RAZVOJ IoT APLIKACIJA U NODE-RED ALATU

Seferović, Damir

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Preddiplomski stručni studij Elektronike

DAMIR SEFEROVIĆ

ZAVRŠNI RAD

RAZVOJ IoT APLIKACIJA U NODE-RED ALATU

Split, rujan 2020.

SADRŽAJ

1. SAŽETAK

Razvoj IoT aplikacija u Node-RED alatu

Kroz ovaj rad ćemo pobliže razmotriti nastanak IoT-a, njegove primjene i značaja u današnje doba, te mogućnosti programiranja jedne IoT aplikacije. Usredotočit ćemo se na vizualno programiranje kao jednu od metoda programiranja za IoT, te vidjeti kako je ono nastalo, na kojim programerskim tehnologijama te kako je preraslo u sve poznatiji koncept i rješenje za potrebe programiranja. Upoznat ćemo se s najpoznatijim alatima za vizualno programiranje, a posebnu pažnju posvetiti alatu Node-RED kao jednom od trenutno najpopularnijih ovakvih alata. Za demonstraciju napravit ćemo jednu jednostavniju aplikaciju koja će u interakciji s Arduino platformom čitati i prikazivati rezultate na našem kreiranom dashboard-u u Node-RED alatu. Za kraj, stavit ćemo sve u perspektivu te pogledati u kojem smjeru se kreće IoT industrija te kakve benefite donosi korisnicima i pametnim uređajima koji su danas na svakom koraku.

KLJUČNE RIJEČI: Internet of things (IoT), vizualno programiranje (VPL), programiranje na osnovu tijeka (FBP), Arduino, Node-RED

SUMMARY

IoT applications development with Node-RED

In this final paper, we will take a closer look at the origins of IoT, its application and importance in this day of age, as well as the possibilities when it comes to programming an IoT application. We will focus on visual programming as a concept and method for IoT programming and development, see how it originated in consideration to programming technology and how it became a more and more recognized method and solution for programming purposes. We will get acquainted with the most famous visual programming tools, paying special attention to the Node-RED tool as one of the currently most popular tools of this kind. For the demonstration, we will create one simple application that will interact with the Arduino platform to read and display the results on created dashboard in the Node-RED. Finally, we will put everything in perspective and look at the direction in which the IoT industry is being created and what benefit sit brings to users and smart devices that are surrounding us at every corner.

KEY WORDS: Internet of things (IoT), Visual Programming Language (VPL), Flow-Based programming (FBP), Arduino, Node-RED

2. UVOD

Budući da u današnje vrijeme, gotovo da nema osobe ili domaćinstva koje posjeduje minimalno jedan ili dva pametna uređaja, „Internet of Things“ postaje sve značajniji u njihovom povezivanju i iskorištavanju punog potencijala. Do nedavno su se u ovakve zadatke upuštali manje-više isključivo poznavatelji elektronike i relativno vješti programeri, danas je uz razne alate i olakšice ovo područje dosta bliže i ljudima koji nisu nužno iskusni u programiranju. Jedan od takvih alata je zasigurno i Node-Red koji spada u grupu alata zvanih „Visual Programming language“ (VPL). Kao što sam naziv kaže, ovi alati omogućuju korisniku programiranje putem povezivanja vizualnih programskih blokova.

U ovom radu upoznat ćemo se s „Internet of Things“ (IoT) konceptom te razmotriti njegov značaj i doprinos u svijetu današnje tehnologije. Također, promotrit ćemo okruženje u kojem se razvija, upoznat se s nekim metodama programiranja za IoT kao i alatima koji nam u tome pomažu, te ćemo razmotrit jedan jednostavniji primjer aplikacije kreirane u Node-RED alatu, opisati ključne značajke istog te glavne razlike naspram drugih VPL alata.

3. VIZAUALNO PROGRAMIRANJE

3.1. Općenito o vizualnom programiranju

Vizualno programiranje je alternativa klasičnom (tradicionalnom) programiranju, gdje korisnik opisuje i kreira procese koristeći ilustrirane programske blokove. Kod klasičnog programiranja, korisnik je prisiljen razmišljati kao računalo, dok vizualnog programiranje dozvoljava da korisnik opiše proces logikom koja je prirodnija i bliža ljudima. Zasnovano je upravo na programiranju na osnovu tijeka (FBP), pa se često za njih same kaže da su FBP, iako će pioniri originalnog FBP-a napraviti razliku i nazvati ih alatima inspiriranim FBP-om.

Iako vizualni programski jezici postoje još od 70-ih, pretjeranu popularnost još uvijek nisu doživjeli. Jedan od glavnih razloga je što prilikom stvaranja kompleksnijih i velikih projekata, njihov izgled i struktura postaju pretrpani i zamršeni teško čitljivom i nepreglednom mrežom blokova. Ovdje se možda nameće pitanje, ako su VPL tako pogodni za programiranje čak i korisnicima slabijih programerskih vještina, kako da uz sav napredak tehnologije i računala, i oni nisu dostigli takvu popularnost, a to je upravo radi svoj nedostataka:

- VPL nisu skalabilni – većina VPL alata je ograničena određenim setom mogućnosti. Programeri nisu skloni već u startu staviti se u neki okvir mogućnosti s kojim možda neće biti moguće napraviti sve zadatke koje program ili aplikacija trebaju izvršavati.
- VPL su obično primjetno sporiji – Iako se program sastoji od „vizualnih blokova“ alat i dalje od tih blokova u pozadini mora generirati računalni kod koji je najčešće neoptimalan, zagušujući računala kod većih zadataka.
- VPL alati često neće dozvoliti lako prenošenje programa u drugi alat ili okruženje, što korisnika čini dugoročno vezanim za alat. Kod tradicionalnog tekstualnog programiranja, transformacija između jezika je obično višestruko lakša za implementirati

Ovi nedostatci naravno ne znače da VPL nema svoje mjesto u svijetu programiranja, te da u budućnosti neće postajati popularniji i učinkovitiji u tome što pružaju.

Općeniti cilj vizualnih programskih jezika je da programiranje učini pristupačnijim početnicima te da programerima pružaju podršku u tri različita dijela[1]:

1. Sintaksa – VPL koriste ikone ili blokove, forme i dijagrame kako bi smanjili ili potpuno eliminirali mogućnost sintaktičnih pogrešaka, pomažući kod programiranja programskih primitiva kako bi stvorili kvalitetno formiran program.
2. Semantika – VPL-ovi mogu nuditi neke mehanizme koji će otvoriti značenje programskih primitiva. Ovo može uključivati pomoćne funkcije koje će dati dokumentaciju za ugrađene funkcije samog programskog jezika.
3. Pragmatika – VPL-ovi podržavaju ispitivanje programa u određenim situacijama. Ovo omogućava korisniku da stavi program ili njegove dijelove u različite moguće situacije njihove uporabe te promatra ponašanje programa u tim situacijama.

[1] Repenning, Alexander (2017). "Moving Beyond Syntax: Lessons from 20 Years of Blocks Programing in AgentSheets". *Journal of Visual Languages and Sentient Systems*

Fokusirat ćemo se prvenstveno na vizualne programske jezike koji rade na principu protoka podataka (eng. data flow). Ovaj tip jezika podrazumijeva da određeni blok predstavlja proces koji će se obaviti kada blok dobije potrebnu informaciju na ulaz, a zatim obrađenu informaciju proslijediti preko svog izlaza novom bloku/elementu.

3.2. Programiranje na osnovu tijeka – Flow-based programming

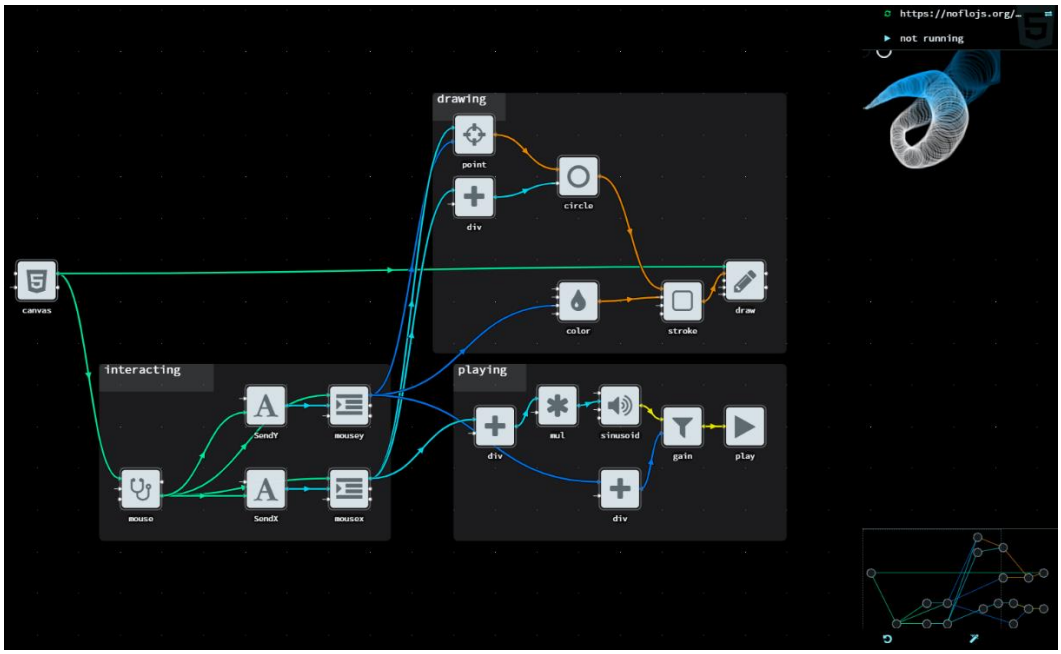
Programiranje na osnovu tijeka („Flow-based programming“, FBP) je paradigma u programiranju koja definira aplikacije kao mreže procesa „crnih kutija“ koje razmjenjuju podatke kroz „informacijske pakete“ preko predefiniranih veza i „propuštanja poruka“ („message passing“), a koje su specificirane eksterno od samih procesa. Ove „crne kutije“ mogu biti višestruko povezivane bez praktičnog ograničenja, tvoreći različite aplikacije bez da ih se mora mijenjati njih same. To čini FBP orijentiranim na komponente.

FBP zasnovao je J. Paul Morrison, krajem 1960-tih[2] i prvobitno ga implementirao u software Kanadske banke. Ovaj način programiranja objašnjava analogijom pokretne trake u proizvodnji, gdje su eksterne predefinirane veze između informacijskih paketa zapravo sama traka. FBP karakteristično je po asinkronim trenutnim procesima u pozadini, „Informacijskim paketima“ sa definiranim trajanjem, imenovanim portovima i definiranim vezama van ovih komponenti, a van ovog koncepta, bitno je spomenuti i da FBP po prirodi koristi sve jezgre procesora računala.

Alate za programiranje slaganjem grafičkih blokova kao NoFlo, Flowhub ili Node-RED, J. Paul M. naziva „FBP-inspirirani alati“, budući da im nedostaju neke od ključnih komponenti originalnog FBP koncepta, iako priznaje da su zaslužni za njegovu popularizaciju.

Henri Bergius je 2012 sa svojim timom krenuo raditi na implementiranju FBP u JavaScript, i njihov projekt „NoFlo“[3] počeo je prikupljati pažnju širom svijeta, popularizirajući FBP paradigmu, što je uz nagli rast pametnih uređaja od 2000. do danas, približilo IoT i vizualno orijentirano programiranje mnogima i koji nisu nužno u uskoj struci programera.

[2] https://jpaulm.github.io/fbp/index.html; J. Paul Morrison “Flow-Based Programming: A New Approach to Application Development”
[3] noflojs.org



Slika 1 - Primjer NoFlo programa (noflojs.org)

Pored NoFlo, neki od poznatijih alta su Node-RED, ThingsBoard, ReactiveBlocks, Visuino, Wia i drugi. Vizualno programiranje se često koristi u proizvodnji, jednostavnijim web aplikacijama i IoT-u.

4. INTERNET OF THINGS

4.1. IoT općenito

Koncept „IoT“ odnosi se na povezanost različitih fizičkih objekata putem interneta i prvi put je spomenuo britanski poduzetnik Kevin Ashton, iz Procter & Gamblea, kasnije zaposlenik MIT-ovog Auto-ID Centra, 1999. godine . Navodeći njegove riječi, računala i Internet su do 2000. godine ovisili manje-više isključivo o ručnom kreiranju podataka od strane ljudi, bilo tipkanjem ili snimanjem različitih sadržaja. Problem s tim je što ljudi imaju ograničeno vrijeme, točnost i pažnju, što znači da nisu dobar i pouzdan način unošenja podataka iz stvarno svijeta. IoT omogućuje interakciju ljudi s uređajima i uređaja s uređajima, integrirajući ih u mrežu kojom se upravlja putem web-aplikacija.

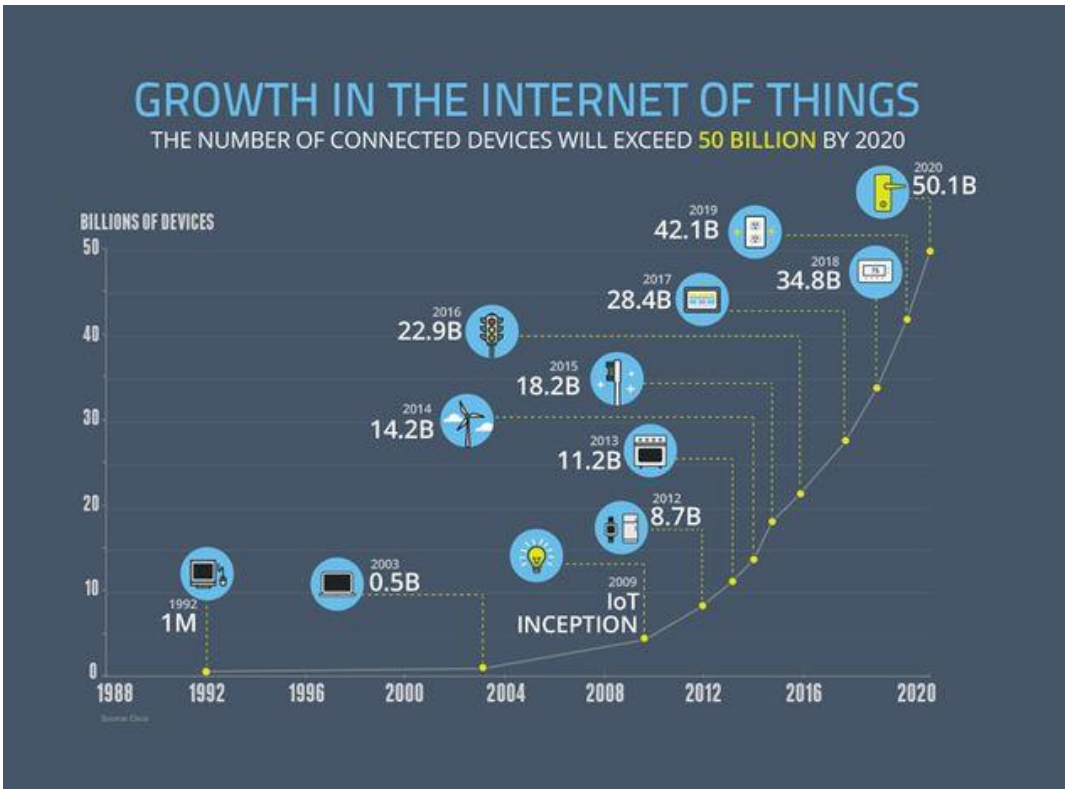
IoT je veoma brzo evoluirao do gotovo neprimjetnog dijela našeg života, iako još uvijek nisu potpuno istražene sve njegove mogućnosti. Danas pametni uređaji, koji su svuda oko nas, konstantno prikupljaju i prenose velike količine informacija koje prikupljaju preko svojih različitih senzora. Ljudi kao korisnici toga često nisu ni svjesni, no ako pogledamo unazad, jasno je da su naši svakodnevni predmeti i okruženja „pametniji“ nego ikad (automobili, mobilni uređaji, kuće, ulična rasvjeta, transport, itd.). Ovi sustavi često komuniciraju između sebe, a s ciljem da bolje analiziraju našu okolinu te savjetuju ili upozore korisnike na određene stvari. Napredniji sustavi često imaju mogućnost i da samo obave dio posla za koji su ma dane informacije i alati. Svi ovi sustavi se uglavnom zasnivaju na odnosu senzor-mikroračunalo-software-korisnik.

Uređaji mogu biti povezani putem interneta, ali i drugih oblika prijenosa kao što su Bluetooth ili RFID (eng. Radio-frequency identification) koji radi na principu radijske frekvencije.

IoT omogućuje tri tipa komunikacije:

- komunikacija stvari (uređaja) s ljudima;
- komunikacija između stvari (uređaja), i
- komunikacije između uređaja (engl. machine to machine / M2M).

Prema istraživačkoj tvrtki Gartner, IoT je jedna od top deset najvažnijih strategijskih tehnologija na svijetu, a tvrtka Cisco je predvidjela preko 50 bilijuna uređaja spojenih na Internet do 2020. godine.



Slika 2 - Trend rasta broja IoT uređaja u svijetu; izvor: Cisco

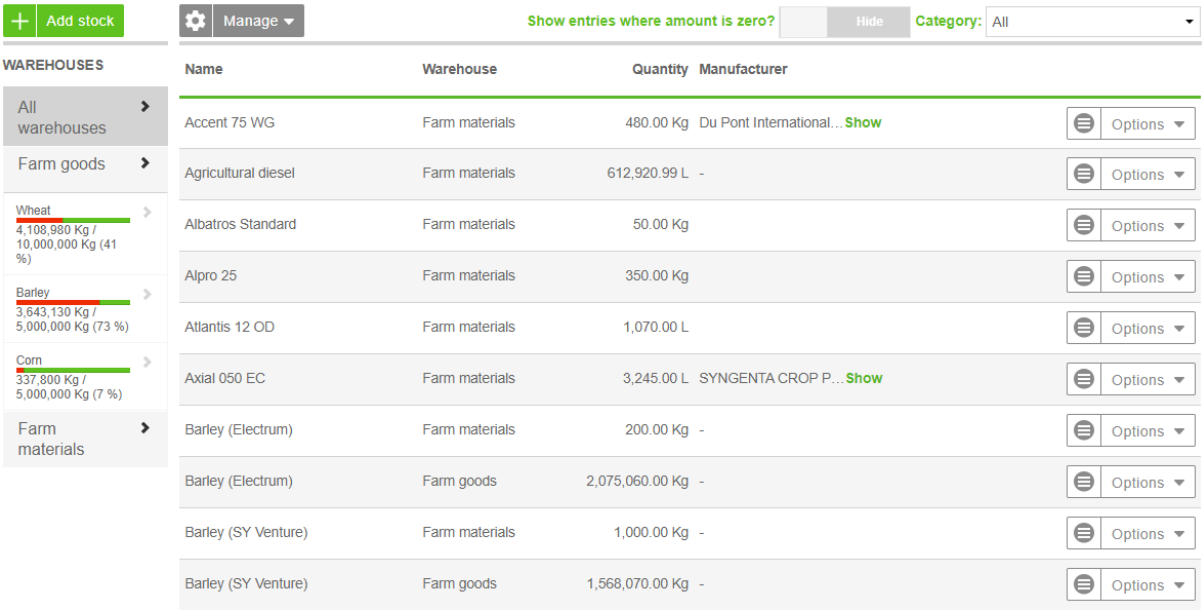
4.2. Primjena IoT-a

Mogućnost primjene IoT-a je praktično neograničena. U bilo kojem sustavu iz kojeg je potrebno dobiti nekakve informacije stanja, iste obraditi te proslijediti računalu ili korisnicima na daljnju manipulaciju, ili na osnovu njih povratno reagirati na sustav, IoT se nameće kao „pametno rješenje“. Sustav može biti izrazito jednostavan, ili s druge strane veoma kompleksan, a ponajviše oviseći o tome koji protokoli su korišteni te koliko individualnih jedinica u sustavu postoji i kakve su njihove funkcije ili mogućnosti. Takvi sustavi su svuda oko nas. Za primjer nekih od njih u kojima IoT već ima čvrstu primjenu možemo navesti pametne kuće, pametna agronomija, pametna proizvodnja (tvornica), logistika, pametni gradovi, m-medicina, i slično.

U logistici se, na primjer, IoT koristi za praćenje i identifikaciju pošiljki. Prilikom svake manipulacije, od zadataka da se pošiljka pošalje, pa do njene isporuke, sustavu je poznato gdje se pošiljka nalazi u skladištu, tko ju je pokupio i u kojem trenutku, u koje vozilo ulazi, kada se očekuje njeno pristizanje u logističke centre, te na koncu njenu zadnju isporuku prema primatelju. Bez mnogo razmišljanja, jasno nam je kako ovakav jedan sustav otvara mnoge mogućnosti optimizacije prilikom logističkih procesa te se postiže organiziranija i brža isporuka, što u konačnici stvara dodatnu vrijednost svim uključenim stranama – pošiljatelju, logistici te primatelju. Primatelj je prilikom cijelog procesa mogao u stvarnom vremenu pratiti što se događa s pošiljkom, budući da kada sustav već posjeduje sve potrebne podatke, pružiti ih online na osnovu jedinstvenog broja pošiljke je poprilično jednostavan zadatak.

Drugi zanimljiv primjer primjene IoT je u poljoprivredi. Moguće je na primjer složiti sustav koji uz pomoć mnogobrojnih senzora nadgleda stanje poljoprivrednog zemljišta te

sprema i obrađuje podatke o vlažnosti tla, količini usjeva, zrelosti i zdravlju usjeva, atmosferskom stanju (vlažnost zraka, tlak, temperatura) itd. Na osnovu ovih podataka sustav može samostalno kontrolirati navodnjavanje ili obavijestiti korisnika o potencijalnom nametniku na usjevu. Sustav može prikupljati i podatke koji nisu direktno sa farme ali su važni za određene farmerske aktivnosti, kao npr. vremenske prognoze koje su dostupne na internetu, stanje o potražnji kulture na tržištu, te tako odrediti idealno vrijeme za sadnju, obradu ili berbu. Sa sličnim izazovima odlučila se suočiti i Hrvatska tvrtka Agrivi, koja između navedenih primjera, omogućava agrarima i kompletno planiranje resursa potrebnih za uzgoj kultura, analizu prikupa plodova, financijsku isplativost, itd.



Slika 3 - Primjer pregleda skladišta i resursa na jednoj farmi; Agrivi.com

4.3. Podatkovni protokoli i komunikacija u IoT-u

Da bi bilo koji IoT sustav funkcionirao stabilno i sigurno, potrebni su podatkovni protokoli koji razmjenjuju podatke između uređaja odnosno poslužitelja sa tim uređajima. Da bismo povezali uređaje koji koriste podatkovne protokole, koriste se komunikacijske tehnike između uređaja, koje onda čine taj uređaj pametnim.

Za komunikaciju između uređaja (D2D, eng. Device to Device) postoje razni podatkovni protokoli koji osiguravaju točnu i pravovremenu razmjenu informacija. Komunikacija između uređaja i servera (poslužitelja) naziva se još i D2S (Device to Server), a između poslužitelja S2S. Neki od protokola su[4]:

- MQTT (eng. Message Queue Telemetry Transport) – protokol za prikupljanje podataka od uređaja i transfer tih podataka na poslužitelj (D2S)

[4] https://www.electronicdesign.com/technologies/iot/article/21798493/understanding-the-protocols-behind-the-internet-of-things

- XMPP (eng. Extensible Messaging and Presence Protocol) – protokol najprikladniji za komunikaciju uređaja i korisnika, poseban slučaj D2S načina zbog toga što su korisnici spojeni na poslužitelje
- DDS (eng. Data-Distribution Service) – služi za brzu komunikaciju između uređaja (D2D)
- AMQP (eng. Advanced Message Queuing Protocol) – Sustav čekanja koji služi za međusobno povezivanje poslužitelja (S2S)

Po pitanju tehnologija preko kojih uređaji komuniciraju postoji relativno širok izbor, ali je uglavnom uvjetovan primjenom. Da bi IoT aplikacija bila široko prihvaćena izrazito je bitno da zadovoljava razne zahtjeve po pitanju robusnosti, stabilnosti, sigurnosti, efikasnosti i naravno kompatibilnosti. Iz tog razloga primjenjuju se široko zastupljeni standardi za bežično povezivanje kao što su Wi-Fi, Bluetooth, NFC, RFID, ZigBee, ANT, i drugi. Kao što je prethodno spomenuto, odabir tehnologije najčešće je uvjetovan primjenom aplikacije ili sustava. Budući da se ove tehnologije međusobno razlikuju po mnogim karakteristikama, u obzir se uzima njihov domet, potrošnja energije, brzina prijenosa podataka, interferencija s drugim uređajima i sl.

4.4. Software i hardware platforme za razvoj IoT-a

Kao i kod komunikacijske tehnologije, izbor software i hardware platforme za IoT najviše utječe aplikacija sustava. Ukoliko se radi o kompleksnijem sustavu, poželjno je koristiti robusnije platforme. To znači da se bira snažniji i stabilniji hardware, mikroračunalo s jačom procesorskom snagom i memorijom, a zatim možda i stabilnošću pri nepovoljnijim radnim uvjetima kao što su ekstremno niske ili visoke temperature okruženja, vibracije, vlaga, i sl. Naravno sve ovo utječe i na ekonomski aspekt projekta, pa ukoliko se radi o jednostavnijem sustavu i primjeni, sukladno tome se bira i prihvatljiviji hardware.

Jedna od popularnijih platformi je Arduino koju karakterizira to što je open source i uključuje razvojnu pločicu sa Atmelovim mikrokontrolerom. Postoji više Arduino inačica koje se primarno razlikuju po tipu mikrokontrolera, a neke od poznatijih su Uno, Due, Mega i Nano.

Malo naprednija platforma od Arduina je Raspberry Pi koju karakterizira znatno moćniji elektronički sklop nego kod Arduina, a nativno je zasnovana na Linuxu, iako je moguće instalirati i neke inačice Windows operativnog sustava.

Pored dvije navedene, vrijedno je spomenuti još i Beagleboard te Intelove Galileo i Edison. Sve ove platforme karakterizira sučelje s višestrukim digitalnim i analognim ulazno/izlaznim portovima, dodatni prateći hardware u obliku prilagođenih kamera, zaslona ili senzora, te u većini slučajeva relativno velika online zajednica korisnika koji međusobno dijele iskustva i pružaju podršku.

Kada promatramo software platforme postoji niz sustava koji su prilagođeni za IoT i razvoj na ugradbenim računalnim sustavima (eng. embedded systems). Danas su najpopularniji Zetta, OpenRemote, Node-RED, ThingsBoard, NoFlo, SiteWhere, Thinger.io i drugi. Većina IoT platformi koristi standardne programske jezike, a najzastupljeniji su C, C++, JavaScript, Java i Python, dok je popularizacija IoT-a dovela i

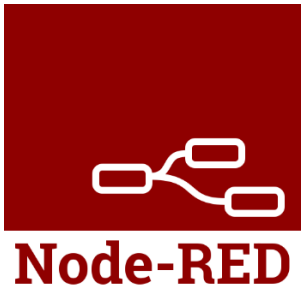
do razvoja programskih jezika namijenjenih baš za razvoj aplikacija u IoT-u (Node.js, Rust).

Uz sve navedeno, uz masivni rast IoT-a pojavilo se i jasno tržište za poslužitelje, a tu dominiraju 3 poznate tvrtke sa svojim proizvodima[5]: Amazon Web Services, Microsoft Azure i Google Cloud Platform. Benefit ovih usluga je jasan kod komercijalnih i kompleksnijih IoT sustava koji prikupljaju i upravljaju izrazito velikim količinama podataka, često tajnih i osjetljivih. Ovakvi poslužitelji osiguravaju skalabilnost, samoodrživost, sigurnost i povjerljivost te izdašnu analitiku.

[5] Eclipse IoT Developer Survey, n1,717,2019; Canalys, 2019

5. NODE-RED PROGRAMSKI ALAT

Node Red je open-source platforma za programiranje slaganjem grafičkih blokova (eng. Visual Programming Language – VPL) koji odrađuju određene programske funkcije. Nastao je 2013. kao manji projekt dvojice IBM-ovih inženjera, Nick O'Leary-a i Dave Conway-Jones-a. U početku je ideja bila da se programom koristi za vizualiziranje i manipuliranje odjeljaka kod MQTT protokola, no brzo je prerastao u mnogo općenitiji alat koji se mogao proširiti u bilo kojem smjeru. Tvorci su brzo shvatili da alat može imati i širu uporabu, te su ga u cilju daljnjeg razvoja odlučili učiniti open-source kako bi i IoT zajednica mogla aktivno doprinositi istom. 2016. godine Node-RED projekt službeno postaje dio JS Fondacije (kasnije OpenJS Fondacija), čiji je jedan od osnivača i IBM.[6] Od tada se Node-RED brzo profilirao kao jedan od boljih VPL alata za programiranje IoT-a.



Slika 4 - Node-RED logotip

5.1. Glavne značajke

Node-RED omogućava rad pomoću Internet preglednika i jedna od glavnih značajki mu je to što je svaka njegova komponenta (blok) svoj vlastiti proces koji ima ulaze i izlaze. Dolazi s izobiljem predefiniranih komponenti koje mogu izvršavati različite procese i obrađivati informacije, a pored toga svaku komponentu je moguće dodatno izmijeniti pomoću tekst editora te u JavaScript-u izmijeniti i prilagoditi funkcije. Ova činjenica ostavlja mogućnost kreiranja procesnih komponenti koje izvršavaju relativno kompleksne zadatke prije nego na izlaz predaju obrađenu informaciju. Ovo omogućava korištenje Node-RED-a na dvije razine, iskusniji programeri koji programiraju i kreiraju vlastite komponente/blokove za obavljanje točno onih funkcija koje su njima potrebne, te nižu razinu za manje vješte poznavatelje kodiranja i programskih jezika. Pozadina alata je građena na Node.js-u, iskorištavajući prednosti njegovog „event-driven, non-blocking“ modela, što ga čini idealnim za pogonjenje na financijski pristupačnijem hardware-u kao što je RPi. Programski tijek (flow) je spremljen koristeći JSON format, što značajno olakšava spremanje i prenošenje podataka, a pogotovo imajući u obzir da se u IoT-u podaci najčešće šalju sa servera na web stranicu. Osim JSON datoteke, cijeli flow se može izvući u obliku šifriranog teksta, koji je dovoljno kopirati i zalijepiti i tako omogućiti prenošenje kompletnog flow-a. Pred toga, urađene biblioteke omogućavaju spremanje korisnih funkcija i predložaka za kasniju uporabu. Ovo se nadovezuje na mogućnost jednostavnog dijeljenja programa (flow-a) unutar zajednice.

[6] https://nodered.org/blog/2016/10/17/js-foundation

5.2. Kompatibilnosti

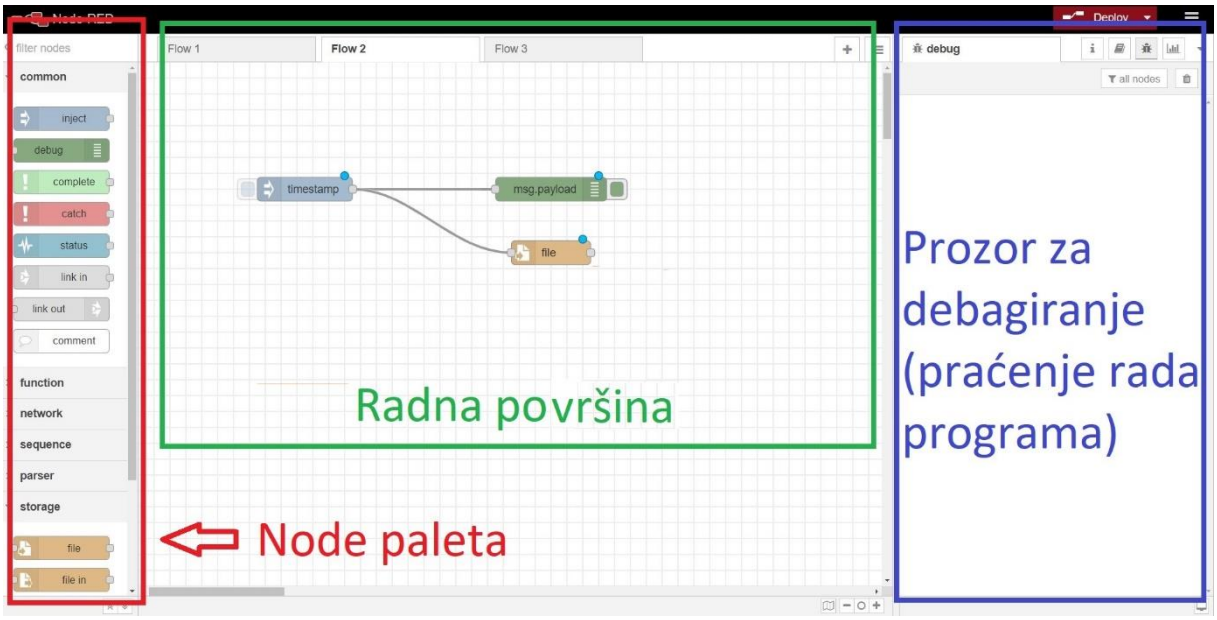
Node-RED je poprilično dobro podržan na raznim platformama, a na službenoj stranici alata nalaze se detaljni vodiči za povezivanje sa svakom od njih. Izdvojit ćemo mogućnosti povezivanja na:

- Lokalno na računalu – potrebna je podržavajuća verzija Node.js, a onda je instalaciju moguće napraviti kroz nekoliko alata kao što su npm (node package manager), Docker, Snap,
- Raspberry Pi mikroračunalima, gdje Node-RED dolazi predinstaliran u Raspbian repozitoriju, te ga je moguće instalirati pozivanjem funkcije kroz komandni terminal,
- Povezivanje s Docker-om je također moguće, što omogućava da se Node-RED koristi za razne arhitekture kao što su amd64, arm32v6, arm32v7, arm32v8 i s390x),
- Arduino uređajima gdje povezivanje ide preko serijskog porta, ili uz instalaciju Firmata protokola ili Johnny-Five biblioteka koje otvaraju mogućnosti poput I2C komunikacije. Također je potrebna podržavajuća verzija Node.js alata.
- BeagleBone, za što je potrebno koristiti Alpha Testing – Debian (10) Buster, a potom ga nadograditi na zadnju verziju,
- Node-RED moguće je pokretati i iz izvornog koda povezivanjem sa git-om, no ovo je namijenjeno prvenstveno programerima koji žele doprinositi razvoju samog Node-RED alata,
- Andraid platformi, uz korištenje Termux aplikacije, iako je ova integracija još uvijek u eksperimentalnom razvoju.

Važno je spomenuti da Node-RED može funkcionirati i u „Cloud“ okruženjima kao što su AWS ili MS-Azure.

5.3. Izgled i dijelovi Node-RED editora

Kao što je prethodno spomenuto, Node-RED editor se pokreće u Internet pretraživaču, što nam odmah otvara glavno radno sučelje. Ono je podijeljeno u tri cjeline: paleta node-ova, radna površina i prozor za debagiranje (praćenje rada programa).



Slika 5 - Radno sučelje Node-RED editor

Node-ovi se načelno dijele na 3 vrste, ovisno o njihovoj funkciji: ulazni, ulazno-izlazni (funkcijski), te izlazni (debug) node-ovi. Moguće ih je lako pretraživati pomoću njihovog naziva, a pored osnovnih koje dolaze perdinstalirane, postoji široka baza dodatnih node-ova koje je moguće preuzeti preko menija i „palette manager“ opcije. Node-ovi se prebacuju u radni prostor „drag and drop“ metodom a povezuju se mišem na svoje spojne točke.

Prozor s desne strane moguće je koristiti za nekoliko važnih stvari, kao što su debugging, konfiguriranje node-ova, informacije o njima, uređivanje dashboard-a, i sl. Više o svemu ovome promotrit ćemo kroz radni zadatak.

6. IMPLEMENTACIJA NODE-RED NA ARDUINO PLATFORMI

6.1. Radni zadatak

Za radni zadatak ćemo napraviti relativno jednostavnu Node-RED aplikaciju koja će u interakciji s Arduinom i senzorom temperature i vlažnosti očitavati podatke te ih u stvarnom vremenu prikazivati u grafičkom sučelju. Za rješenje ovog zadataka bit će potrebno pravilno podesiti i instalirati sve potrebne alate za rad u Node-RED-u na lokalnom računalu i povezati ga s Arduinom te se upoznati s nekim od osnovnih node-ova i njihovoj prilagodbi za našu primjenu

6.2. Pristup rješenju i odabir hardware-a

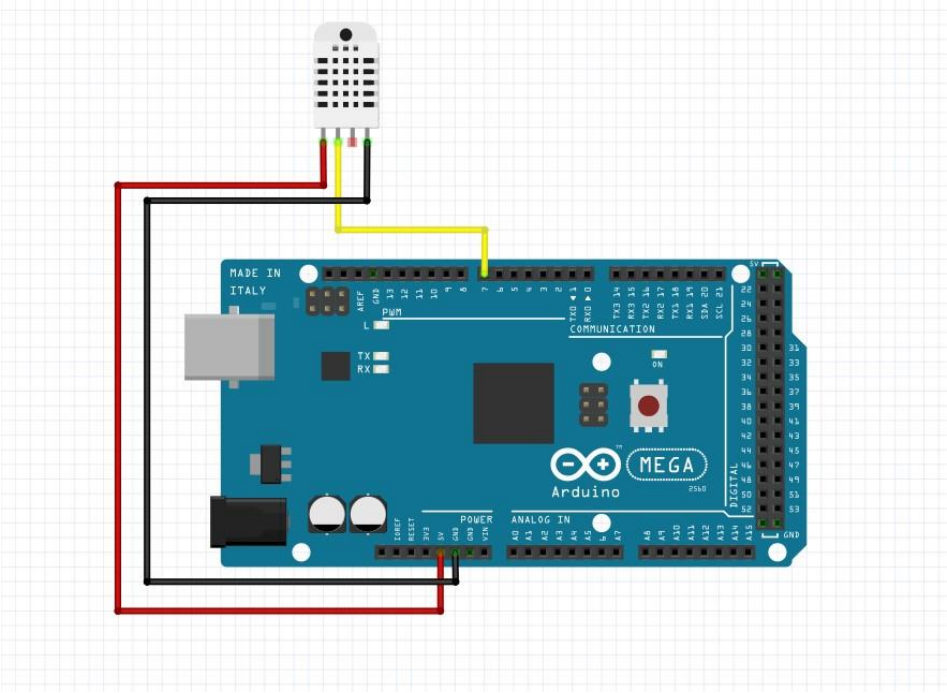
Za rješenje ovog zadatka odabran je slijedeći hardware:

- Mikroračunalo: Arduino Mega 2560
- Senzor vlažnosti i temperature: DHT22
- Žice za spajanje

Da bismo povezali Arduino s računalom na kojem ćemo programirati u Node-REDu, potrebno je koristiti serijsku vezu između Arduina i računala, kao kod programiranja Arduina s Arduino IDE-om, budući da je moguće slati i primati podatke preko ovakve veze. Prvo smo instalirali zadnju verziju Node.js programskog seta preuzetu sa službene stranice nodejs.org te uz pomoć njega instalirali Node-RED. Također, potrebno je napisati kod za Arduino po kojemu će DHT22 slati podatke na serijski izlaz.

6.3. Postupak rješenja zadatka

Senzor vlažnosti i temperature DHT22 spojili smo žicama direktno na Arduino. Za podatkovnu vezu koristili smo PIN 7. Za kreiranje shematskog prikaza korišten je alat Fritzing.



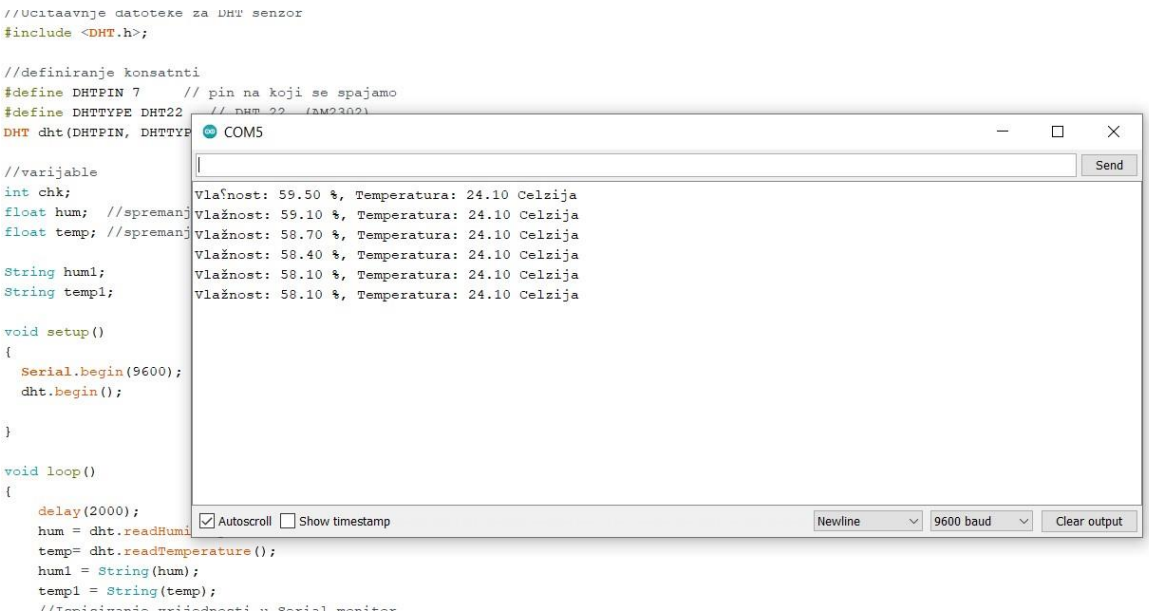
Slika 6 - Shematski prikaz spoja (Fritzing)

Kako bi Arduino, poslije i naša aplikacija, mogla čitati podatke sa senzora, potrebno je napisati jednostavni kod te ga podići na Arduino.

6.4. Programski kod za DHT22 i Arduino

```
//Učitavanje datoteke za DHT senzor
#include <DHT.h>;

//definiranje konstanti
#define DHTPIN 7     // pin na koji se spajamo
#define DHTTYPE DHT22   // DHT 22  (AM2302)
DHT dht(DHTPIN, DHTTYPE); //// Inicijalizacija DHT senzora

//varijable
int chk;
float hum;  //spremanje vrijednosti vlažnosti
float temp; //spremanje vrijednosti temperature

String hum1;
String temp1;

void setup()
{
  Serial.begin(9600); //Boud rate je postavljen na 9600. Bitno je da se
podudara s onim u Node-RED aplikaciji
  dht.begin();
}

void loop()
{
    delay(2000);
    hum = dht.readHumidity();
    temp= dht.readTemperature();
    hum1 = String(hum);
    temp1 = String(temp);
    //Ispisivanje vrijednosti u Serial monitor
    Serial.print("Vlažnost: ");
    Serial.print(hum1);
    Serial.print(" %");
    Serial.print (", Temperatura: ");
    Serial.print(temp1);
    Serial.println(" Celzija");
    delay(2000); //Delay 2 sec.
}
```

Ovaj kod nam ispisuje vrijednosti vlažnosti i zraka svake dvije sekunde, a rezultat izgleda kao što je prikazano na slici 7:

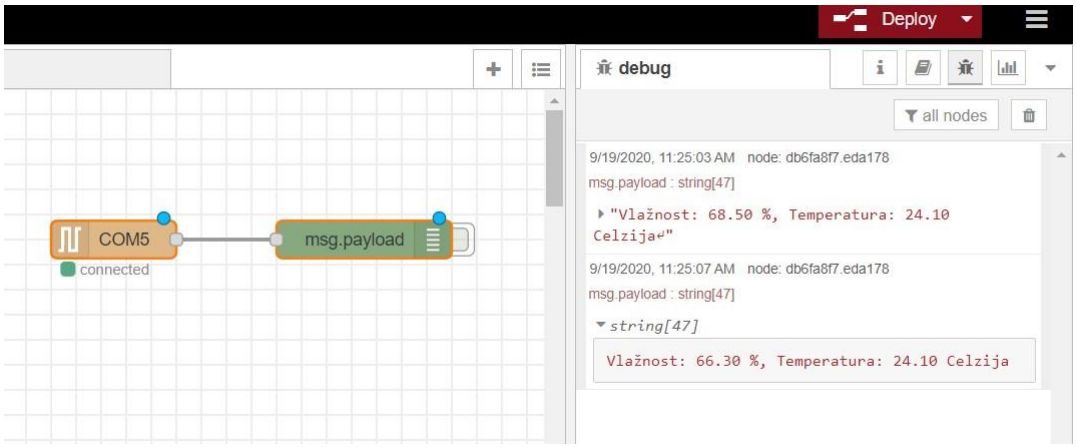


Slika 7 - Prikaz rezultata Arduino koda (Serial Monitor Arduino IDE)

6.5. Kreiranje flow-a u Node-RED-u

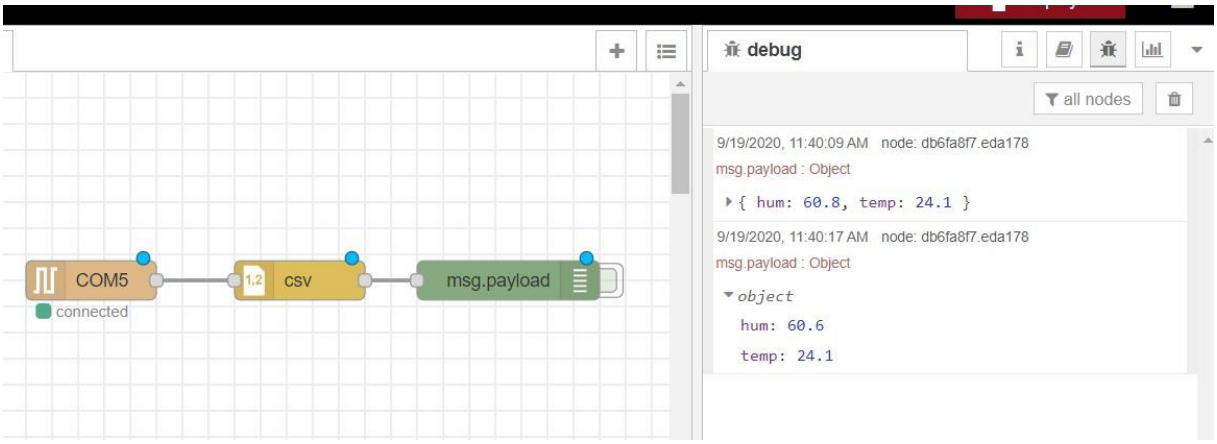
Budući da i Arduino IDE i Node-RED koriste istu serijsku vezu za komunikaciju s pločicom, nije moguće istovremeno imati otvorenu vezu sa Node-RED-om te mijenjati/podizati kod na Arduino, kao ni pratiti izlaz na njegovom Serial Monitor-u.

Da bi bismo ostvarili vezu između sa Node-RED-om, koristimo „Serial input“ node, u kojemu definiramo da preuzima podatke sa istog porta na kojem je Arduino spojen na računalo, u našem slučaju COM5. Provjeravamo da je veza uspješna i da podatci stižu u program tako što privremeno povezujemo „debug“ node.



Slika 8 - Povezivanje s Arduinom i čitanje podataka

Kao što je vidljivo na slici, podaci koje Node-RED dobija su zapisani u obliku jednog stringa. Da bismo ih mogli koristiti za prikaz u našoj aplikaciji, potrebno je „pročistiti“ ulaznu poruku u program. To ćemo napraviti sa CSV node-om (Comma-Separated Values) tako što ćemo izbaciti nepotrebne riječi i oznake jedinica, a dio stringa koji predstavlja vrijednosti ćemo dodijeliti novim oznakama „hum“ i „temp“. Novi rezultat je vidljiv na desnoj strani slike 9.

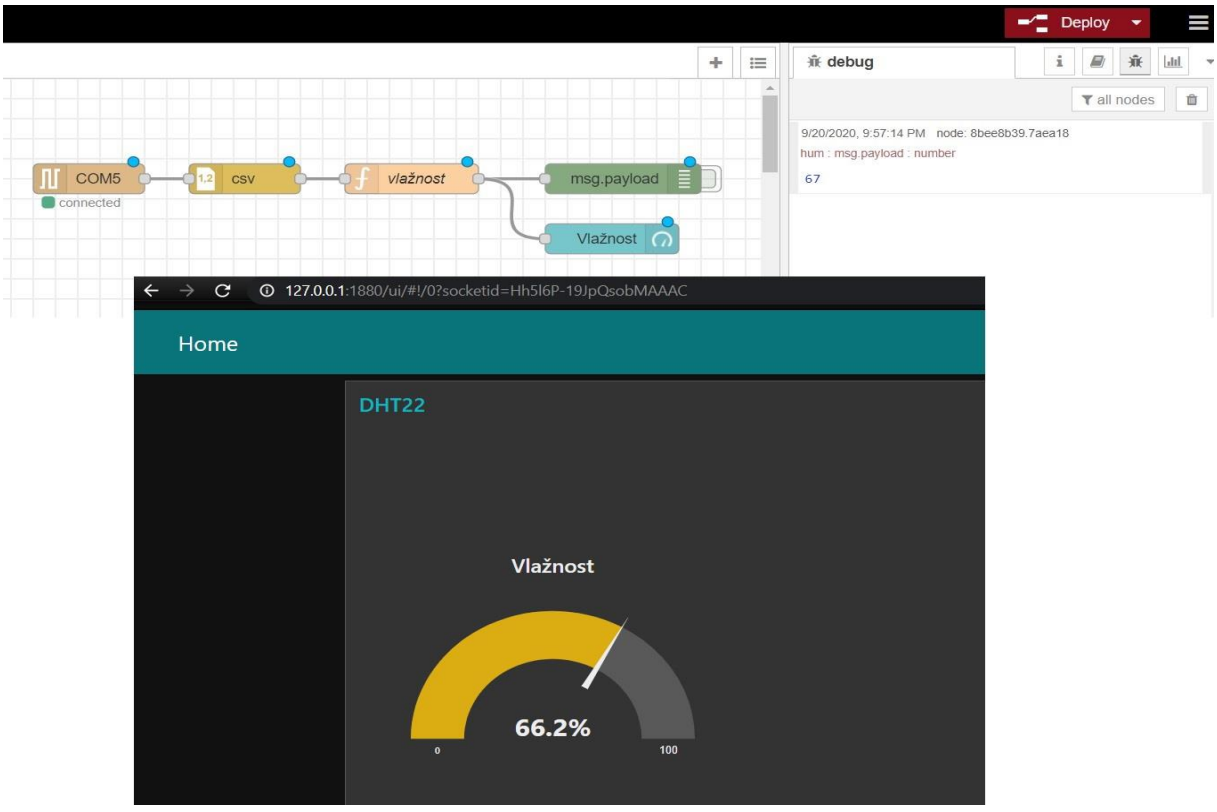


Slika 9 - "Čišćenje" poruke pomoću CSV node-a

Sada imamo jasniju poruku koja je podijeljena u 2 reda, no i dalje nam ne odgovara što su vrijednosti sa senzora zapisane kao string. Da bi dashboard mogao primiti podatke, potrebno ih je pretvoriti u pravu brojčanu vrijednost. To ćemo napraviti sa funkcijskim node-om koji ćemo prilagoditi za našu potrebu jednom jednostavnom linijom JavaScript koda:

```
var msg1 = {topic:"hum", payload:msg.payload.hum}
return msg1;
```

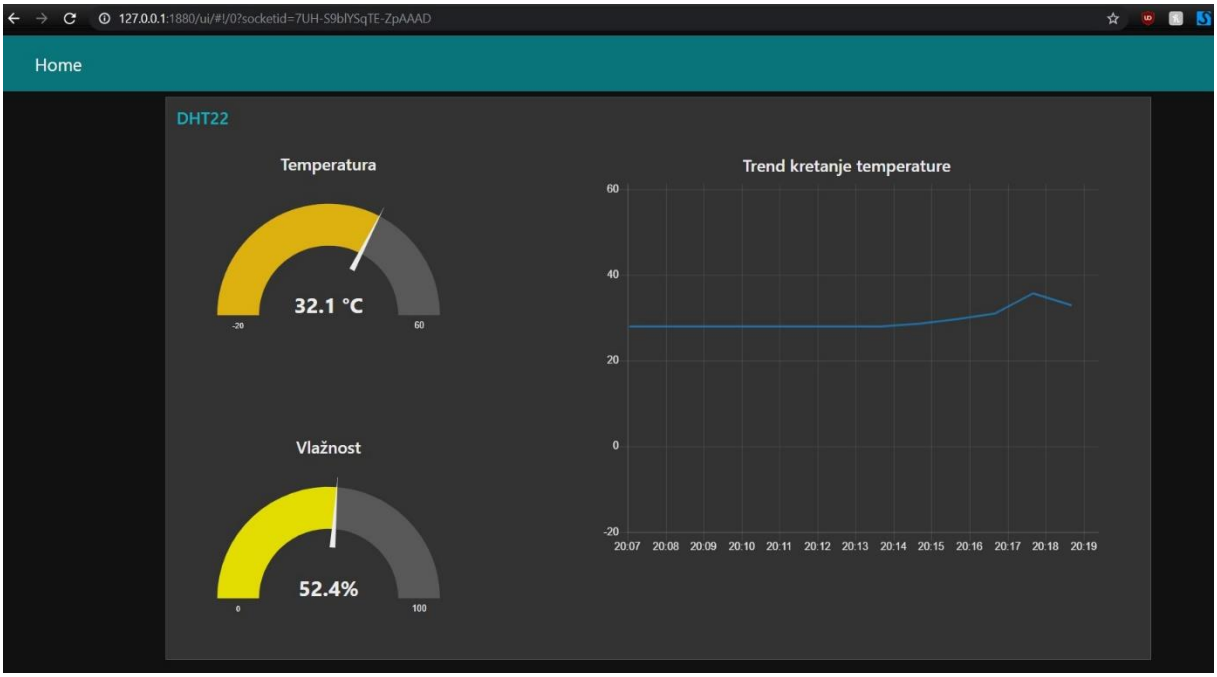
Kada imamo brojčanu vrijednost koju grafički node-ovi prihvaćaju, možemo dodati izlazne grafičke elemente koji će sačinjavati završni izgled našeg dashboarda. Za početak ćemo dodati node pod nazivom „gauge“, odnosno mjerač, a rezultat izgleda kao na slici 10.



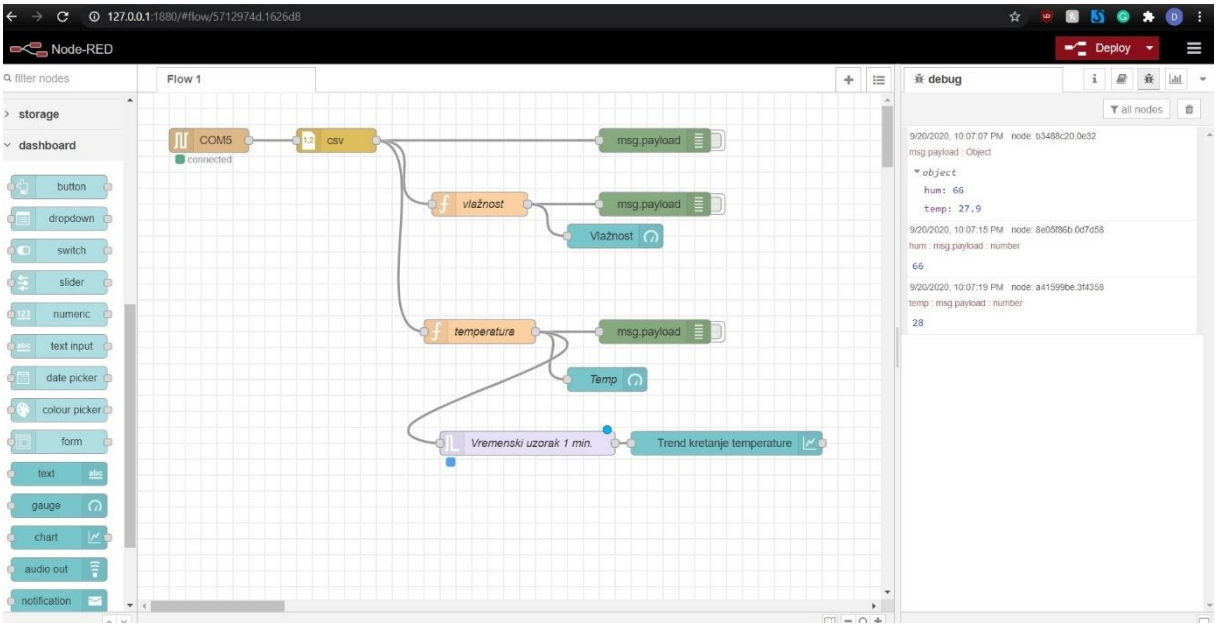
Slika 10 - Prikaz rezultata u dahsboardu s jednim mjeračem

U nastavku zadatka dodat ćemo još i mjerač temperature, te graf u kojem se bilježi trenutna temperatura. Budući da program mjeri temperaturu svake dvije sekunde, graf bi nam izgledao pretrpan s jako malim izmjenama temperature po svakom zapisu. Stoga smo iskoristili „trigger“ node koji smo podesili na način da u graf propušta vrijednost tek svake minute, a izmjenom jedne brojke, ovaj okidač se može podesiti i na bilo koji drugi željeni interval.

Konačan rezultat izgleda dashboard-a vidljiv je na slici 11, a njegov flow na slici 12.



Slika 11 - Završni izgled dashboarda



Slika 12 - Završni flow programa

7. ZAKLJUČAK

Računalne znanosti rastu i šire se eksponencijalnom brzinom. S napretkom tehnologije koja postaje sve pristupačnija, sve više populacije ima dodira s njom, makar u nekom osnovnom obliku. IoT kao grana industrije koja objedinjuje računalnu i komunikacijsku tehnologiju sa sveprisutnijim pametnim uređajima, postaje gotovo neizbježan sastavni dio svakog korisnika modernije računalne tehnologije, pri tome pružajući mu dodatnu vrijednost. Eksponencijalan porast broja pametnih uređaja iz godine u godinu stavlja imperativ na osmišljavanje i dizajniranje sustava koji su sposobni sami pratiti i prikupljati podatke, iste obrađivati te na osnovu unaprijed određenih pravila i okvira, sami i izvršavati određene zadatke. S porastom korisničke populacije, stvorila se i jasna potreba za pojednostavljenjem načina i modela programiranja za IoT. Sukladno tome, kroz ovaj rad istražili smo tijek razvoja IoT, modela programiranja za isti te smo pokazali da je danas relativno jednostavno osmisliti i isprogramirati jednu IoT aplikaciju koja će obavljati određenu funkciju za korisnika. Uz proširenje znanja rastu i mogućnosti, pa je moguće kreirati i velike i kompliciranе IoT sustave koristeći open-source alate dostupne svima.

Uzimajući sve u obzir, očigledno je da IoT postaje veliki posao budućnosti i doprinosi sveopćoj digitalizaciji koja podrazumijeva senzore u svim područjima, život u „cloudu“ i internetsku povezanost ne samo nas već i svega što nas okružuje.

8. POPIS SLIKA I TABLICA