podatkovni (engl. *data*) priključak spojen na digitalni priključak 3 (zelena žica).

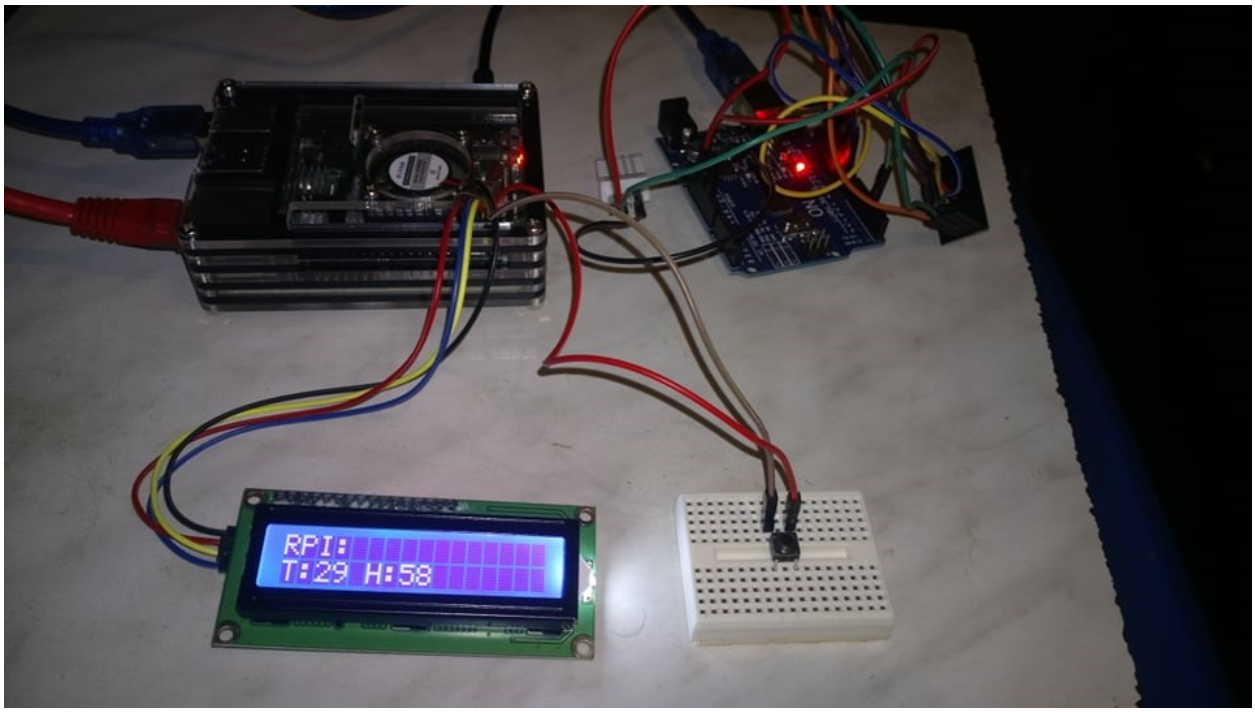


Slika 4.13. *Shematski prikaz spajanja Arduino Uno-a.*

Za razliku od vanjske stanice, na ovu stanicu nije spojen DS3231 RTC modul pošto Uno cijelo vrijeme radi i napaja se preko Raspberry Pi-a. Iako im je taj dio različit zajednički im je još i nRF24L01 senzor koji omogućava njihovu međusobnu komunikaciju koji je spojen na sljedeći način:

- Napajanje spojeno na 3.3V priključak (crvena žica).
- Uzemljenje spojeno na GND priključak (crna žica).
- CSN spojen na digitalni priključak 10 (žuta žica).
- MOSI spojen na digitalni priključak 11 (ljubičasta žica).
- CE spojen na digitalni priključak 9 (plava žica).
- MISO spojen na digitalni priključak 12 (narančasta žica).

Slikom 4.14. prikazan je spoj Arduino Uno-a sa Raspberry Pi-em.



Slika 4.14. *Slika spojenog Arduino Uno-a sa baznim Raspberry Pi-em i odgovarajućim senzorima i modulima.*

4.2.1. Kôd za Arduino Uno baznu stanicu

Nakon što je sve spojeno kao na gore navedenim slikama 4.13. i 4.14. piše se program kao što je bilo i za vanjsku stanicu. Kôd je dosta sličan zbog komponenti koje su zajedničke ili se koriste na oba Arduina.

Na slici 4.15. je prikazano koje su točno biblioteke uključene te definiranje priključaka i funkcija potrebnih za funkcioniranje spoja. Naspram vanjskog senzora ovdje se još koristi i biblioteka „*SPI.h*“ koja omogućava serijsku komunikaciju s Raspberry Pi-em. Također naveden je i „*pipeIn*“ koji se sastoji od identičnih 14 znamenki kao i vanjska stanica. Njegova je svrha primanje podataka. Osim njih definirana je također i struktura „*MyData*“ koja postoji i u vanjskoj stanici, te podaci tipa *byte* „*rpiHumidity*“ i „*rpiTemperature*“ koji služe za pohranu temperature i vlage na toj stanici.

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>

#define DHTPIN 3 // What pin we are using
#define DHTTYPE DHT22 // DHT22
DHT dht(DHTPIN,DHTTYPE); //// Initialize DHT senzor

LiquidCrystal_I2C lcd(0x27, 16, 2);
const uint64_t pipeIn = 0xE8E8F0F0E1LL;
RF24 radio(9, 10);

struct MyData {
  byte h;
  byte t;
};

MyData data;

byte rpiHumidity;
byte rpiTemperature;
```

Slika 4.15. *Uključivanje potrebnih biblioteka i definiranje potrebnih funkcija i varijabli.*

Na slici 4.16. vidi se pozivanje svih funkcija koje su potrebne za početak korištenja gore prethodno definiranih biblioteka, te dodatno definirana funkcija „*void recvData*()“ kojoj je svrha provjeravanja da li je radio komunikacija s vanjskom stanicom moguća te ako je da očita podatke s nje.

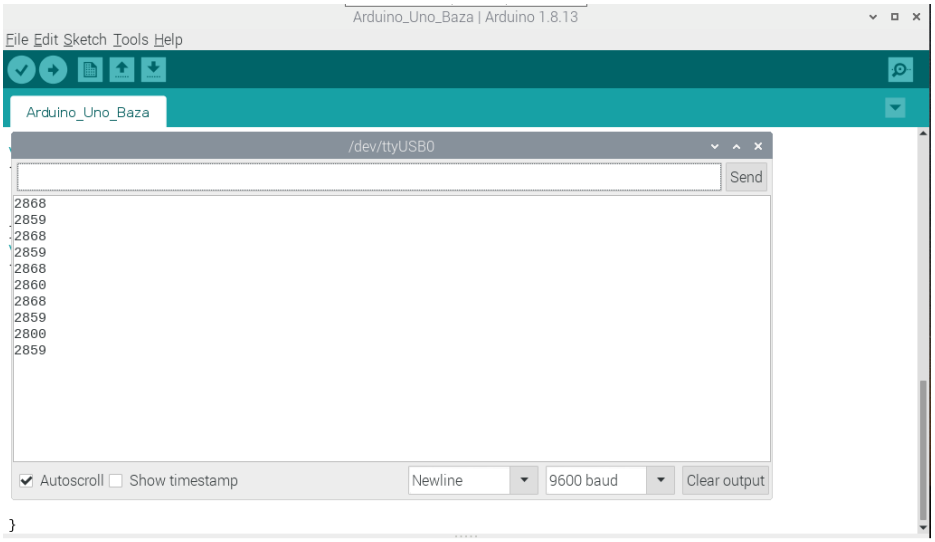
```
void setup()
{
  Serial.begin(9600);
  dht.begin();
  lcd.init();
  radio.begin();
  lcd.home();
  lcd.backlight();
  lcd.clear();
  radio.setAutoAck(false);
  radio.setDataRate(RF24_250KBPS);
  radio.openReadingPipe(1, pipeIn);
  radio.startListening();

}


void recvData()
{
  if ( radio.available() ) {
    radio.read(&data, sizeof(MyData)); //Checking if radio is available and read data
    }
}
```

Slika 4.16. *Prikaz osnovne funkcije „void setup()“ i dodatno definirane funkcije „void recvData()“.*

Slikom 4.17. su prikazane funkcije koje se beskonačno obavljaju liniju po liniju. Na prvoj liniji pokušava očitati podatke sa vanjske stanice, te očitati podatke sa vlastitih senzora. Nakon toga ispisuje te podatke red po red, u prvom redu ispisuje četiri znamenke koje označavaju temperaturu i vlagu dok u idućem redu prikazuje četiri znamenke koje prikazuju temperaturu i vlagu na baznoj stanici. Primjer ispisa podataka vidi se na dolje prikazanoj slici 4.18.

```
void loop()
{
  recvData();
  rpiHumidity = dht.readHumidity();
  rpiTemperature = dht.readTemperature();


  Serial.print(data.t);
  Serial.print(data.h);
  Serial.print("\n");

  Serial.print(rpiTemperature);
  Serial.print(rpiHumidity);
  Serial.print("\n");
  delay(10000);


}
```

Slika 4.17. *Prikaz osnovne „void loop()“ funkcije koja služi kao beskonačna petlja izvršavanja kôda.*



Slika 4.18. *Prikaz ispisa podataka koji su primljeni i koje je bazna stanica očitala.*

4.3. Python skripta

Nakon što se dobije ispis podataka kao što je na slici 4.18. ostatak posla preuzima skripta pisana u Python jeziku. Njena svrha je da oblikuje taj ispis u korisniku prihvatljiv oblik, uz očitane vrijednosti da ispiše i trenutno vrijeme te da te iste podatke ispiše na LCD 16x2 ekran i spremi ih u bazu podataka.

Na slici 4.19. prikazano je uključivanje raznih biblioteka korištenjem ključne riječi „*import*“. Na prvih sedam linija kôda uključuju se sve biblioteke vezane uz MySQL bazu podataka, serijsku komunikaciju, vrijeme i korištenje ulaza i izlaza opće namjene Raspberry Pi-a preko „*RPi.GPIO*“ biblioteke. Na linijama 9 i 10 definirano je da se priključci na Raspberry Pi-u koriste po brojevima na ploči, te da je prekidač koji služi za mijenjanje izbora podataka na izborniku spojen na priključak 10 [21][22]. Na linijama 13 i 14 definirano je korištenje MySQL baze podataka odnosno spajanje i definiranje pozicije kojom se pomiče po bazi podataka.

Na linijama 16 do 21 definirana je funkcija „*pushButtonCallback*()“ koja pritiskom na tipkalo mijenja ispis na LCD-u. Na 21. liniji je prikazano dodavanje detekcije promjene visokog stanja nakon kojeg se izvršava funkcija „*pushButtonCallback*()“. Od linije 22 pa do kraja slike je prikazana „*main*“ funkcija u kojoj je definirana serijska komunikacija preko USB porta, korištenje LCD-a preko funkcije „*display*()“ te očitavanje trenutnog vremena na linijama 27 i 29.

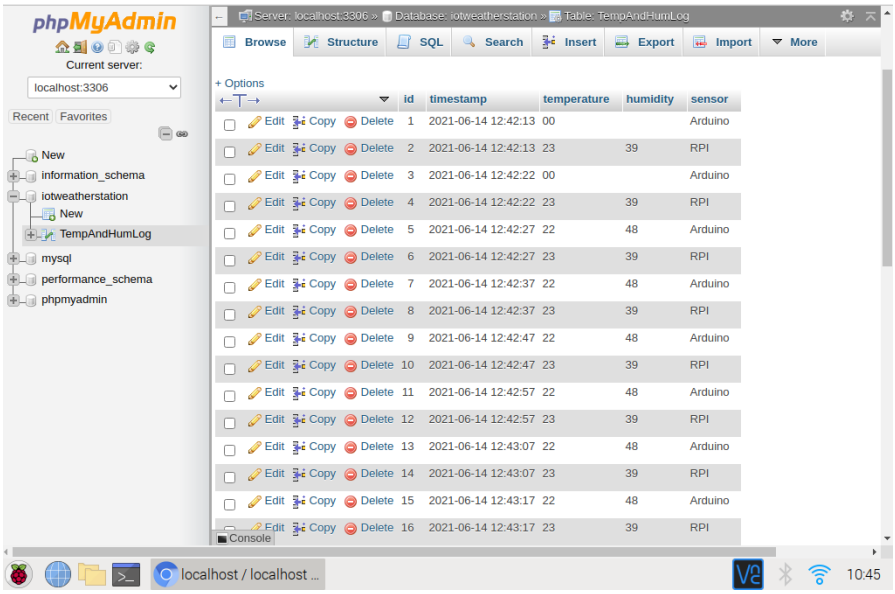
```
1  import mysql.connector  #Import of mysql
2  import drivers
3  import serial
4  from datetime import datetime
5  import time
6  from time import sleep
7  import RPi.GPIO as GPIO
8
9  GPIO.setmode(GPIO.BOARD) #Defining Raspberry Pi pin number mode
10 GPIO.setup(10,GPIO.IN,pull_up_down =GPIO.PUD_DOWN) #and pin 10 which is used for pushbutton
11
12
13 db = mysql.connector.connect(host="localhost",user="root",passwd="",db="iotweatherstation")
14 cursor = db.cursor()
15
16 def pushbuttonCallback(channel): #Callback function for pushbutton
17     display.lcd_display_string('Arduino:', 1)
18     display.lcd_display_string('T:'+ t1 + ' H:' + h1 , 2)
19     sleep(1)
20
21 GPIO.add_event_detect(10,GPIO.RISING,callback=pushbuttonCallback)
22 if __name__ == '__main__':
23     ser = serial.Serial('/dev/ttyUSB0',9600,timeout=10)
24     ser.flush()
25
26     display = drivers.Lcd() # Using lcd display
27     now = datetime.now()
28     # dd/mm/YY H:M:S
29     dt_string = now.strftime("%d/%m/%Y %H:%M:%S")
```

Slika 4.19. *Početak skripte pisane u Pythonu.*

Na slici 4.20 je prikazano prvo oblikovanje vremena te njihov ispis, a zatim i oblikovanje i ispis podataka koji su primljeni serijskom komunikacijom od Arduino Uno-a, njihovo razdvajanje i ispis, te na kraju unos podataka u bazu podataka u zadanom formatu prikazan slikom 4.21.

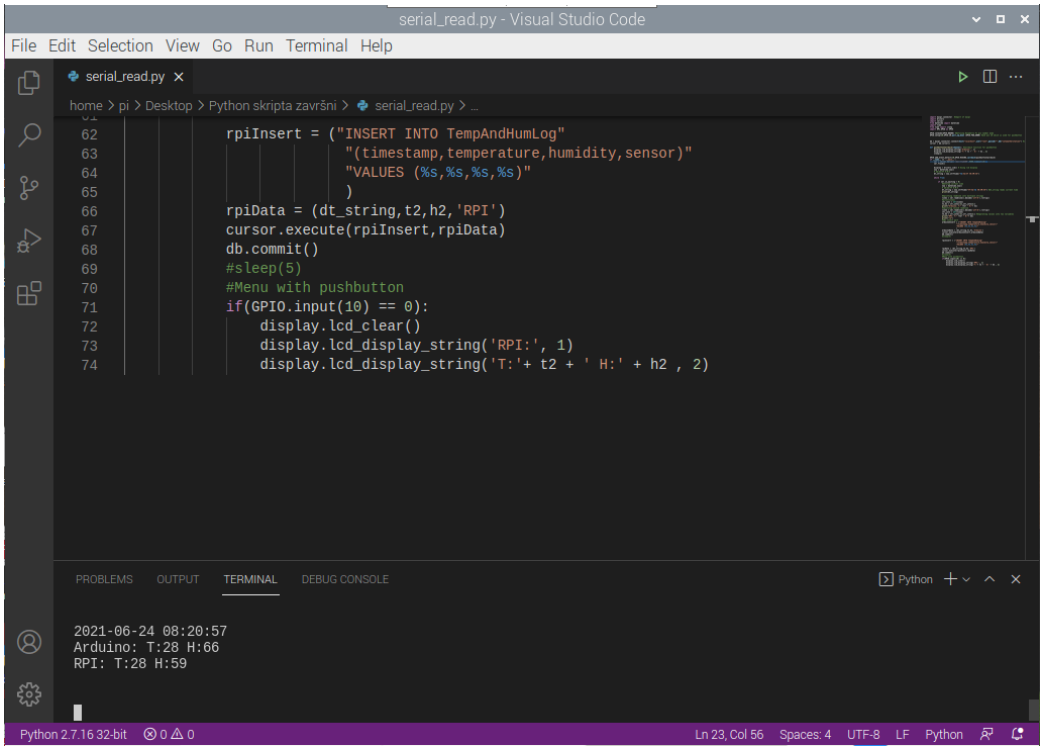
```
31      while True:
32
33          if ser.in_waiting > 0:
34              #Getting current time
35              now = datetime.now()
36              # dd/mm/YY H:M:S
37              dt_string = now.strftime("%Y-%m-%d %H:%M:%S") #dt_string reads current time
38              print(dt_string)
39
40              #Splitting readline into multiple strings
41              line1 = ser.readline().decode('utf-8').rstrip()
42              #print(line1)
43              str_int1 = str(line1)
44              t1,h1 = str_int1[:2],str_int1[2:]
45              print("Arduino: T:" +t1+ " H:"+ h1)
46              #Splitting second readline
47              line2 = ser.readline().decode('utf-8').rstrip()
48              str_int2 = str(line2)
49              t2,h2 = str_int2[:2],str_int2[2:] #Separating values into two variables
50              print("RPI: T:" +t2+ " H:"+ h2)
51              print("\n")
52              #SQL insert part
53              arduinoInsert = ("INSERT INTO TempAndHumLog"
54                               "(timestamp,temperature,humidity,sensor)"
55                               "VALUES (%s,%s,%s,%s)"
56                               )
57              arduinoData = (dt_string,t1,h1,'Arduino')
58              cursor.execute(arduinoInsert,arduinoData)
59              db.commit()
```

Slika 4.20. *Oblikovanje ispisa podataka te njihov unos u bazu podataka.*



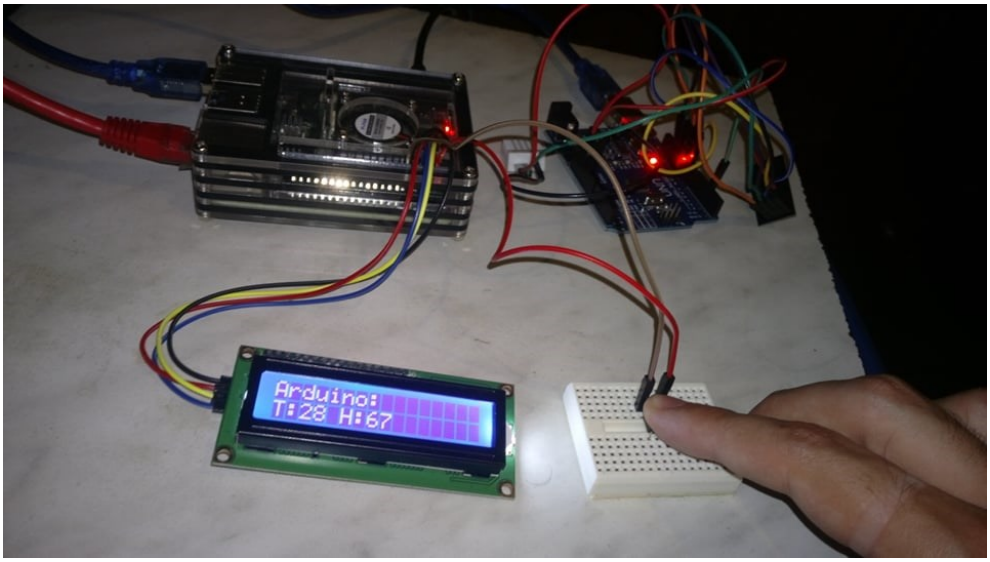
Slika 4.21. *Prikaz unesenih podataka u bazi MySQL.*

Osim što se vidi unos podataka vanjske Arduino stanice na slici 4.20., vidi se i unos podataka sa baznog Arduino Uno-a na slici 4.22.



Slika 4.22. *Unos podataka sa bazne stanice i ispis podataka u terminalu.*

Osim toga na slici se vidi postavljanje ispisa na LCD ekran ako tipka nije pritisnuta, te je na kraju vidljiv formatirani ispis podataka na terminal unutar Visual Studio Code IDE-a. Promjena ispisa podataka na ekranu prikazana je slikom 4.23.



Slika 4.23. *Promjena ispisa podataka na ekranu prilikom pritiska na tipkalo.*

5. ZAKLJUČAK

Cilj ovog završnog rada bio je razviti IoT stanicu za praćenje temperature i vlage u kućanstvu pomoću nekoliko senzorskih stanica koje su u ovom slučaju bile Arduino Nano i Arduino Uno, te Raspberry Pi koji je služio za obradu i pohranu tih podataka u *MySQL* bazu podataka. Kôd je podijeljen na tri dijela, dva koja su pisana u *C++*u te jedan koji je pisan u *Pythonu*. Arduino mikrokontroleri prikupljaju podatke pomoću DHT22 senzora za temperaturu i vlagu te ih međusobno razmjenjuju pomoću nRF24L01 bežičnog modula. Prednost ovog načina rada je ta što se može obuhvatiti čitavo kućanstvo te za malu količinu novca pratiti temperatura i vlaga s prilično velikom preciznošću.

Oba dijela Arduino kôda mogu i ne moraju vršiti ispis podataka na ekran, dok je *Python* skripta napisana tako da preuzima podatke s oba Arduina te ih oblikuje u korisniku ljepši i razumljiviji ispis, te također pruža uvid u podatke preko LCD-a 16x2 dok istovremeno te podatke sprema u bazu. Prednost ovog dijela je što samom izmjenom te skripte mogu se prilagoditi ispisi i unosi u bazu te ujedno i ispis na LCD. Mana je u tome što Raspberry Pi mora cijelo vrijeme biti upaljen da bi ta skripta radila, što je nezgodno zbog zagrijavanja samog Raspberry Pi-a, te nestanka struje. Problem zagrijavanja je u ovom slučaju riješen ugradnjom hladnjaka i ventilatora na njega. Također, potencijalni problem je pristup Raspberry Pi radnoj površini koja je ovdje riješena preko softvera VNC koji služi za mrežni pristup te se njime može riješiti problem pristupa bezglavim (engl. *Headless*) računalima kao što je u ovom slučaju Raspberry Pi. Osim što korisnik ima uvid u bazu podataka preko LCD-a ima i izbornik kojim upravlja pomoću tipke te sam bira ispis koji želi vidjeti na ekranu.

Ovakav sustav može biti koristan ljudima koji žele imati širu sliku o temperaturi i vlazi u svome kućanstvu, nebitno da li zbog toga da spriječe nastanak pljesni ili da si olakšaju boravak u prostorijama znajući kada prozračiti koju prostoriju. Isto tako koristan bi mogao biti za raznorazne staklenike u kojima se uzgaja povrtna ili voćna kultura radi održavanja optimalnih uvjeta za rast i razvoj.

LITERATURA

[1] https://repozitorij.fer.unizg.hr/islandora/object/fer:5994 [25.3.2021.]

[2] https://repozitorij.fer.unizg.hr/islandora/object/fer:6168 [25.3.2021.]

[3] https://repozitorij.oss.unist.hr/islandora/object/ossst:775 [25.3.2021.]

[4] https://repozitorij.etfos.hr/islandora/object/etfos%3A1651 [25.3.2021.]

[5] https://components101.com/sensors/dht22-pinout-specs-datasheet [25.3.2021.]

[6] https://www.ofir.hr/iot-ili-internet-stvari-2/ [17.6.2020.]

[7] https://static.raspberrypi.org/files/product-briefs/200521+Raspberry+Pi+4+Product+Brief.pdf [17.6.2020.]

[8] https://www.tomshardware.com/news/raspberry-pi-4-8gb-tested [25.3.2021.]

[9] https://store.arduino.cc/arduino-nano/ [17.6.2020.]

[10] https://components101.com/microcontrollers/arduino-uno [25.3.2021.]

[11] https://components101.com/wireless/nrf24l01-pinout-features-datasheet [25.3.2021.]

[12] https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf [25.3.2021.]

[13] https://www.arduino.cc/en/guide/introduction/ [4.7.2020.]

[14] https://www.arduino.cc/reference/en/language/structure/sketch/loop/ [4.7.2020.]

[15] https://www.arduino.cc/reference/en/language/structure/sketch/setup/ [4.7.2020.]

[16] https://www.python.org/about/apps/ [4.7.2020.]

[17] https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html [4.7.2020.]

[18] https://lastminuteengineers.com/ds3231-rtc-arduino-tutorial/ [25.3.2021.]

[19] https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/ [22.6.2021.]

[20] https://www.instructables.com/Sending-Temperature-Humidity-Data-From-Arduino-to-/ [22.6.2021.]

[21] http://raspi.tv/2014/rpi-gpio-update-and-detecting-both-rising-and-falling-edges [22.6.2021]

[22] https://www.ardumotive.com/how-to-use-push-buttonen.html [22.6.2021.]

[23] https://raspberrypihq.com/use-a-push-button-with-raspberry-pi-gpio/ [22.6.2021.]

[24] https://www.smart-industry.net/interview-with-iot-inventor-kevin-ashton-iot-is-driven-by-the-users/ [17.6.2020.]

[25] https://pcchip.hr/internet/internet-things-iot/ [17.6.2020.]

[26] https://www.python.org/about/ [4.7.2020.]

[27] https://lastminuteengineers.com/i2c-lcd-arduino-tutorial/ [22.6.2021.]

[28] https://roboticsbackend.com/raspberry-pi-arduino-serial-communication/ [22.6.2021.]

[29] https://diyi0t.com/arduino-to-raspberry-pi-serial-communication/ [22.6.2021.]

[30] https://dev.mysql.com/doc/connector-python/en/connector-python-example-cursor-transaction.html [22.6.2021.]

SAŽETAK

Naslov: IoT stanica za praćenja temperature i vlage unutar kuće

U sklopu završnog rada izrađena je IoT stanica za praćenje temperature i vlage unutar kuće koja rješava problem praćenja temperature i vlage u više prostorija u kućanstvu. Programski kôdovi za *Arduino* senzorske stanice pisani su u *Arduino IDE*-u kôdom koji je u stvari prilagođeni *C++* s dodatnim metodama i funkcijama, te *Python* programskim jezikom koji služi za serijsku komunikaciju između stanica i baze koja je predstavljena Raspberry Pi-em. Vanjska stanica Arduino Nano na sebi ima spojene senzore DHT22 za temperaturu i vlagu, nRF24L01 radiofrekvencijski senzor pomoću kojega šalje podatke na bazni Arduino Uno, te DS231 RTC pomoću kojega se budi i vraća u duboki san nakon prikupljanja i slanja podataka. Bazni Arduino Uno prima podatke preko nRF24L01 prijamnika, te pomoću svoga DHT22 senzora prikuplja podatke u prostoriji u kojoj se nalazi te ih zajedno predaje *Python* skripti koja vrši serijsko iščitavanje podataka. Osim iščitavanja ta skripta uljepšava ispis podataka na ekran i na LCD 16x2 koji je spojen na Raspberry Pi te vrši unos podataka u *MySQL* bazu podataka. Isto tako preko tipkala koje je spojeno na Raspberry Pi prebacuje se ispis koji se događa na LCD-u. Dok tipka nije stisnuta na ekranu se ispisuje temperatura i vlaga prostorije u kojoj se nalazi Raspberry Pi i Arduino Uno, no nakon pritiska na tipku ispis se mijenja u temperaturu i vlagu prostorije u kojoj se nalazi Arduino Nano.

Ključne riječi: Arduino, baza podataka, IoT, Raspberry Pi, temperatura, vlaga

ABSTRACT

Title: IoT station for monitoring temperature and humidity inside the house

As part of the final thesis, an IoT station for monitoring temperature and humidity inside the house was created which solves the problem of monitoring temperature and humidity in several rooms in the household. Program codes for Arduino sensor stations are written in the Arduino IDE using code that is actually customized C ++ with additional methods and functions, and a Python programming language that serves for serial communication between stations and the base represented with the Raspberry Pi. The Arduino Nano outdoor station has connected DHT22 sensor for temperature and humidity, an nRF24L01 radio frequency sensor which sends data to the Arduino Uno base, and a DS231 RTC which wakes it up and returns it to deep sleep after collecting and sending data. The base Arduino Uno receives data via the nRF24L01 receiver, and with the help of its DHT22 sensor collects data in the room in which it is located and passes it to a Python script that performs serial data reading. In addition to reading, this script enhances the output of data on the screen and on the 16x2 LCD connected to the Raspberry Pi and inserts that data into the MySQL database. Also via the button connected to the Raspberry Pi we can switch the printing that happens on the LCD. While the button is not pressed, the temperature and humidity of the room where the Raspberry Pi and Arduino Uno are located are displayed on the screen, but after pressing the button the print changes to the temperature and humidity of the room where the Arduino Nano is located.

Keywords: Arduino, database, humidity, IoT, Raspberry Pi, temperature

ŽIVOTOPIS

Vedran Brodar, rođen je 14. srpnja 1993. godine u Osijeku gdje je odrastao i dan danas živi. Školovanje započinje 2000. godine u Osnovnoj školi Vladimira Becića u Osijeku, dok je srednjoškolsko obrazovanje stekao u Graditeljsko-geodetskoj školi u Osijeku kao građevinski tehničar. Nakon srednje škole upisuje Fakultet elektrotehnke, računarstva i informacijskih tehnologija u Osijeku 2017. godine stručni studij informatike. Na 2.godini započinje rad kao demonstrator na FERIT-u držeći radionice Arduina i robotike u sklopu Slavonske STEM evolucije, te nakon toga nastavlja surađivati sa Zajednicom tehničke kulture Osječko-baranjske županije kroz razne projekte kao što je Tehnicoolum, te i dan danas surađuje s njima. Praksu je odradio 2020. godine kao front-end developer u Atos-u gdje se detaljnije proširuje svoje znanje web developmenta.

PRILOZI

Projektna mapa s izvornim kôdom te priloženim word i pdf dokumentima nalazi se na optičkom disku koji je priložen uz ispisanu verziju završnog rada.