NADZOR PARAMETARA BRODICE KORIŠTENJEM LoRaWAN TEHNOLOGIJE

Rodić, Antonio

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Stručni diplomski studij Elektrotehnika

ANTONIO RODIĆ

ZAVRŠNI RAD

NADZOR PARAMETARA BRODICE KORIŠTENJEM LoRaWAN TEHNOLOGIJE

Split, Rujan 2024.

SVEUČILIŠTE U SPLITU

SVEUČILIŠNI ODJEL ZA STRUČNE STUDIJE

Stručni diplomski studij Elektrotehnika

Predmet: Senzorske mreže

ZAVRŠNI RAD

Kandidat: Antonio Rodić

Naslov rada: Nadzor parametara brodice korištenjem LoRaWAN tehnologije

Mentor: Marko Meštrović

Split, Rujan 2024.

SADRŽAJ

Sažetak

Nadzor parametara brodice korištenjem LoRaWAN tehnologije

Zadatak ovog rada je bio da se prikupe podatci sa senzora, konkretno BMP180 senzora za tlak i temperaturu, GPS modula(Velleman VMA430 - GPS / GLONASS u-Blox NEO-7M UART modul) te detektora plamena te se prikupljeni podatci pošalju LoRaWAN-om na Datacake platformu. Koristili smo ESP32 pločicu za obradu podataka te je programirali pomoću Arduino IDE-a. Podatci se šalju na The Things Network gdje smo kreirali aplikaciju i unutar aplikacije uređaj (end device). Identifikatore NWKSKEY, APPSKEY, DEVADDR za OTAA način aktivacije generira TTN, koriste se za spajanje na mrežu. Pomoću „payload decodera“ podatci se dekodiraju i prikazuju u aplikaciji na The Things Networku. TTN cloud se povezao s datacake platformu pa se podatci prosljede na platformu i prikazuju na dashboardu platforme. U radu se nalazi detaljan opis svako djela ovog projekta.

Ključne Riječi: LoRaWAN, LoRa, ESP32, The Things Network, Datacake

Summary

Monitoring of boat parameters using LoRaWAN technology

The task of this work was to collect data from sensors, specifically the BMP180 sensor for pressure and temperature, the GPS module (Velleman VMA430 - GPS / GLONASS u-Blox NEO-7M UART module), and a flame detector, and then transmit the collected data via LoRaWAN to the Datacake platform. We used an ESP32 board for data processing and programmed it using the Arduino IDE. The data is sent to The Things Network, where we created an application and an end device within that application. The identifiers NWKSKEY, APPSKEY, and DEVADDR for OTAA activation mode are generated by TTN and are used for network connection. Using the "payload decoder," the data is decoded and displayed in the The Things Network application. The TTN cloud is connected to the Datacake platform, so the data is forwarded to the platform and displayed on the platform’s dashboard. This work includes a detailed description of each part of this project.

Keywords: LoRaWAN, LoRa, ESP32, The Things Network, Datacake

1.UVOD

Razvojem Internet of Things-a (IoT) posljednih godina svjedoci smo brzog razvoja novih inovacija kao i njihovih primjena, kako u gospodarstvu, znanosti tako i u svakodnevnoj upotrebi. IoT donosi brojne prednosti, uključujući povećanje efikasnosti, smanjenju troškova, praćenju raznih parametara i slično. Pametni gradovi koriste IoT za optimizaciju resursa, smanjenje zagađenja i poboljšanje kvaliteta života svojih stanovnika. U poljoprivredi, IoT senzori omogućavaju precizno praćenje usjeva i optimizaciju navodnjavanja, čime se povećava prinos i smanjuje potrošnja vode. Digitalizacijom svakodnevnog života svakako podižemo kvalitetu života. Korisnicima nije prihvatljivo izvoditi velike infrastrukturne radove za ugradnju uređaja, pa se teži bežičnoj komunikaciji. Ona omogućuje brzo i jednostavno postavljanje uređaja kako bi se iskoristio puni potencijal uređaja. Danas postoje različite bežične tehnologije koje se koriste , a svaka od njih ima drukčije karakteristike. Najčešće korištene su WiFi, Bluetooth, GSM mreža, LoRaWAN, Sigfox i NB-IoT.

LoRaWAN mreža se pokazala kao idealna za uređaje koji funkciju obavljaju na mjestima gdje nema stalnog izvora napajanja jer ima iznimno malu potrošnju. Također ne može slati velike količine podataka to je čini idealnom za primjene poput meteoroloških stanica, mjerača vlage tla, nadzoru brodica i slično. Upravo je izrada sustava za nadzor brodice preko LoRaWAN-a cilj ovog rada. Cilj je prikupiti podatke sa senzora brodice i poslat ih preko LoRaWAN mreže te na kraju prikazati u Datacake platformi.

2. LORA

2.1. Uvod o LoRa

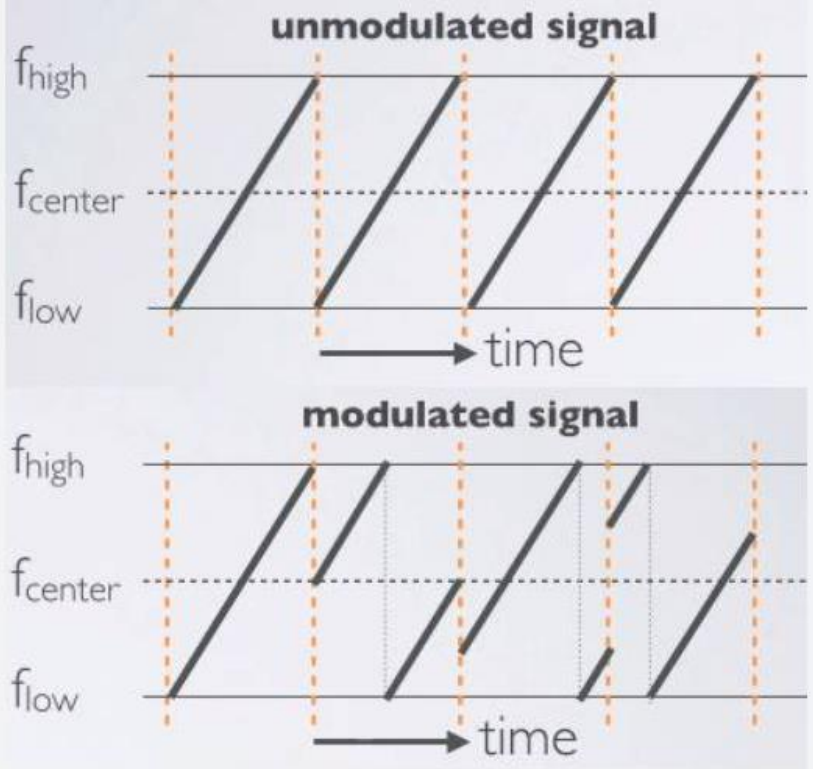
LoRa je kratica za Long Range Radio, riječ je o bežičnoj tehnologiji koja služi za prijenos male količine podataka na velike udaljenosti, uz malu potrošnju energije. Uglavnom je namijenjena za IoT (Internet of Things) te M2M (machine to machine) tehnologije. LoRa tehnologiju razvila je francuska kompanija „Cycleo“ 2009. godine, a 2012.otkupila je tvrtka Semtech. 2015. godine LoRa je standardizirana od strane LoRa Alliance.

Veliki domet te niska potrošnja energije glavne su prednosti ove tehnologije, te čine LoRaWAN mrežu jednom od najraširenijih. LoRa koristi frekvencijski spektar koji je rezerviran za industrijske, znanstvene i medicinske svrhe. Razlika između LoRaWAN i LoRa je taj da LoRaWAN predstavlja komunikacijski protokol, a LoRa fizički sloj.

LoRa je definirana kao tehnika širokog spektra modulacije tzv. Chirp Spread Spectrum (CSS) koja djeluje na fizičkom sloju LoRaWAN mreže i predstavlja rješenje za LPWAN mreže.

Chirp Spread Spectrum je tehnika raširenog spektra koja za kodiranje informacija koristi širokopojasne linearne frekvencijski modulirane chirp impulse. Chirp, odnosno skraćenica od Compressed High Intensity Radar Pulse

Predstavljena je tonom u kojem se frekvencija povećava od minimalne prema maksimalnoj (up-chirp) ili smanjuje od maksimalne prema minimalnoj (down-chirp) tijekom vremena, kao što je prikazano na slici 2.1. Kod up-chirpa frekvencija se povećava tijekom vremena i kada je postignuta maksimalna frekvencija spušta se do najmanje te se proces ponavlja ispočetka. Down-chirp je postupak inverzan up-chirpu.



Slika 2.1. Izgled signala u CSS modulaciji. [18]

Zbog linearnosti chirp signala, frekvencijska odstupanja između prijemnika i odašiljača jednaka su vremenskim odstupanjima. Ova karakteristika čini modulaciju imunu na Dopplerov efekt. Odstupanje frekvencije između odašiljača i prijemnika može doseći 20% širine pojasa bez utjecaja na performanse dekodiranja. To pridonosi u smanjenju ukupne cijene LoRa odašiljača, jer kristali ugrađeni u odašiljače ne moraju biti izrađeni do krajnje točnosti.[8]

Spreading Factor definira broj bitova koji predstavljaju simbol.[7]

Zaključak je da CSS modulacija ima brojne prednosti: Velik domet, pouzdanost, otpornost na smetnje, mala potrošnja energije te precizno lociranje korisnika. [8]

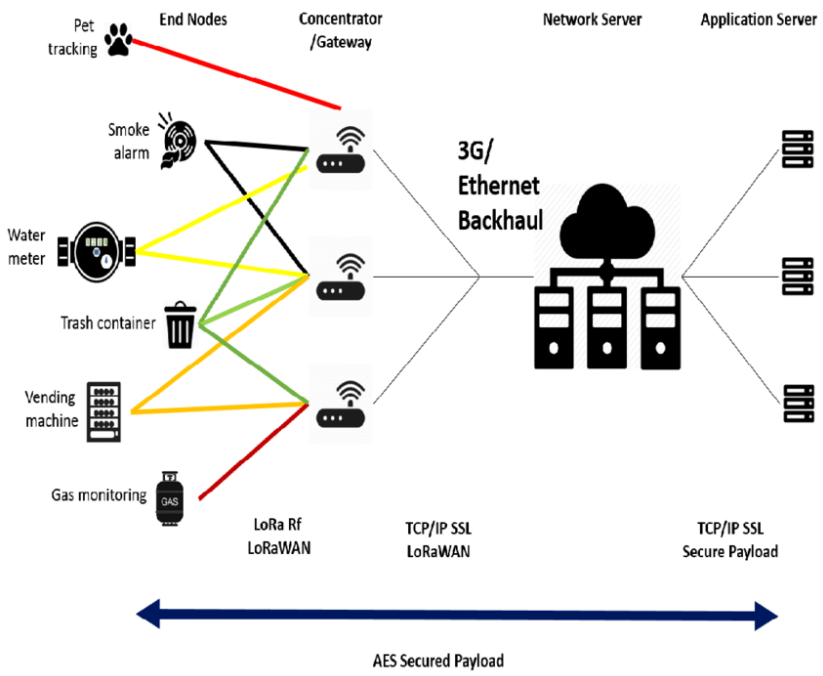
2.2 LoRaWAN

LoRaWAN definira komunikacijski protokol i arhitekturu sustava mreže, dok fizički sloj LoRa-e omogućuje komunikacijsku vezu velikog dometa.

Protokol i mrežna arhitektura najviše definiraju parametre poput: trajanja baterije čvora, kapaciteta mreže, kvalitete usluge, sigurnosti i raznolikosti aplikacija. LoRaWAN tehnologija koristi tzv. Star network topologiju, što znači da postoje centralni čvorovi koji su povezani s IoT uređajima.

LoRaWAN arhitektura se sastoji od:

- Krajnji uređaj, čvor (node) - se koriste za mjerenje, a ponekad i za daljinsko upravljanje, to su nekakvi senzori i aktuatori koji se sastoje od antene (primopredajnika) i mikrokontrolera. Male su snage te komuniciraju bežično s jednim ili više gateway-a. Mikrokontroler ima funkciju upravljanja čvorom, tj odlučuje hoće li primati ili slati podatke. Antena (primopredajnik) služi da bi se podatci mogli slati odnosno primati između čvora i gateway-a.
- Gateway – su zapravo antene koje omogućuju komunikaciju između dva krajnja uređaja.
- Mrežni poslužitelj (engl. network server) – poslužitelji koji usmjeravaju poruke s krajnjih uređaja do odgovarajuće aplikacije i natrag.
- Aplikacija – dio softvera, pokrenut na poslužitelju

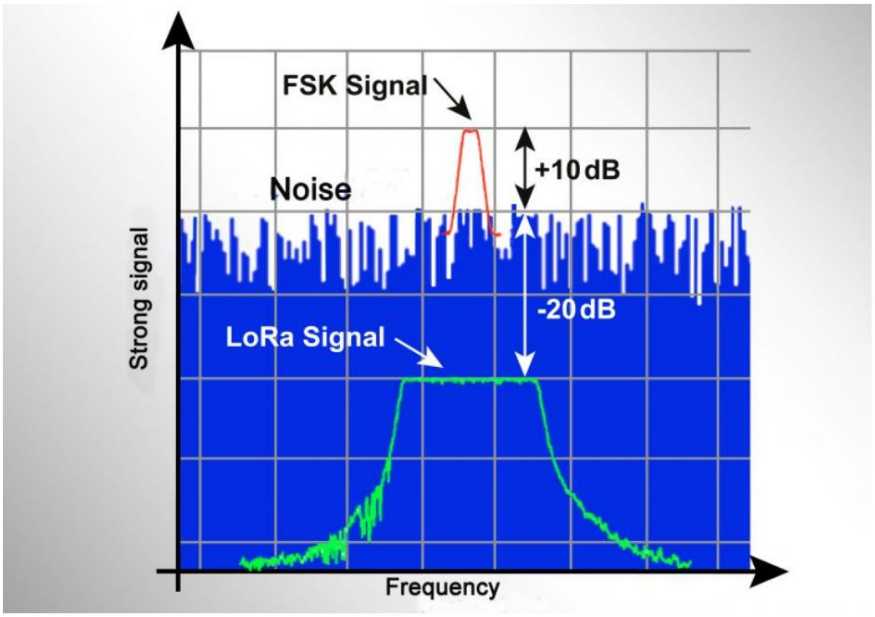


Slika 2.2. Arhitektura LoRaWAN mreže [5]

2.2.1 Frekvencije rada kod LoRaWAN mreže.

LoRaWAN radi na frekvencijskim ISM pojasevima. Budući da LoRa radi u ISM frekvencijskim pojasevima (433 MHz, 868 MHz i 915 MHz), snaga prijenosa zračenja je ograničena.

Kako bi imali veći radio raspon od konvencionalnih tipova modulacije kao što je za postizanje FSK (digitalna frekvencijska modulacija), osjetljivost prijemnika je značajno poboljšana s LoRaWAN-om. LoRa prijemnik i dalje može uspješno primati i dekodirati koristan LoRa signal do 20 dB ispod razine šuma (noise), što rezultira osjetljivošću prijemnika od maksimalno -149 dBm. U usporedbi s maksimalnom FSK osjetljivošću od cca. –125 dBm do -130 dBm, LoRa nudi značajno poboljšanje. Slika 2.3 prikazuje usporedbu FSK i LoRa signala. Zahvaljujući ovome svojstvu LoRa je puno robusnija od klasične FSK modulacije.[6]



Slika 2.3. Razlika LoRa signala i klasičnog FSK signala.[6]

Međutim, postoje dva ograničenja:

• LoRa modulacija ima širi spektar od FSK modulacije, što znači da je šum kod LoRa prijamnika jači nego kod FSK prijamnika.
• LoRa može primiti samo koristan signal do 20 dB ispod razine buke pri vrlo malim brzinama prijenosa podataka od ≤ 0.5 kbit / s. Čim se brzina prijenosa podataka poveća, ili se negativni omjer signal-šum dalje povećava prema nuli ili se širina pojasa mora dodatno povećati, što zauzvrat povećava razinu buke.

2.2.2. Brzine prijenosa kod LoRaWAN-a

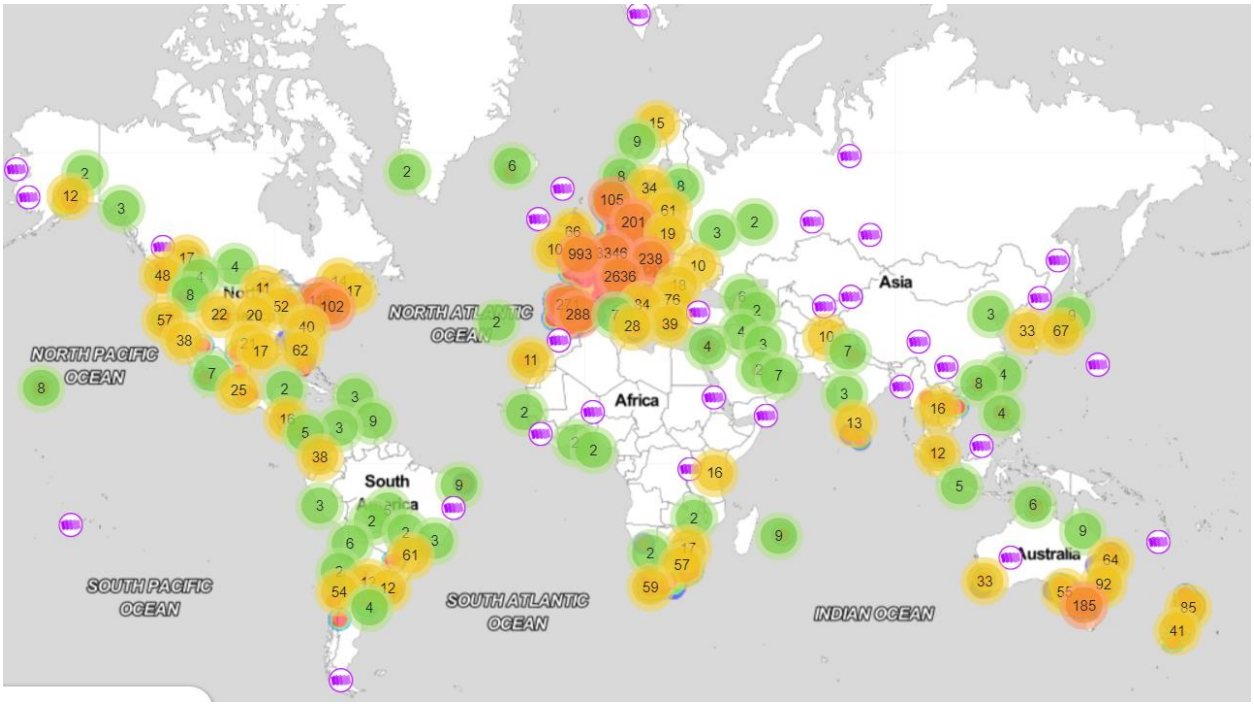
Kao i kod ostalih mreža brzina prijenosa podataka (data rate) se gleda kao količina podataka u nekoj jedinici vremena. Bitno je znati sljedeće pravilo da brzina prijenosa i udaljenost na koju se prenosi su obrnuto proporcionalni.

Kod brzine imamo dvi brzine prijenosa jedna brzina je brzina na fizičkom sloju, a druga je brzina prijenosa podataka na mrežnom nivou. Brzina na fizičkom sloju je doslovno broj bitova u sekundi koji se prebace nekim medijem. Brzina mreže je količina podataka koja se prebaci mrežom u jedinici vremena.

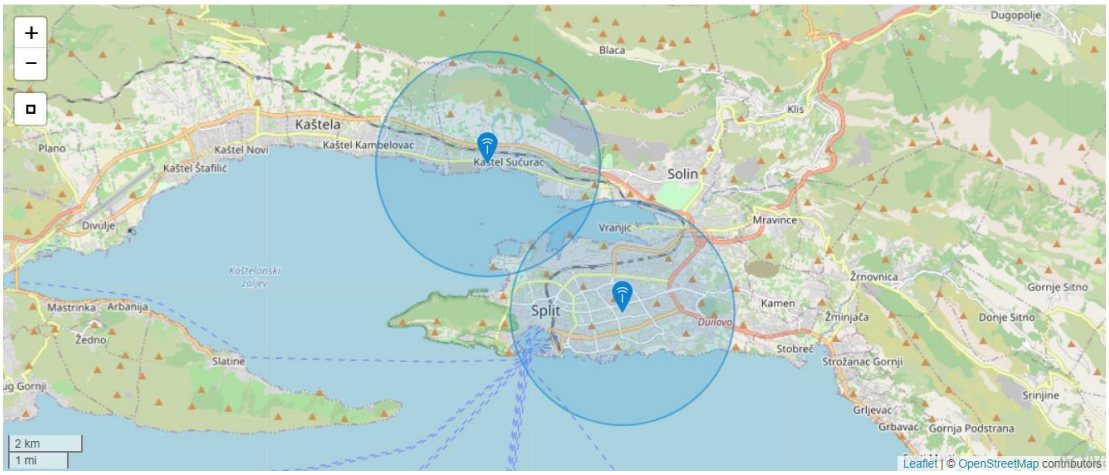
Brzina na fizičkom sloju kod LoRaWAN-a iznosi od 0.3 kbps do 50-tak kbps. Fizičku brzinu možemo prilagoditi određenim potrebama aplikacije, a ovisi o odabiru modulacije i širini kanala (bandwidth).

2.2.3. Rasprostranjenost LoRaWAN-a

Veoma bitna stavka kad gledamo mrežu općenito je rasprostranjenost. Prema podatcima LoRa Alliance-a postoji 156 operatora u 142 zemlje svijeta koje nude LoRaWAN usluge. Ovo je trenutna brojka koja raste iz dana u dan.[17]



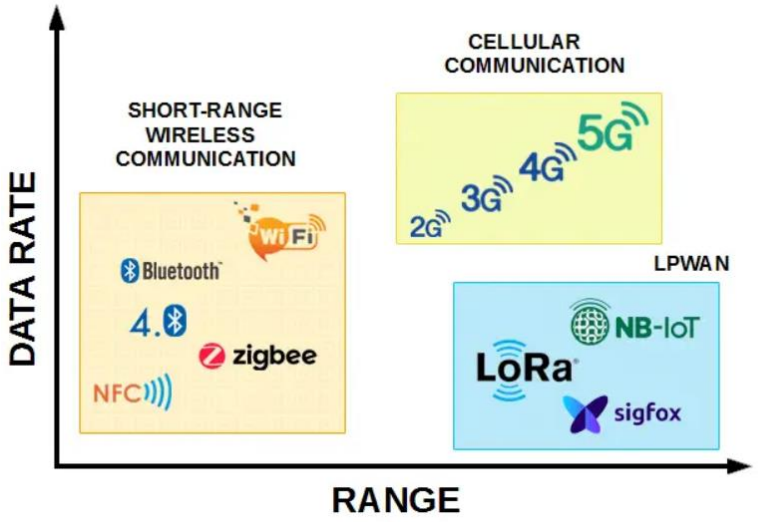
Slika 2.4. Broj Gateway-a u svijetu. [16]



Slika 2.5. Rasprostranjenost LoRaWAN-a u gradu Splitu. [15]

2.2.4. Usporedba s drugim mrežama

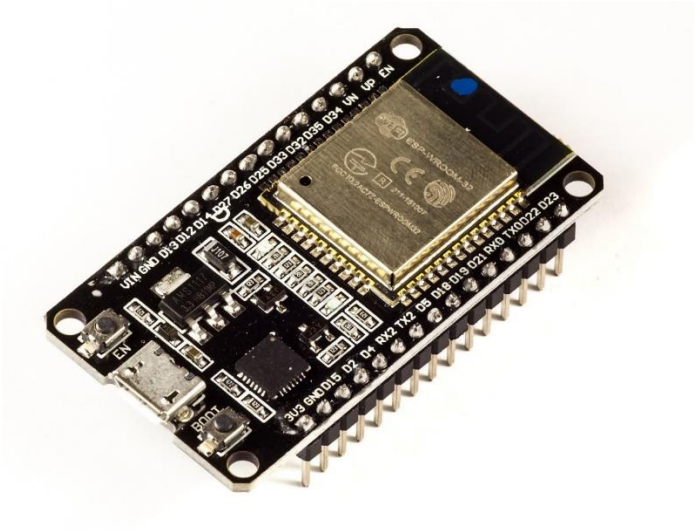
Razvojem IoT-a razvila se i potreba za tehnologijama koje omogućuju prijenos podataka, ali zbog raznovrsnih potreba jedna tehnologija nije dovoljna. LoRaWAN im prednosti u vidu udaljenosti na koju može poslati podatke i malu potrošnju energije, ako je uspoređujemo s drugim mrežama poput WIFI-a ili Bluetooth-a.



Slika 2.6. Usporedba tehnologija gledajući domet i brzinu prijenosa.[19]

3. ESP32

ESP32 je SoC(System of Chip) mikrokontroler, koji se koristi za povezivanje uređaja s internetom. Predstavlja nadogradnju ESP8266, jer ima više funkcija, veću snagu i bolju povezanost. U ovom seminarskom radu ćemo predstaviti funkcije i karakteristike ESP-WROOM-32 te njegovu primjenu. ESP32 je razvila kineska kompanija Espressif Systems company .



Slika 3.1. Izgled ESP32 pločice[11].

Trenutno na tržištu ima nekoliko verzija i modela ESP32 mikrokontrolera. Korisnik bira ovisno o potrebi koji od modela je najbolji za željenu primjenu. Neki od modela su:

1. ESP32-WROOM-32 - Ovo je standardna verzija ESP32 modula koja sadrži sve potrebne funkcije u jednom paketu, uključujući WiFi i Bluetooth.
2. ESP32-PICO-D4 - Ovaj modul je manji od ESP32-WROOM-32, ali također sadrži WiFi i Bluetooth funkcionalnosti. Ima ugrađenu antenu i manje je sklopova potrebno za spajanje s drugim komponentama.
3. ESP32-S2 - Ovo je verzija ESP32 koja ne sadrži Bluetooth, ali ima poboljšane performanse WiFi veze i manju potrošnju energije. Ima manje I/O pinova nego ESP32-WROOM-32.
4. ESP32-CAM - Ovaj modul ima ugrađenu kameru i WiFi funkcionalnost, što ga čini idealnim za projektiranje bežičnih kamera za sigurnosne svrhe.

5. ESP32-S3 - Ovo je nova verzija ESP32 koja nudi još bolje performanse u pogledu WiFi povezivanja, veću brzinu procesora i više I/O pinova nego prethodne verzije.
6. ESP32-SOLO-1 - Ovaj modul ima samo jednozvučnu funkciju WiFi, ali nudi manju potrošnju energije i manji form faktor.
7. ESP32-WROVER-B - Ovo je verzija ESP32 modula koja ima više prostora za pohranu, uključujući RAM i Flash memoriju, što je čini idealnom za projektiranje složenijih aplikacija.

Slika 3.2. Modeli ESP32 mikrokontrolera.[13]

3.1. ESP-WROOM-32

ESP-wroom-32 je veoma moćan modul koji se može koristiti za razne primjene od onih jednostavnih, odnosno upravljanja jednostavnim senzorskim mrežama, do puno složenijih zadataka. Modul se temelji na ESP32-D0WDQ6 čipu koji sadrži dva Xtena32-bitna LX6 mikroprocesora male snage.

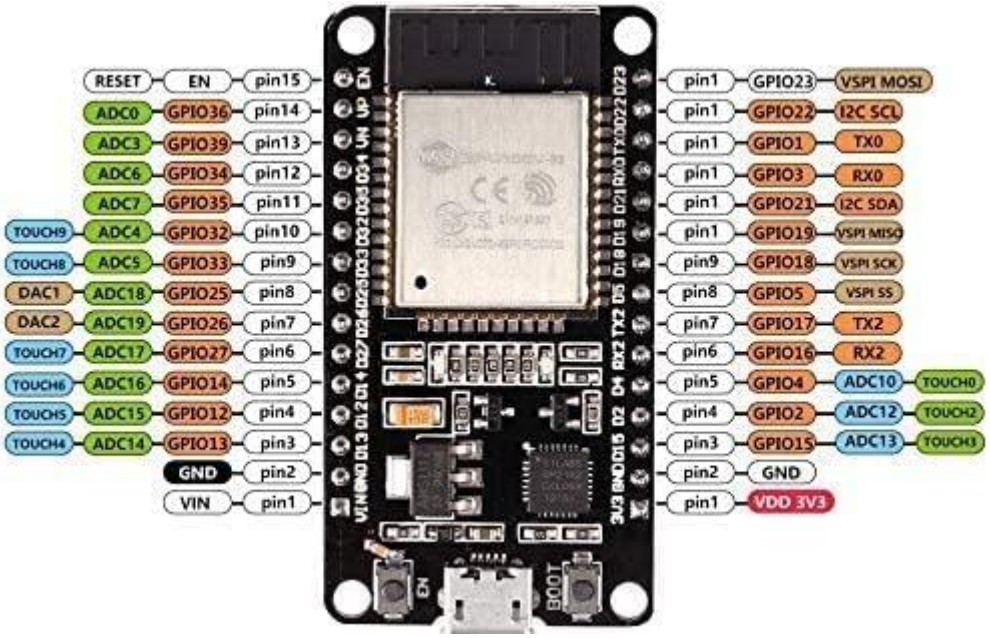
ESP32 ima 520 K B radne memorije, 12-bitne analogno-digitalne pretvarače, dva 8-bitna digitalno-analogna pretvarača, deset senzora dodira (kapacitivnost), temperaturni senzor, četiri puta SPI, dva puta I2C sučelja, tri puta UART sučelja, PWM, Hallov senzor (engl. Hall effect sensor), LED PWM i slično.

ESP32 je dizajniran za mobilne, nosive elektroničke naprave te Internet of Things primjene. Ima opcije za uštedu energije, uključujući više različitih 5 postavki potrošnje. ESP32 se također može podesiti da se aktivira periodički ili kada se ostvari određeni uvjet što će isto smanjiti potrošnju energije. [21]

Pločicu je moguće programirati putem USB priključka i Arduino IDE sučelja, no i putem nativnog ESP-IDF alata te ograničeno putem MicroPythona, Javascripta odnosno PictoBlox grafičkog sučelja koje je izvedeno i Scratch 3.0. ESP-IDF moguće je povezati s najpopularnijim editorima koda kao što su Visual Studio Code i PlatformIO plugin te Eclipse ili vašim drugim omiljenim editorima koda.

Arduino IDE podržava programiranje ESP32 uz malo podešavanje, a programira se kao i Arduino pločice direktno iz C++ jezika.

Samo programiranje ne razlikuje se previše od onog za Arduino, ali potrebno je znati da je ESP32 mnogo moćniji od Atmega328p mikroprocesora koji se nalazi na većini Arduino pločica, mogućnosti ESP32 mikroračunala su neusporedivo veće, pogotovo jer sadrži Wi-Fi i BT komunikaciju te omogućuje uparivanje više takvih pločica međusobno. [22]



Slika 3.3. Funkcije pojedinih pinova na ESP-WROOM-32 modulu.[20]

4. DATACAKE PLATFORMA

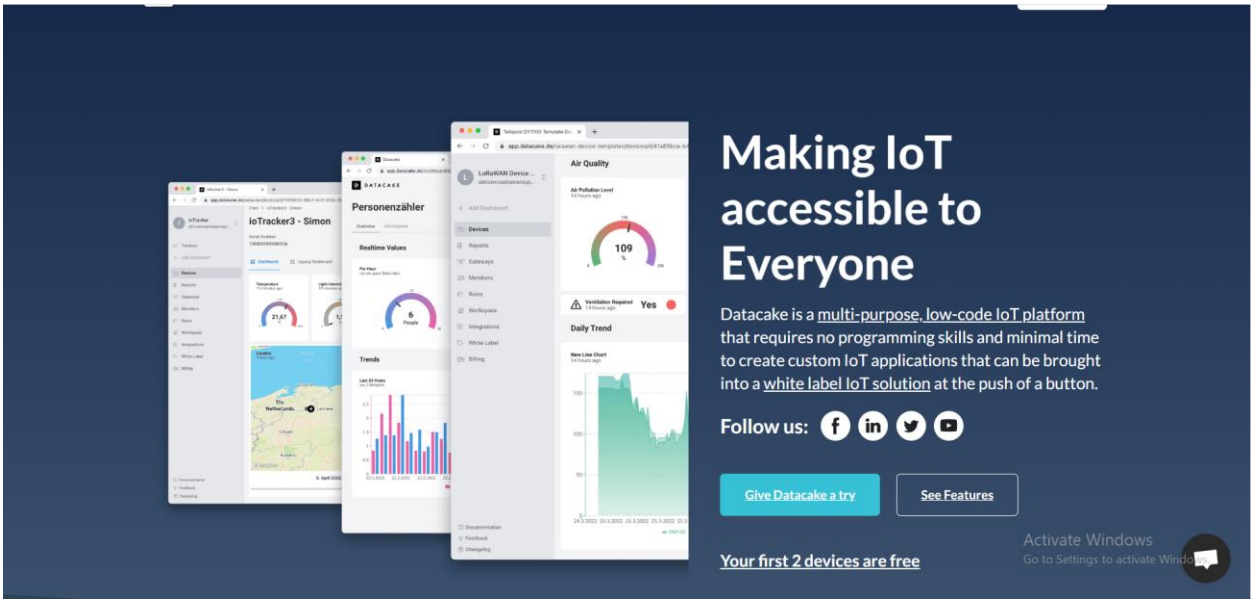
Slika 4.1.DataCake logo.

U današnje vrijeme, podaci su postali jedan od najvažnijih resursa za poslovanje. Sve više tvrtki prepoznaje važnost analize podataka kako bi donijeli bolje odluke, te imali uvid i mogućnosti poboljšanja svojih poslovanja. Automatizirane IoT aplikacije generiraju gotovo neiscrpne količine podataka 24 sata dnevno. Kako bi se industrijski procesi učinili učinkovitijima, utvrdile korelacije ili spriječili kvarovi, informacije i izmjerene vrijednosti moraju se prikupljati, pohranjivati i učiniti dostupnima za analizu.

Obrada podataka je složen proces koji zahtijeva stručnost, a pogotovo vrijeme. Upravo zato se pojavljuju razni alati i platforme koji pomažu tvrtkama da brzo i jednostavno analiziraju svoje podatke. U ovom poglavlju opisat će se s DataCake platformama čiji je logo na slici 4.1.

DataCake je platforma koja omogućuje jednostavnu analizu podataka te razne načine za prikaz rezultata odnosno podataka. Za prikaz nudi razne grafove, tablice te interaktivne karte. Podatke je moguće učitati iz raznih izvora poput: Excel-a, Google Sheetsa, SQL baza podataka i slično.

Uz sve to DataCake ima mogućnost prepoznavanja nekakvih obrazaca, trendova i slično, što bi moglo biti korisno korisniku. Aplikacija je jednostavna za korištenje i vrlo intuitivna(user friendly). Potrebno se registrirati i možete početi s kreiranjem projekata, uvozom podataka te njihovom obradom.



Slika 4.2. Početna stranica DataCake platforme.

Softver je neovisan o dobavljaču i može se prilagoditi svim zahtjevima kupaca. Datacake je proizvoljno skalabilan, ne zahtijeva znanje programiranja i pokriva najvažnije bežične tehnologije za industriju i tehnologiju senzora s LTE, LoRaWAN i Narrowband-IoT.

Platforma ima jasnu i intuitivnu drag & drop kontrolnu ploču(dashboard). Zahvaljujući unaprijed izgrađenim predlošcima, rad i konfiguracija ne zahtijevaju gotovo nimalo vremena.

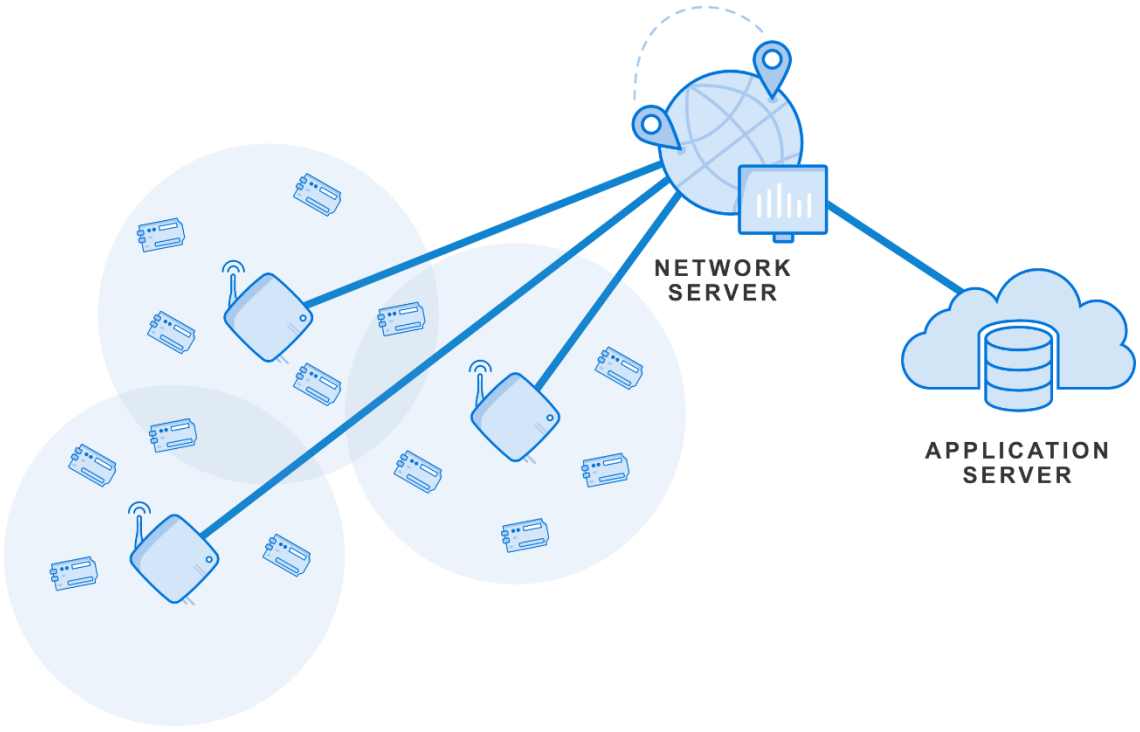
Specifična pravila i pragovi mogu se definirati za sve povezane senzore i IoT uređaje. Ako izmjerena vrijednost prelazi definirani standardni raspon, sustav automatski šalje upozorenje putem SMS-a ili e-pošte. Snimljeni podaci i očitanja mogu se izvesti u CSV datoteke putem izvješća putem e-pošte. CSV datoteka (vrijednosti odvojene zarezima) jest tekstualna datoteka koja ima određeni format koji omogućuje spremanje podataka u tablično strukturiranom formatu.

Nedostatak ove platforme je ograničenost u mogućnostima u usporedbi s drugim aplikacijama za obradu podataka.[9]

5. THE THINGS NETWORK

The Things Network je globalna, otvorena i besplatna mreža za IoT uređaje, bazirana na LoRaWAN tehnologiji. Ona omogućava komunikaciju između uređaja (senzora, aktuatora, itd.) i interneta, bez potrebe za korištenjem tradicionalnih mobilnih mreža ili Wi-Fi-a.

Nastao je kao rezultat zajedničkog napora gradova, tvrtki, sveučilišta i samih pojedinaca koji se bave ovom idejom te su postavili na tisuće gateway uređaja diljem svijeta te ih povezali na TTN, stvarajući svjetsku LoRaWAN mrežu. Prijaviti se može svatko te tako koristit postojeće uređaje gdje je dobra pokrivenost, ili sami prijaviti svoj gateway i tako sudjelovat u proširenju mreže. Preko 21 000 gateway-a imamo danas u svijetu, a taj broj raste iz dana u dan.[36]



Slika 5.1. Način rada The Things Networka[37]

Prvi korak nakon registracije u samu web aplikaciju TTN-a je kreiranje vlastite aplikacije.

Aplikacija služi kao struktura za upravljanjem više senzora koji se u TTN Cloud-u zovu end devices. Prilikom dodavanja samog senzora postoji opcija odabira unaprijed konfiguriranih

senzora u TTN Cloud-u ili ručno dodavanje novih senzora popunjavajući odgovarajuće parametre predstavljene u nastavku.

DevEUI je 64-bitni jedinstveni ID koji krajnjem uređaju dodjeljuje proizvođač, vrijednost DevEUI-a povezana je s hardverom i ne može se mijenjati. DevAddr je 32-bitna adresa koja identificira krajnji uređaj unutar mreže i sva komunikacija nakon pridruživanja mreži obavlja se s njim. Vrijednost DevAddr sastoji se od NwkAddr (adresa krajnjeg uređaja unutar mreže) ispred koje stoji NwkID (mrežni identifikator).

LoRaWAN podatci počinju život kao bežični LoRa radio prijenos od krajnjeg uređaja do gateway uređaja. Zatim se podatci šalju na mrežni poslužitelj u oblaku. Mrežni poslužitelj izvodi funkcije potrebne za rad LoRaWAN-a prije prosljeđivanja podataka na odgovarajući aplikacijski poslužitelj za obradu. Podaci su šifrirani duž cijelog putovanje, najprije ključem mrežne sesije (NwkSKey), a zatim putem ključa sesije aplikacije (AppSKey).

DevAddr i ključevi sesije dodjeljuju se krajnjem uređaju tijekom aktivacije. LoRaWAN podržava dva načina aktivacije krajnjeg uređaja: ABP (eng. Activation By Personalization) i OTAA (eng. Over-The-Air Activation)

6.PRAKTIČNI DIO

6.1. Senzori

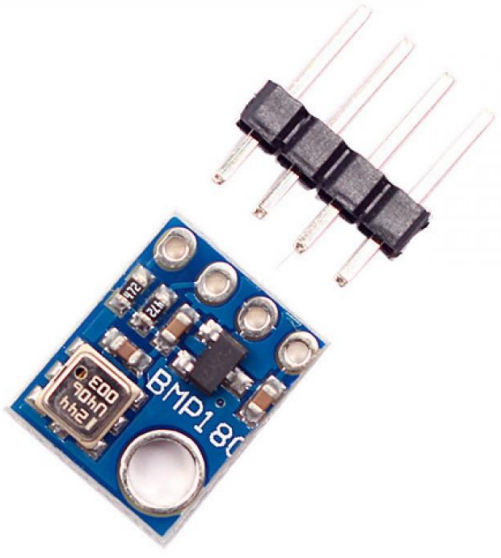
6.1.1 Barometar BMP180

Barometar BMP180 je minijaturni barometar koji omogućava mjerenje atmosferskog tlaka. Dimenzije su mu 3.6mm x 3.8mm

BMP180 proizveden od strane tvrtke Bosch Sensortec. Iako je minijaturan, ovaj senzor ima izuzetno visoku preciznost u mjerenju atmosferskog tlaka. Opremljen je s visokom razlučivošću i brzinom očitavanja, što ga čini idealnim za različite primjene, uključujući meteorologiju, navigaciju, kontrolu visine u letjelicama i slično.

Ima sposobnost mjerenja tlaka u rasponu od 300 do 1100hPa, također ima veliku osjetljivost od 0.02hPa. Princip rada je piezoelektrični. Takav način rada ima 4 otpora koja mjere silu naprezanja silikonske dijafragme, koja se zatim preko Wheatstoneovog mosta pretvara u električni signal.

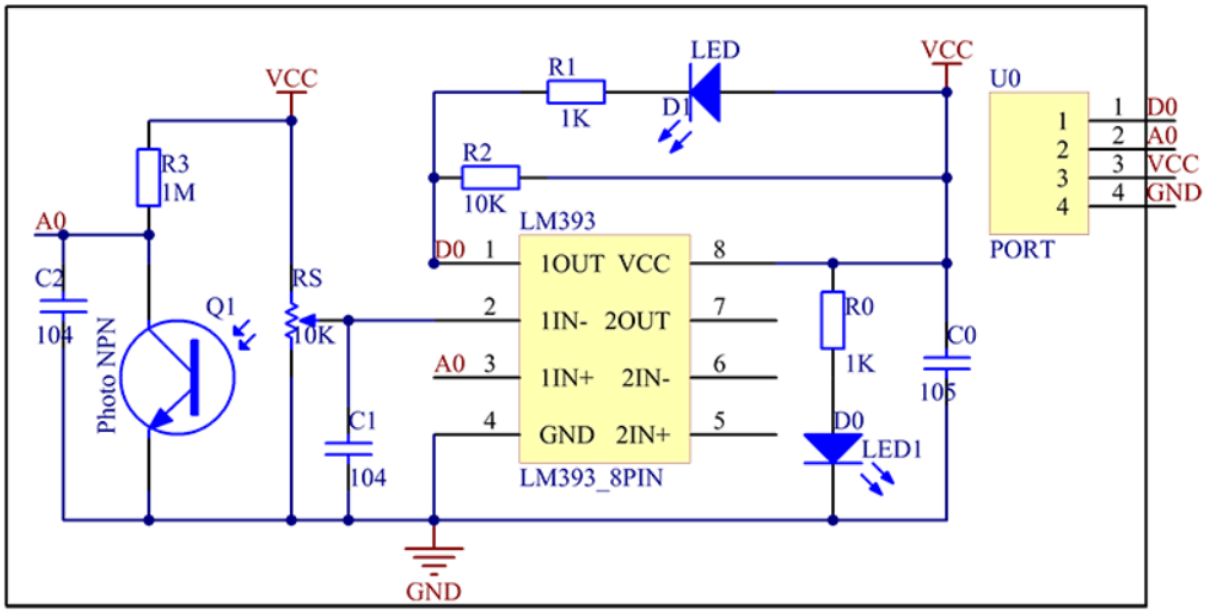
Ovaj senzor koristi I2C sučelje za komunikaciju s mikroračunalom. Ima malu potrošnju energije zbog toga je idealan za baterijski napajane uređaje. Osim tlaka može mjeriti i temperaturu, a podatke šalje u digitalnom formatu. [25]



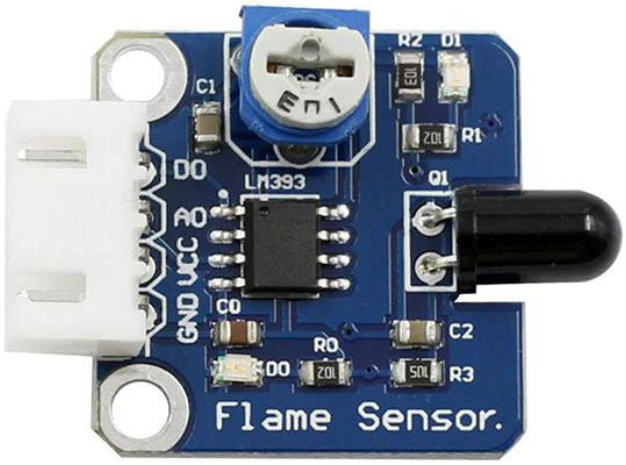
Slika 6.1. Barometar BMP180 [24]

6.1.2. Senzor plamena

Modul detektora plamena sadrži: otpornike, kondenzatore, senzor, potenciometar te komparator LM393. Može detektirati plamen infracrvenog spektra između 700 nm i 1000 nm. Kut „gledanja“ je 60 stupnjeva, također se uz pomoć potenciometra može podešavati osjetljivost senzora.

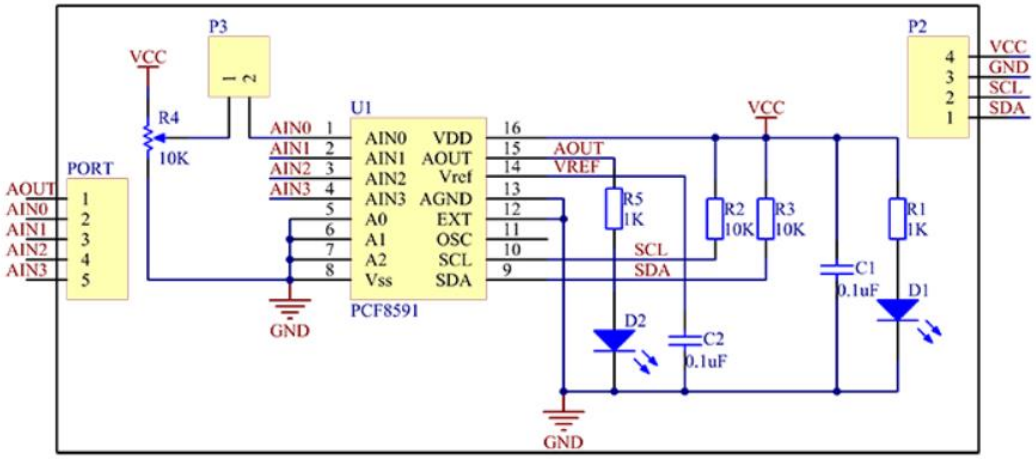


Slika 6.2. Elektronička shema modula detektora plamena.[33]

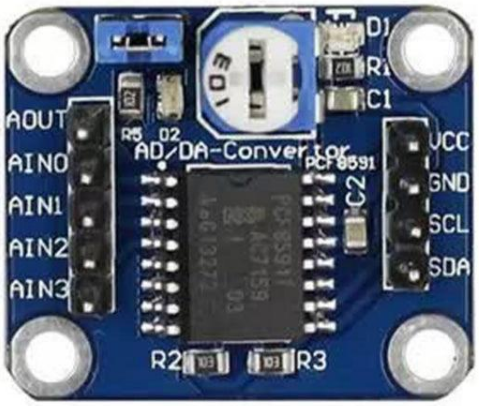


Slika 6.3. Modul za detekciju plamena [27]

Senzor plamena u radu smo spojili preko modula PCF8591 AD/DA convertera . PCF8591 je 8-bitni čip koji ima četiri analogna ulaza, jedan analogni izlaz i I2C sučelje za komunikaciju. Tri pina se koriste za adresiranje (A0, A1, A2), što omogućuje adresiranje do 8 uređaja spojenih na I2C sabirnicu[34].



Slika 6.4. El. shema AD/DA modula PCF8591[35]



Slika 6.5. PCF8591 AD/DA converter [29]

6.1.3 Velleman VMA430 - GPS / GLONASS u-Blox NEO-7M UART modul s antenom.

Slika 6.6. GPS modul Velleman VMA430 koji se koristi u radu. [26]

Serija NEO 7 čipa koji se koristi na ovom modulu podržava GNSS sustave: GLONAS(Rusija), QZSS(Japan), i SBAS(EU). Prednost modula je vrlo velika osjetljivost i preciznost, uz vrlo malu potrošnju energije. Verzija NEO-7M ima lošije performanse, ali je dosta jeftinija od NEO- 7N verzije. NEO-7M modul je također izuzetno pouzdan.

Ugrađeni mehanizmi za samo dijagnostiku omogućavaju automatsko otkrivanje i ispravljanje grešaka, što smanjuje mogućnost prekida u radu.

Model	Type								Supply			Interfaces				Features												Grade		
	GPS / QZSS	GLONASS	Galileo	BeiDou	Timing	Dead Reckoning	Precise Point Positioning	Raw Data	1.65 V – 3.6 V	2.7 V – 3.6 V	Lowest power (DC/DC)	UART	USB	SPI	DDC (I²C compliant)	Programmable (Flash)	Data logging	Additional SAW	Additional LNA	RTC crystal	Internal oscillator	Active antenna / LNA supply	Active antenna / LNA control	Antenna short circuit detection / protection pin	Antenna open circuit detection pin	Frequency output	Standard	Professional	Automotive
NEO-7N	•	•								•	•	•	•	•	•	•	•	•	•	•	T	o	•						
NEO-7M	•	•							•		•	•	•	•	•					•	C	o							

o = Optional, not activated per default or requires external components

C = Crystal / T = TCXO

Slika 6.7. Specifikacije različitih verzija modula.

6.1.4. RFM 92/95 Breakout V2.1

Kod oznake RMF 92/95 je modul proizveden od strane tvrtke HopeRF. Modul koristi LoRa tehnologiju te omogućuje mnogo veće udaljenosti prijenosa odnosno veći domet uz malu potrošnju energije kao i ostale prednosti LoRa modulacije. Postoji više verzija koje se razlikuju u parametrima.

Tablica 6.1. Karakteristike RFM modula [30]

Part Number	Frequency Range	Spreading Factor	Bandwidth	Effective Bitrate	Est. Sensitivity
RFM95W	868/915 MHz	6 - 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm
RFM97W	868/915 MHz	6 - 9	7.8 - 500 kHz	0.11 - 37.5 kbps	-111 to -139 dBm
RFM96W/RFM98W	433/470MHz	6- 12	7.8 - 500 kHz	.018 - 37.5 kbps	-111 to -148 dBm

Slika 6.8. Modul RF92/95 [31]

RFM92/95 moduli podržavaju rad na frekvencijama u ISM (Industrial, Scientific, Medical) opsegu, uključujući 433MHz, 868MHz i 915MHz. Ovi moduli podržavaju prilagodljivu brzinu prijenosa podataka, što omogućava podešavanje brzine u skladu sa zahtjevima specifične aplikacije. Tipične brzine prijenosa podataka kreću se od nekoliko desetina kbit/s do nekoliko stotina kbit/s, u ovisnosti od frekvencijskog opsega i drugih faktora. RFM92/95 moduli se povezuju s mikrokontrolerima ili drugim uređajima putem serijske komunikacije (UART) ili SPI interface-a.

Tablica 6.2. Raspored pinova RFM modula

Number	Name	Type	Description Description Stand Alone Mode
1	GND	-	Ground
2	MISO	I	SPI Data output
3	MOSI	O	SPI Data input
4	SCK	I	SPI Clock input
5	NSS	I	SPI Chip select input
6	RESET	I/O	Reset trigger input
7	DIO5	I/O	Digital I/O, software configured
8	GND	-	Ground
9	ANT	-	RF signal output/input.
10	GND	-	Ground
11	DIO3	I/O	Digital I/O, software configured
12	DIO4	I/O	Digital I/O, software configured
13	3.3V	-	Supply voltage
14	DIO0	I/O	Digital I/O, software configured
15	DIO1	I/O	Digital I/O, software configured
16	DIO2	I/O	Digital I/O, software configured

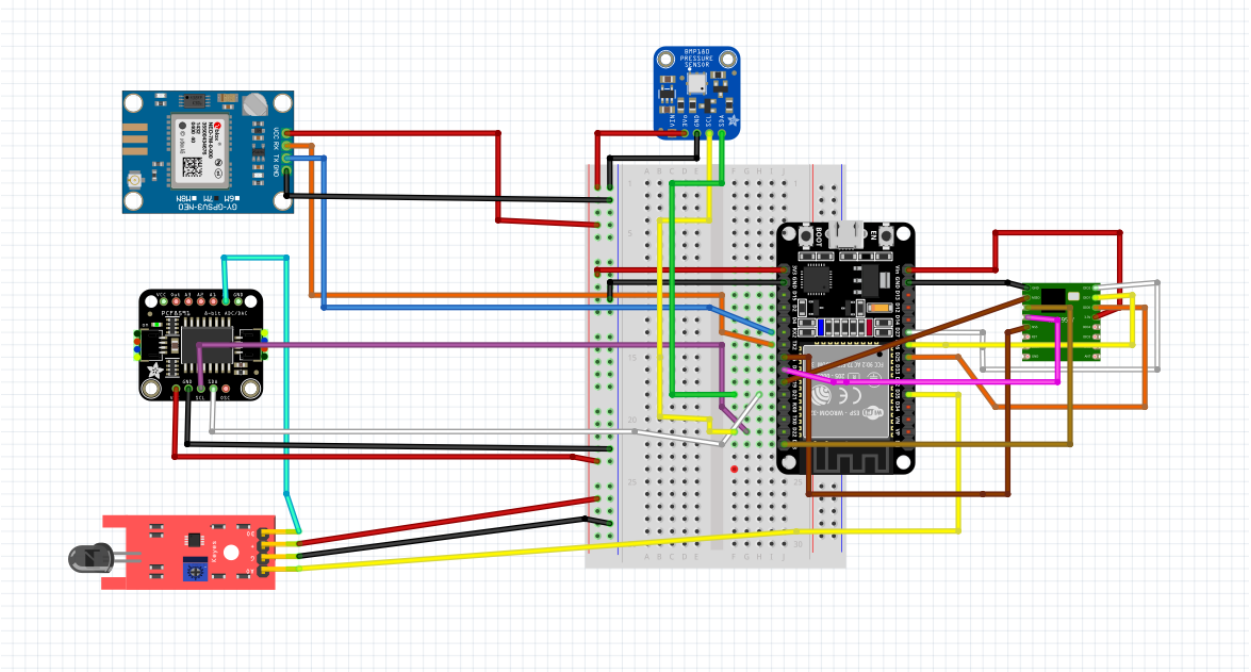
6.2. Shema spajanja

Senzori su spojeni na napajanje na izlaz 3.3V na ESP32 modulu, također i na GND izlaz. BMP180 spojen je na pinove D21 i D22, te komunicira I2C protokolom preko SDA (serial dana) i SCL (serial clock) linija. Ulazi D21 i D22 su predviđeni za I2C komunikaciju. I2C sabirnica koristi dvije žice (SDA i SCL) za prijenos podataka između sabirnice i uređaja, serijsku komunikaciju između mikrokontrolera i vanjskih uređaja ili dvosmjerni prijenos podataka između master i slave uređaja. I2C je višestruka glavna sabirnica, tako da bilo koji uređaj može raditi kao glavni i upravljati sabirnicom. Svaki uređaj u sabirnici ima jedinstvenu adresu, a
mogu raditi kao odašiljači ili prijamnici.
GPS modul koristi serijsku komunikaciju preko RX i TX sabirnica, te je spojen na pinove TXD2 i RXD2. Brzina prijenosa (baud rate) je 9600, koristi 8 bitova podataka te 1 stop bit, također nema pariteta ni flow control-a.

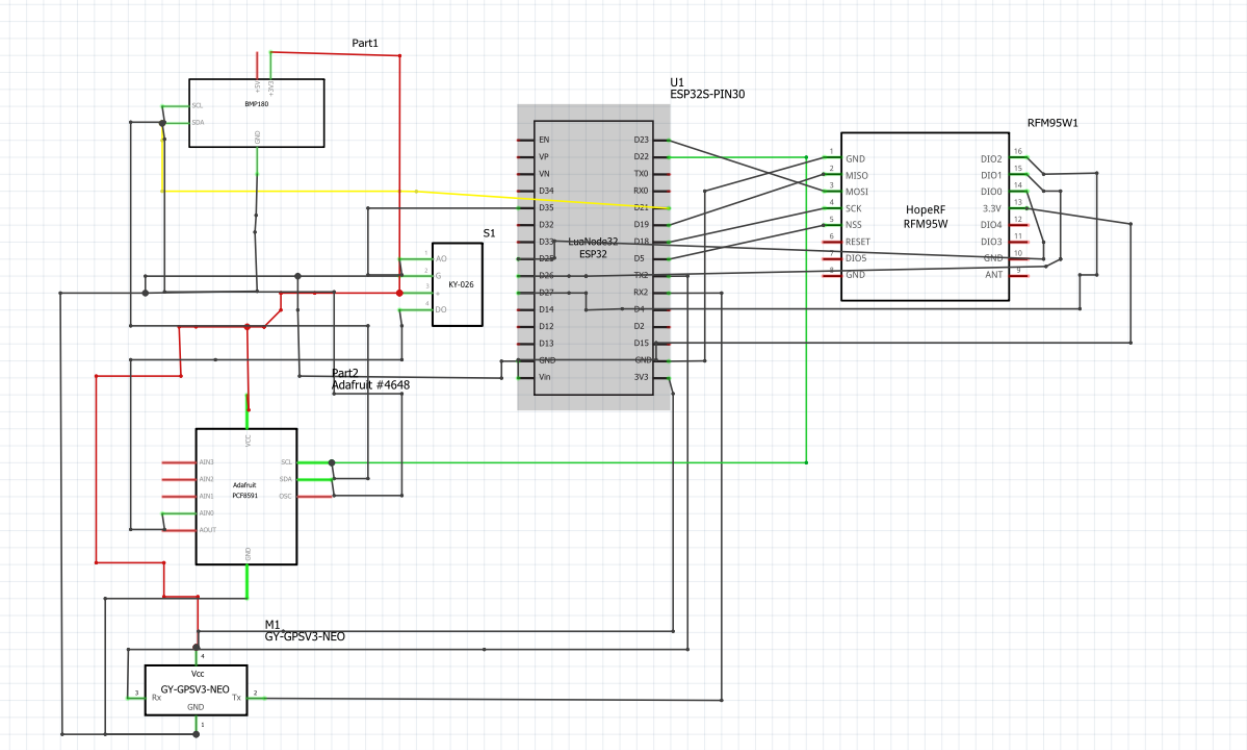
Detektor plamena ima dva izlaza jedan digitalni (D0) i jedan analogni (A0). Analogni izlaz je spojen na ulaz D35 na ESP32 pločici, dok se digitalni izlaz spaja na AD/DA converter. Modul PCF8591 koristi I2C komunikaciju preko SDA i SCK sabirnica, također je spojen na pinove D21 i D22 ESP32 pločice.

RFM 92/95 modul koristi SPI za komunikaciju s ESP32 mikrokontroler-om preko MISO, MOSI, SCK i NSS sabirnica. MISO (Master In Slave Out) i MOSI (Master Out Slave In) služe za prijenos podataka između master i slave uređaja. Komunikaciju inicira master uređaj tako što spusti NSS signal na logičku nulu, a zatim slijedi prijenos podataka preko MOSI linije. Ako master uređaj ne očekuje odgovor od slave uređaja onda će odmah nakon poslanog podatka signal NSS vratiti na logičku jedinicu, a ako slave treba odgovoriti on će odgovoriti preko MISO linije pa će nakon toga master uređaj CS postaviti na jedinicu. Prijenos podataka preko MOSI i MISO linija je sinkronizirana s taktom SCK. [38]

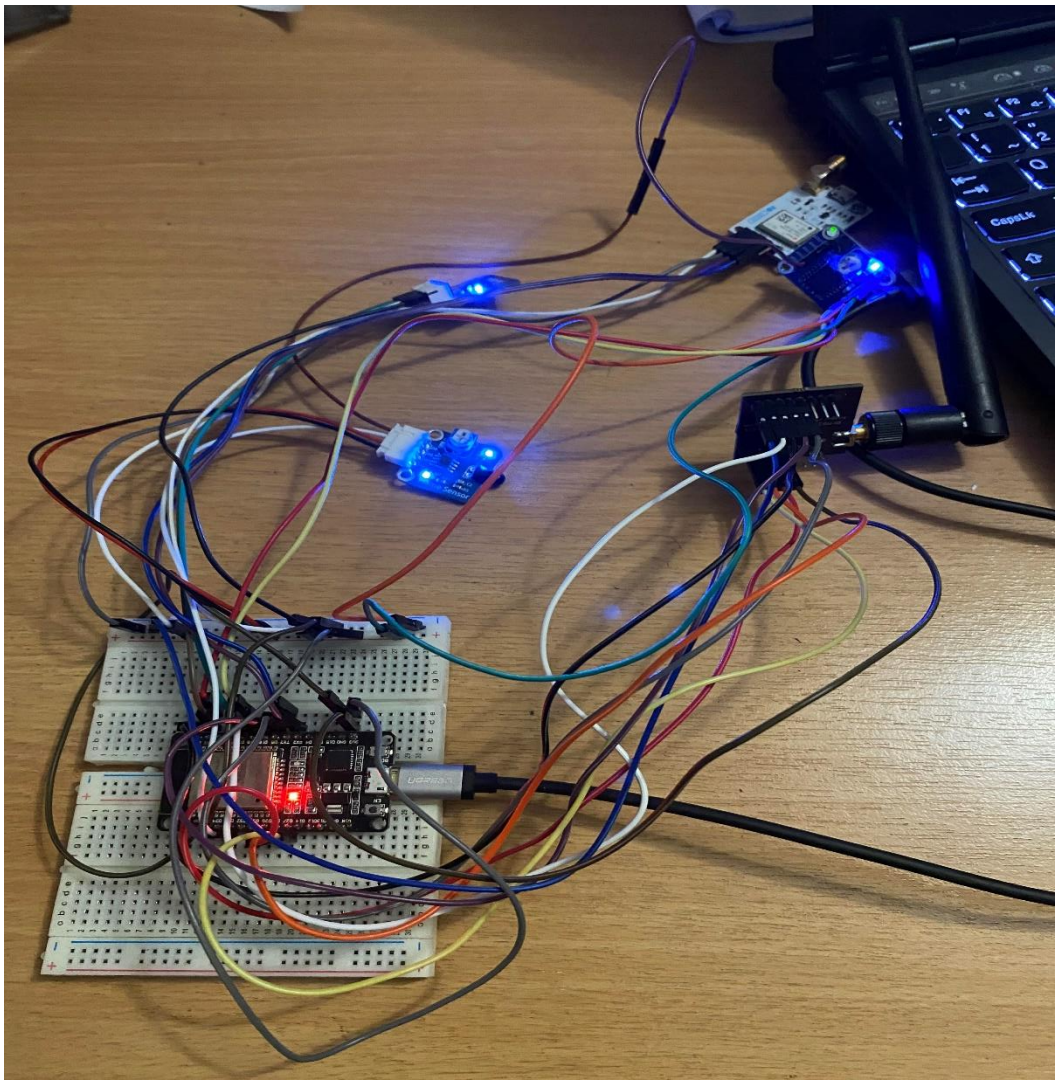
Digitalni izlazi DIO0, DIO1 i DIO2 RFM92/95 modula su spojeni na pinove D25, D26 i D27.



Slika 6.9. Board shema sklopa napravljena u fritzing programu.



Slika 6.10. Shematic shema sklopa napravljena u fritzing programu.



Slika 6.11. prikazuje fizički izgled sklopa

6.3 Programski Kod

Programski kod koji je korišten u radi napisan je u Arduino IDE-u, C++ programskim jezikom. Da bi se mogli skupljati podatci sa senzora potrebno je prvo pozvati potrebne Arduino libary-e. u ovom programu su korištene biblioteke: lmic.h, hal/hal.h, WiFi.h, esp_sleep.h, Wire.h, SPI.h, Adafruit_BMP085.h, TinyGPS++.h, HardwareSerial.h. Nakon toga definirani su pinovi i njihove funkicije.

<lmic.h> uključuje LMIC biblioteku za LoRaWAN komunikaciju.

<hal/hal.h> uključuje HAL (Hardware Abstraction Layer) za LoRa module.

<WiFi.h> uključuje biblioteku za Wi-Fi funkcionalnost na ESP32 (nije korištena u ovom kodu, ali se koristi za isključivanje Wi-Fi-a).

<esp_sleep.h> uključuje biblioteku za upravljanje načinom spavanja ESP32.

<Wire.h> uključuje biblioteku za I2C komunikaciju.

<SPI.h>: uključuje biblioteku za SPI komunikaciju.

<Adafruit_BMP085.h> uključuje biblioteku za BMP180 senzor.

<TinyGPS++.h> uključuje biblioteku za GPS modul.

<HardwareSerial.h> omogućuje korištenje dodatnih serijskih portova na ESP32.

Kod sadrži dvije opcije za način aktivacije, odnosno OTAA I APB. Korištena je OTAA aktivacija pa su upisani indetifikatori DEVEUI, APPEUI, i APPKEY koje generira The Things Network. Ovi identifikatori i ključevi omogućuju uređaju da se autentificira i poveže s mrežom.

Funkcija `preparePayload()` prikuplja podatke sa senzora, uključujući temperaturu, tlak, GPS lokaciju i podatke s flame detektor. Podaci se enkodiraju u niz bajtova i spremaju u payload za slanje putem LoRaWAN.

```
void preparePayload() {
  float temperature = bmp.readTemperature();
  int32_t pressure = bmp.readPressure();

  int16_t temperature_scaled = temperature * 100;
  mydata[0] = (temperature_scaled >> 8) & 0xFF;
  mydata[1] = temperature_scaled & 0xFF;
  mydata[2] = (pressure >> 24) & 0xFF;
  mydata[3] = (pressure >> 16) & 0xFF;
  mydata[4] = (pressure >> 8) & 0xFF;
  mydata[5] = pressure & 0xFF;

  if (gps.location.isValid()) {
    float latitude = gps.location.lat();
    float longitude = gps.location.lng();
```

```
    int32_t lat_scaled = latitude * 1000000;
    int32_t lng_scaled = longitude * 1000000;
    mydata[6] = (lat_scaled >> 24) & 0xFF;
    mydata[7] = (lat_scaled >> 16) & 0xFF;
    mydata[8] = (lat_scaled >> 8) & 0xFF;
    mydata[9] = lat_scaled & 0xFF;
    mydata[10] = (lng_scaled >> 24) & 0xFF;
    mydata[11] = (lng_scaled >> 16) & 0xFF;
    mydata[12] = (lng_scaled >> 8) & 0xFF;
    mydata[13] = lng_scaled & 0xFF;
  }

  int flame_value = readPCF8591();
  mydata[14] = (flame_value >> 8) & 0xFF;
  mydata[15] = flame_value & 0xFF;
}
```

Gore je priložen dio koda koji služi za pripremanje tj prikupljanje podataka sa senzora i formatira ih u niz bajtova spremnih za slanje.

`onEvent()` služi za obradu različitih LoRaWAN događaja kao što su uspješno slanje podataka ili primanje paketa.

Funkcija `do_send()` upravlja slanjem podataka putem LoRaWAN mreže, a `onEvent()` upravlja različitim događajima vezanim za LoRaWAN, kao što su uspješno slanje podataka i primanje potvrde. Funkcija `setup()` inicijalizira serijsku komunikaciju, GPS modula, BMP180 senzora i I2C modul. Također postavlja LoRaWAN sesiju i konfigurira GPIO pinove.

Funkcija `loop()` kontinuirano prikuplja i provjerava GPS podatke i obrađuje LoRaWAN događaje.

```
#include <lmic.h>
#include <hal/hal.h>
#include <WiFi.h>
```

```
#include <esp_sleep.h>
#include <Wire.h>
#include <SPI.h>
#include <Adafruit_BMP085.h>   // BMP180 library
#include <TinyGPS++.h>         // GPS library
#include <HardwareSerial.h>    // For Serial2 (GPS communication)

// Pin definitions
#define FLAME_ANALOG_PIN 35   // Analog input pin for Flame Detector
#define PCF8591_ADDRESS 0x48  // I2C address for the PCF8591 module
#define RX2_PIN 16            // RX2 pin for GPS
#define TX2_PIN 17            // TX2 pin for GPS

Adafruit_BMP085 bmp;          // BMP180 instance
TinyGPSPlus gps;              // TinyGPS++ instance for GPS
HardwareSerial GPS_Serial(2); // Serial2 for GPS communication

#define USE_OTAA 1

#ifdef USE_OTAA
static PROGMEM u1_t DEVEUI[8]  = { 0xCB, 0xA3, 0x06, 0xD0, 0x7E, 0xD5, 0xB3,
0x70 }; // Device EUI
static PROGMEM u1_t APPEUI[8]  = { 0x78, 0xC2, 0x01, 0xD0, 0x7E, 0xD5, 0xB3,
0x70}; // Application EUI
static PROGMEM u1_t APPKEY[16] = { 0x2B, 0x49, 0x5C, 0xE8, 0x12, 0x84, 0x9C,
0xD4, 0xD3, 0x75, 0xFC, 0x41, 0x5E, 0x10, 0xA4, 0x27 }; // App Key
#else
static PROGMEM u1_t NWKSKEY[16] = { 0x8C, 0xF1, 0xA8, 0x74, 0xD4, 0x85, 0xAC,
0xCB, 0x49, 0x4F, 0x04, 0x6C, 0x0F, 0x0B, 0x3E, 0x5C }; // Network session
key
static PROGMEM u1_t APPSKEY[16] = { 0x0F, 0xDC, 0xAC, 0x09, 0xA2, 0xAB, 0xA3,
0xA3, 0x67, 0xFF, 0x58, 0xBC, 0x7E, 0x7E, 0xCC, 0x47 }; // Application
session key
static PROGMEM u4_t DEVADDR = 0x20114BC; // LoRaWAN DevAddr
#endif

#undef BUILTIN_LED
#define BUILTIN_LED 2
```

```
char s[32]; // Buffer for Serial output
static uint8_t mydata[16];  // Payload array
static osjob_t sendjob;

RTC_NOINIT_ATTR int RTCseqnoUp, RTCseqnoDn;
#ifdef USE_OTAA
RTC_NOINIT_ATTR u4_t otaaDevAddr;
RTC_NOINIT_ATTR u1_t otaaNetwKey[16];
RTC_NOINIT_ATTR u1_t otaaApRtKey[16];
#endif

const unsigned TX_INTERVAL = 30;

const lmic_pinmap lmic_pins = {
  .nss = 5,  // NSS pin for LoRa module
  .rxtx = LMIC_UNUSED_PIN,
  .rst = LMIC_UNUSED_PIN,
  .dio = {25, 26, 27}  // DIO0, DIO1, DIO2
};

#ifdef USE_OTAA
void os_getDevEui(u1_t* buf) { memcpy_P(buf, DEVEUI, 8); }
void os_getArtEui(u1_t* buf) { memcpy_P(buf, APPEUI, 8); }
void os_getDevKey(u1_t* buf) { memcpy_P(buf, APPKEY, 16); }
#else
void os_getArtEui(u1_t* buf) { }
void os_getDevEui(u1_t* buf) { }
void os_getDevKey(u1_t* buf) { }
#endif

void storeFrameCounters() {
  RTCseqnoUp = LMIC.seqnoUp;
  RTCseqnoDn = LMIC.seqnoDn;
  sprintf(s, "Counters stored as %d/%d", LMIC.seqnoUp, LMIC.seqnoDn);
  Serial.println(s);
}
```

```
void restoreFrameCounters() {
  LMIC.seqnoUp = RTCseqnoUp;
  LMIC.seqnoDn = RTCseqnoDn;
  sprintf(s, "Restored counters as %d/%d", LMIC.seqnoUp, LMIC.seqnoDn);
  Serial.println(s);
}

void setOrRestorePersistentCounters() {
  esp_reset_reason_t reason = esp_reset_reason();
  if ((reason != ESP_RST_DEEPSLEEP) && (reason != ESP_RST_SW)) {
    Serial.println(F("Counters both set to 0"));
    LMIC.seqnoUp = 0;
    LMIC.seqnoDn = 0;
  } else {
    restoreFrameCounters();
  }
}

int readPCF8591() {
  Wire.beginTransmission(PCF8591_ADDRESS);
  Wire.write(0x00); // Command to read the analog input
  Wire.endTransmission();
  Wire.requestFrom(PCF8591_ADDRESS, 2);
  Wire.read(); // Dummy read
  return Wire.read(); // Actual value
}

void collectGPSData() {
   // Continuously collect and process GPS data without printing raw NMEA
sentences
  while (GPS_Serial.available() > 0) {
    char c = GPS_Serial.read();  // Read a character from GPS
    gps.encode(c);  // Feed GPS data to TinyGPS++
  }
}

bool isGPSValid() {
  return gps.location.isValid(); // Check if GPS data is valid
```

```
}

void preparePayload() {
  // Read temperature and pressure from BMP180
  float temperature = bmp.readTemperature();
  int32_t pressure = bmp.readPressure();

  Serial.print(F("Temperature = "));
  Serial.print(temperature);
  Serial.println(" *C");

  Serial.print(F("Pressure = "));
  Serial.print(pressure);
  Serial.println(" Pa");

  int16_t temperature_scaled = temperature * 100;
  mydata[0] = (temperature_scaled >> 8) & 0xFF;
  mydata[1] = temperature_scaled & 0xFF;
  mydata[2] = (pressure >> 24) & 0xFF;
  mydata[3] = (pressure >> 16) & 0xFF;
  mydata[4] = (pressure >> 8) & 0xFF;
  mydata[5] = pressure & 0xFF;

  // Check if GPS data is valid
  if (gps.location.isValid()) {
    float latitude = gps.location.lat();
    float longitude = gps.location.lng();
    int32_t lat = latitude * 10000;  // Scale to avoid float usage
    int32_t lng = longitude * 10000; // Scale to avoid float usage

    Serial.print(F("Valid GPS data acquired! Latitude: "));
    Serial.println(latitude, 6);
    Serial.print(F("Longitude: "));
    Serial.println(longitude, 6);

    // Add GPS data to payload
    mydata[6] = (lat >> 24) & 0xFF;
    mydata[7] = (lat >> 16) & 0xFF;
```

```
    mydata[8] = (lat >> 8) & 0xFF;
    mydata[9] = lat & 0xFF;
    mydata[10] = (lng >> 24) & 0xFF;
    mydata[11] = (lng >> 16) & 0xFF;
    mydata[12] = (lng >> 8) & 0xFF;
    mydata[13] = lng & 0xFF;
  } else {
    Serial.println(F("Waiting for valid GPS data..."));
  }

  // Read the Flame Detector signals
  int flame_analog = analogRead(FLAME_ANALOG_PIN);
  int flame_digital = readPCF8591();

  Serial.print(F("Flame Analog Value: "));
  Serial.println(flame_analog);
  Serial.print(F("Flame Digital Value: "));
  Serial.println(flame_digital);

  // Add flame sensor data to payload
  mydata[14] = flame_analog & 0xFF;
  mydata[15] = flame_digital & 0xFF;
}

void do_send(osjob_t* j) {
  if (LMIC.opmode & OP_TXRXPEND) {
    Serial.println(F("OP_TXRXPEND, not sending"));
  } else {
    preparePayload();
    LMIC_setTxData2(1, mydata, sizeof(mydata), 0);
    Serial.println(F("Packet queued"));
    digitalWrite(BUILTIN_LED, HIGH);
  }
}

void onEvent(ev_t ev) {
  switch (ev) {
    case EV_SCAN_TIMEOUT:
```

```
      Serial.println(F("EV_SCAN_TIMEOUT"));
      break;
    case EV_BEACON_FOUND:
      Serial.println(F("EV_BEACON_FOUND"));
      break;
    case EV_BEACON_MISSED:
      Serial.println(F("EV_BEACON_MISSED"));
      break;
    case EV_BEACON_TRACKED:
      Serial.println(F("EV_BEACON_TRACKED"));
      break;
    case EV_JOINING:
      Serial.println(F("EV_JOINING"));
      break;
    case EV_JOINED:
#ifdef USE_OTAA
      otaaDevAddr = LMIC.devaddr;
      memcpy_P(otaaNetwKey, LMIC.nwkKey, 16);
      memcpy_P(otaaApRtKey, LMIC.artKey, 16);
      sprintf(s, "got devaddr = 0x%X", LMIC.devaddr);
      Serial.println(s);
#endif
      LMIC_setLinkCheckMode(0);
      LMIC.dn2Dr = DR_SF9;
      break;
    case EV_JOIN_FAILED:
      Serial.println(F("EV_JOIN_FAILED"));
      break;
    case EV_TXCOMPLETE:
      Serial.println(F("EV_TXCOMPLETE (includes waiting for RX windows)"));
      digitalWrite(BUILTIN_LED, LOW);
      if (LMIC.txrxFlags & TXRX_ACK) {
        Serial.println(F("Received Ack"));
      }
      if (LMIC.dataLen) {
        sprintf(s, "Received %i bytes of payload", LMIC.dataLen);
        Serial.println(s);
        sprintf(s, "RSSI %d SNR %.1d", LMIC.rssi, LMIC.snr);
```

```
        Serial.println(s);
      }
      storeFrameCounters();
      Serial.println("Good night...");
      esp_sleep_enable_timer_wakeup(TX_INTERVAL * 1000000);
      esp_deep_sleep_start();
      break;
    default:
      Serial.print(F("Unknown event: "));
      Serial.println(ev);
      break;
  }
}

void setup() {
  Serial.begin(115200);
   GPS_Serial.begin(9600, SERIAL_8N1, RX2_PIN, TX2_PIN);  // Initialize GPS
Serial2
  Serial.println(F("ESP32 LoRaWAN with BMP180, Flame Detector, and GPS"));

  WiFi.mode(WIFI_OFF);
  btStop();

  if (!bmp.begin()) {
    Serial.println(F("Could not find a valid BMP180 sensor, check wiring!"));
    while (1) delay(10);
  }

  Wire.begin(21, 22);

  os_init();
  LMIC_reset();
  LMIC_setClockError(MAX_CLOCK_ERROR * 1 / 100);

#ifdef USE_OTAA
  esp_reset_reason_t reason = esp_reset_reason();
  if ((reason == ESP_RST_DEEPSLEEP) || (reason == ESP_RST_SW)) {
    LMIC_setSession(0x1, otaaDevAddr, otaaNetwKey, otaaApRtKey);
```

```
  }
#else
  LMIC_setSession(0x1, DEVADDR, NWKSKEY, APPSKEY);
  LMIC.dn2Dr = DR_SF9;
  LMIC_setLinkCheckMode(0);
#endif

  setOrRestorePersistentCounters();
  LMIC_setDrTxpow(DR_SF7, 14);

  do_send(&sendjob);

  pinMode(BUILTIN_LED, OUTPUT);
  digitalWrite(BUILTIN_LED, HIGH);
  delay(1000);
  digitalWrite(BUILTIN_LED, LOW);
}

void loop() {
  collectGPSData();  // Continuously collect GPS data
  os_runloop_once();
}
```

Također ove kodirane podatke je potrebno dekodirati unutra The Things Networka i Datacake platforme. Decoder u TTN-u osim što dekodira donosi odluku u ovisnosti o analognoj vrijednosti s detektora plamena. Ako je vrijednost veća od 1500 detektiran je plamen, ako je ispod te vrijednosti plamena nema. Važno je naglasiti da analogna vrijednost kod dekodera u TTN-u nije ista ko u serial monitoru radi korištenja 8 bitova za kodiranje. Kod koji se koristi kao Payload decoder unutar The Things Networka:

```
function decodeUplink(input) {
  var bytes = input.bytes;

  // Decode temperature (signed 16-bit, big-endian)
  var temperatureRaw = (bytes[0] << 8) | bytes[1];
```

```
  var temperature = temperatureRaw / 100;  // Scale by 100

  // Decode pressure (32-bit unsigned, big-endian)
  var pressure = ((bytes[2] << 24) | (bytes[3] << 16) | (bytes[4] << 8) |
bytes[5]);

  // Decode latitude (32-bit signed, big-endian, scaled by 10,000)
  var latRaw = (bytes[6] << 24) | (bytes[7] << 16) | (bytes[8] << 8) |
bytes[9];
  var latitude = latRaw / 10000;  // Scale by 10,000

  // Decode longitude (32-bit signed, big-endian, scaled by 10,000)
  var lonRaw = (bytes[10] << 24) | (bytes[11] << 16) | (bytes[12] << 8) |
bytes[13];
  var longitude = lonRaw / 10000;  // Scale by 10,000

  // Decode flame sensor analog value (8-bit unsigned)
  var flameAnalog = bytes[14];

  // Decode flame sensor digital value (8-bit unsigned)
  var flameDigital = bytes[15];

  // Determine flame status
  var flameStatus = flameAnalog > 1500 ? "Plamen detektiran" : "Nema
plamena";

  return {
    data: {
      temperature: temperature,         // Temperature in °C
      pressure: pressure,               // Pressure in Pascals
      latitude: latitude,               // Latitude
      longitude: longitude,             // Longitude
      flame_analog: flameAnalog,        // Flame analog value
      flame_digital: flameDigital,      // Flame digital value
      flame_status: flameStatus         // Flame detection status
    }
  };
}
```

Slijedi kod koji se koristi kao payload decoder unutar Datacake platforme:

```
function Decoder(payload, port) {

    var decoded = {};

    // Decode temperature (signed 16-bit, big-endian)
    var temperatureRaw = (payload[0] << 8) | payload[1];
    decoded.temperature = temperatureRaw / 100;  // Scale by 100

    // Decode pressure (32-bit unsigned, big-endian)
    decoded.pressure = ((payload[2] << 24) | (payload[3] << 16) | (payload[4]
<< 8) | payload[5]);

    // Decode latitude (32-bit signed, big-endian, scaled by 10,000)
    var latRaw = (payload[6] << 24) | (payload[7] << 16) | (payload[8] << 8)
| payload[9];
    decoded.latitude = latRaw / 10000;  // Scale by 10,000

    // Decode longitude (32-bit signed, big-endian, scaled by 10,000)
    var lonRaw = (payload[10] << 24) | (payload[11] << 16) | (payload[12] <<
8) | payload[13];
    decoded.longitude = lonRaw / 10000;  // Scale by 10,000

    // Decode flame sensor analog value (8-bit unsigned)
    decoded.flame_analog = payload[14];

    // Decode flame sensor digital value (8-bit unsigned)
    decoded.flame_digital = payload[15];

    // Determine flame status
    decoded.flame_status   =   (decoded.flame_analog   >   1500)   ?   "Plamen
detektiran" : "Nema plamena";

    // Formatted location
```

```
    decoded.LOCATION = "(" + decoded.latitude + "," + decoded.longitude +
")";

    // Extract Gateway Information
    try {
        // Assuming normalizedPayload is accessible and contains the gateway
information
        decoded.LORA_RSSI       =       (!!normalizedPayload.gateways       &&
!!normalizedPayload.gateways[0] && normalizedPayload.gateways[0].rssi) || 0;
        decoded.LORA_SNR        =       (!!normalizedPayload.gateways       &&
!!normalizedPayload.gateways[0] && normalizedPayload.gateways[0].snr) || 0;
        decoded.LORA_DATARATE = normalizedPayload.data_rate;
    } catch (e) {
        // Logs can be seen on Debug-Tab of Device-View
        console.log("Error decoding LoRa metadata");
        console.log(JSON.stringify(e));
    }

    return decoded;
}
```

6.4. Rezultati

Cilj ovog rada je bio skupiti i poslati podatke sa senzora preko LoRaWAN-a na Datacake platformu te ih tamo prikazati. Prikazat ćemo rezultate u nekoliko koraka. Na slici 6.10. se vidi kako podatci izgledaju na Serial monitoru u Arduino IDE-u. Na početku se vide podatci o samom uređaju, podatci su informativni npr. Vrijeme kompajliranja, adrese u programa i podataka, mod rada i slično. Zatim slijede podatci koji su očitani na senzorima te se šalju, a to su: Temperatura, tlak(pressure), geografska širina i dužina (latitude i longitude), analogna i digitalna vrijednost s detektora plamena. I vide se povrate informacije o tijeku slanja podataka LoRaWAN-om.

„Packet queued“ znači da je paket podataka poslan u red za slanje, odnosno zapakiran je i spreman za slanje.

“EV_JOINING” je događaj kada se uređaj pokušava spojiti se s mrežom, odnosno označava da je uređaj u fazi prijavljivanja u mrežu.

Got devaddr = 0x260B6F2F je adresa uređaja koji šalje poruku

“EV_TXCOMPLETE” (includes waiting for RX windows)” poruka koja označava da je slanje paketa dovršenoj da uređaj čeka odgovor u RX pinu (receiving window).

“Counters stored as /1” označava broj prijenosa koji je odrađen u ovom slučaju konkretno jedan

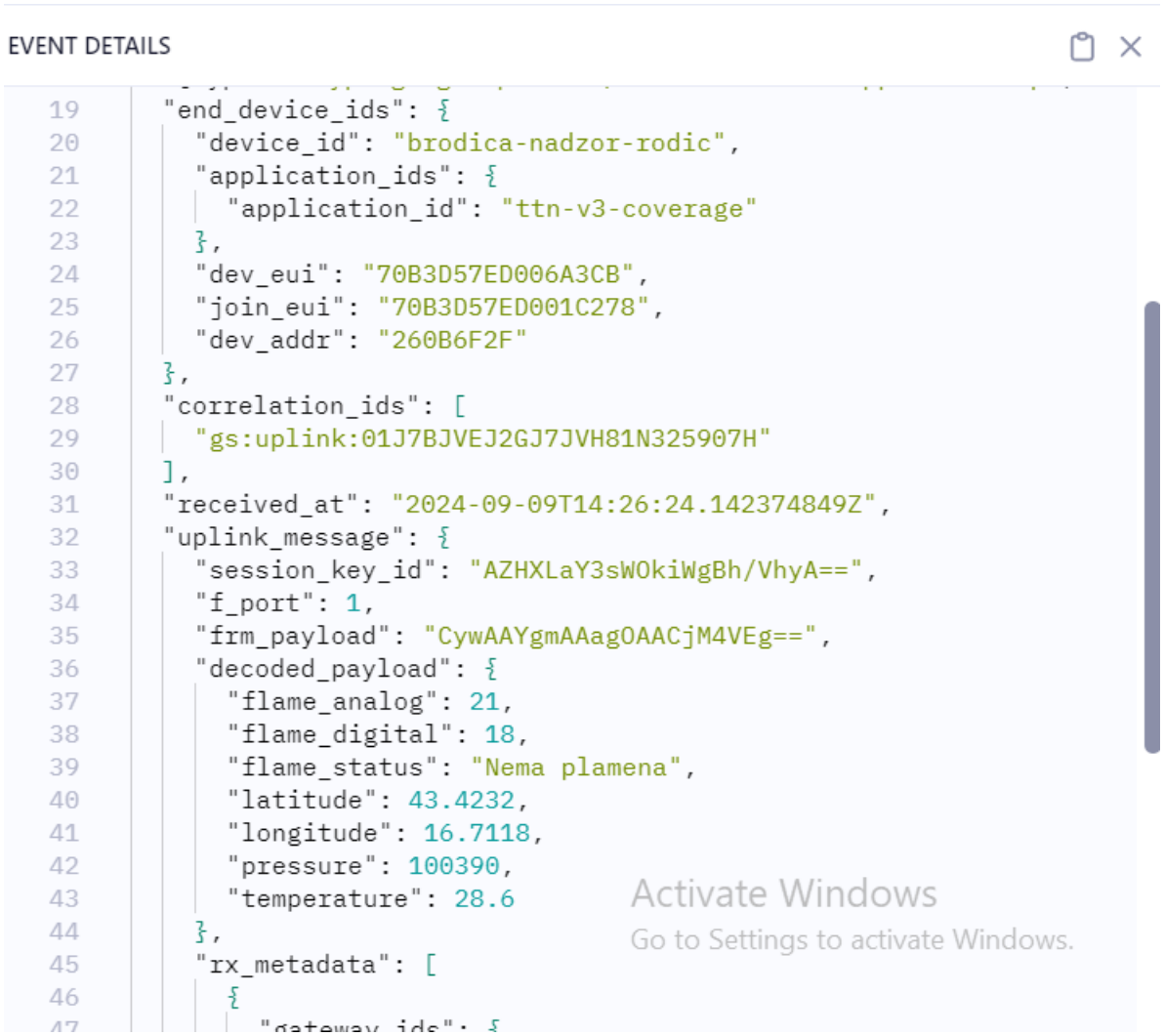
“Good night…” na kraju poruka koja označava kraj operacija za taj ciklus, te se uređaj prebacuje u mod kad se nema slanje tj mirovanje.

```
ets Jun  8 2016 00:22:57

rst:0x1 (POWERON_RESET),boot:0x13 (SPI_FAST_FLASH_BOOT)
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:1
load:0x3fff0030,len:4604
ho 0 tail 12 room 4
load:0x40078000,len:15488
load:0x40080400,len:4
load:0x40080404,len:3180
entry 0x400805b8
ESP32 LoRaWAN with BMP180, Flame Detector, and GPS
Counters both set to 0
Temperature = 28.60 *C
Pressure = 100390 Pa
Using fixed GPS data: Latitude: 43.423199
Longitude: 16.711800
Flame Analog Value: 3093
Flame Digital Value: 18
Packet queued
EV_JOINING
Unknown event: 17
got devaddr = 0x260B6F2F
Unknown event: 17
EV_TXCOMPLETE (includes waiting for RX windows)
Counters stored as 1/1
Good night...
```

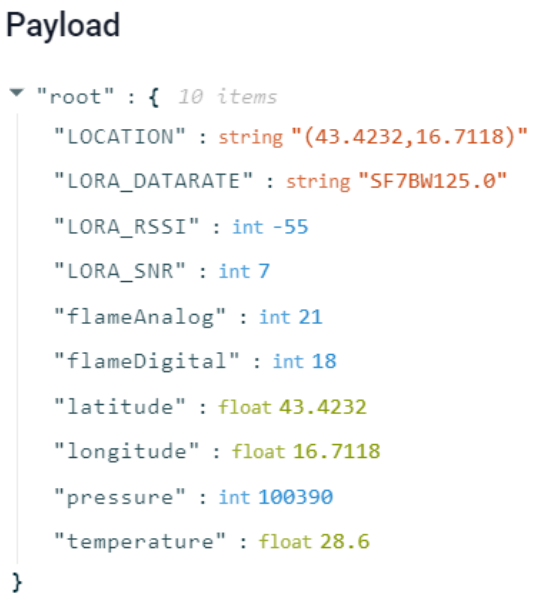
Slika 6.12. prikazuje podatke kakve vidimo u Serial monitoru IDE-u.

Sljedeće gdje se mogu vidjeti podatci je The Things Network Console. Prvo što se vidi su podatci o uređaju s kojeg je poslana. Kako je za način aktivacije korištena OTAA još imamo i podatke: dev_eui, join_eui, dev_addr koje generira The Things Network i služe za spajanje na mrežu. Vidi se i vrijeme primanja poruke. Zatim slijede podatci o tlaku, temperaturi, GPS podatci, i signal s detektora plamena. U dekoderu na TTN platformi dodan je i dio koji odlučuje u odnosu na vrijednost analognog signala sa detektora plamena da ispiše dali je detektiran plamen ili nije. Potrebno je napomenuti da analogna vrijednost u serial monitoru nije ista kao u TTN-u i Datacake radi kriptiranja za slanje tj nismo koristili dovoljno bajtova da smanjimo količinu podataka.



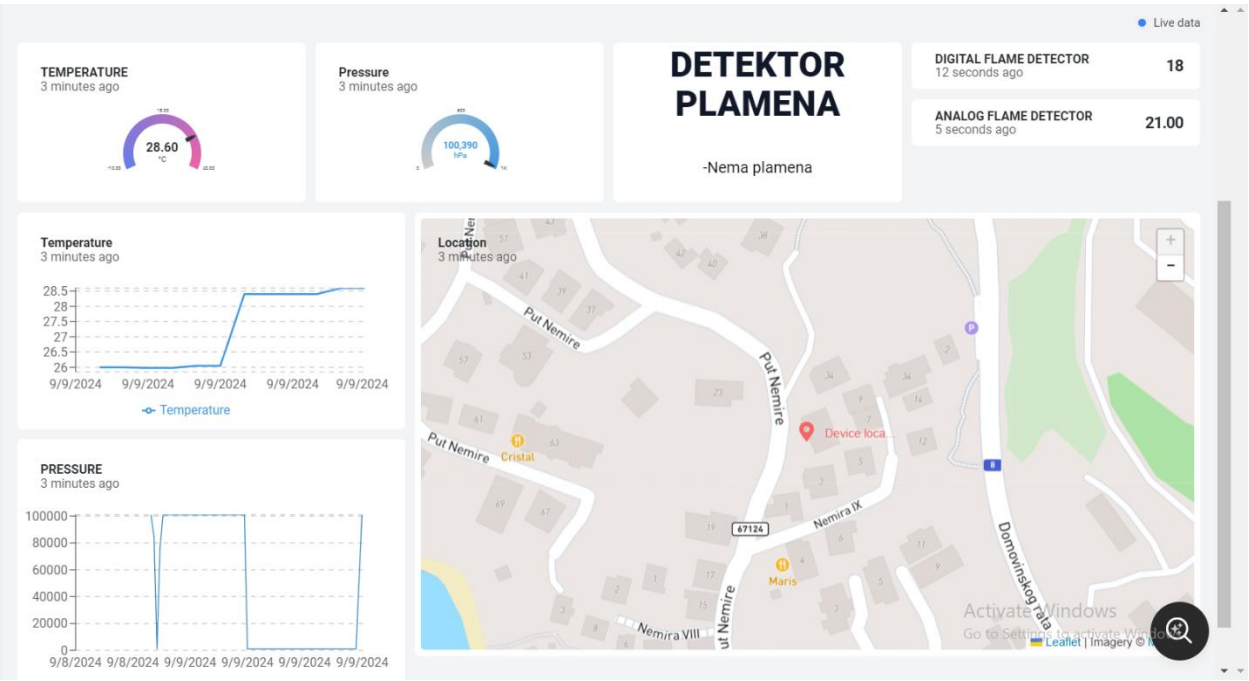
Slika 6.13. prikazuje podatke u The Things Networku.

Podatci se zatim mogu vidjeti u Datacake debug prozoru unutar aplikacije. Podatke koje primi aplikacija koju smo prethodno povezali s našim TTN računom je potrebno dekodirati. Nakon dekodiranja podatci izgledaju kao na slici 6.12.



Slika 6.14. prikazuje izgled podataka u debug prozoru u Datacake aplikaciji.

Nakon što su podatci dekodirani potrebno ih je prikazati u dashboardu. Prvo je potrebno kreirati polja (engl. Fields) u kojima definiramo ime, vrstu podatka i ulogu. Zatim se ta polja povežu s podatcima koji pristižu u aplikaciju. Na kraju kreiramo dashboard izgled tako da dodajemo widget-e i povezujemo ih s određenim poljima. Na kraju svaki podatak koji stigne, dekodira se i povezan je preko nekog polja s dashboard-om bude prikazan ovisno o vrsti dashboarda koji se koristi. Slika 6.13. pokazuje izgled dashboarda korištenog za ovaj rad.



Slika 6.15. prikazuje izgled dashboarda u aplikaciji Datacake

7. ZAKLJUČAK

Zaključak ovog rada je da LoRaWAN kao takav ima velik potencijal i sigurno je jedna od tehnologija budućnosti. Mala potrošnja i mogućnost za prebacivanje na velike udaljenosti su karakteristike koje joj daju prednost u odnosu na druge tehnologije za prijenos podataka u IoT-u. Izrađen je sklop koji prikuplja podatke o temperaturi, tlaku, GPS lokaciji i detektira plamen. Za to je napisan potreban program u C++ jeziku koristeći Arduino IDE-e. Podatci koje je Esp32 pločica prikupila sa senzora se dalje šalju u The Things Network. Komunikacija između uređaja i gateway-a koji podatke šalje na internet odvija se LoRa tehnologijom. TTN račun povezan je s DataCake platformom na koju se prosljeđuju podatci i prikazuju na dashboardu koje je moguće prilagodit i dizajnirat po vlastitom izboru.

LITERATURA

1) https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/ (6.3.2023.)
2) https://lora.readthedocs.io/en/latest/#modulation-types-and-chirp-spread-spectrum (3.6.2023.)
3) https://www.mokosmart.com/hr/lora-technology/ (6.3.2023.)
4) https://hz137b.p3cdn1.secureserver.net/wp-content/uploads/2020/11/what-is-lorawan.pdf?time=1680542827 (5.4.2023.)
5) https://www.researchgate.net/publication/323258913/figure/fig5/AS:631607719391275@1527598423702/LoRa-star-network-architecture.png (11.4.2023.)
6) https://www.mokosmart.com/hr/lora-frequency/ (11.4.2023.)
7) https://lora.readthedocs.io/en/latest/#modulation-types-and-chirp-spread-spectrum (18.4.2023)
8) https://www.inpixon.com/technology/standards/chirp-spread-spectrum (18.4.2023.)
9) https://datacake.co/low-code-iot-platform (18.4.2023.)
10) https://www.youtube.com/watch?v=5yUbUV-KgFE (18.4.2023.)
11) https://predictabledesigns.com/wp-content/uploads/2019/11/word-image-67.jpeg (23.4.2023.)
12) https://www.researchgate.net/publication/334572715_Using_the_ESP32_Microcontroller_for_Data_Processing (23.4.2023.)
13) https://makeradvisor.com/esp32-development-boards-review-comparison/ (23.4.2023.)
14) https://www.espressif.com/sites/default/files/documentation/esp32-wroom-32_datasheet_en.pdf (23.4.2023.)
15) https://www.thethingsnetwork.org/community/split/ (4.5.2023.)
16) https://ttnmapper.org/heatmap/ (4.5.2023.)
17) https://lora-alliance.org/ (6.5.2023.)
18) https://www.rfwireless-world.com/images/LoRa-modulated-signal.jpg (6.5.2023.)
19) https://www.mokolora.com/lora-and-wireless-technologies/ (11.5.2023.)
20) https://m.media-amazon.com/images/I/51FzgrVl-cL._AC_.jpg (12.5.2023.)
21) https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf (16.5.2023.)
22) https://vidi-x.org/primjena/ (16.5.2023.)

23) https://wiki.seeedstudio.com/Grove-Barometer_Sensor-BMP180/ (20.10.2023.)
24) https://www.makerfabs.com/bmp180-barometric-pressure-temperature-altitude-sensor.html (25.10.2023.)
25) https://www.adafruit.com/product/1603 (25.10.2023.)
26) https://botland.store/img/art/inne/13032_5.jpg (27.10.2023.)
27) https://www.sunfounder.com/cdn/shop/products/TS0226D-01_700x.jpg?v=1600170786 (23.1.2024.)
28) https://www.sunfounder.com/products/flame-sensor-module (23.1.2024.)
29) https://uk.robotshop.com/cdn/shop/products/sunfounder-ad-da-converter-pcf8591-module_38e928d1-12f7-40be-a35f-06ae318ccbf6_800x.jpg?v=1696282438 (31.1.2024.)
30) https://cdn.sparkfun.com/assets/learn_tutorials/8/0/4/RFM95_96_97_98W.pdf (21.2.2024.)
31) https://5.imimg.com/data5/WU/YU/RK/SELLER-44114218/rfm95w-868s2-rfm95-lora-ultra-long-range-transceiver-module-gfsk-gmsk-lora-ook-spi-500x500.jpg (21.2.2024.)
32) https://content.u-blox.com/sites/default/files/products/documents/NEO-7_ProductSummary_%28UBX-13003342%29.pdf (6.3.2024.)
33) http://wiki.sunfounder.cc/index.php?title=Flame_sensor_Module (18.3.2024.)
34) https://researchdesignlab.com/projects/pcf8951.pdf (19.3.2024.)
35) http://wiki.sunfounder.cc/index.php?title=PCF8591_8-bit_A/D_and_D/A_converter_Module (22.3.2024.)
36) https://www.thethingsnetwork.org/ (6.5.2024.)
37) https://www.thethingsnetwork.org/docs/network/architecture/ (11.5.2024.)
38) https://www.elektronika.ftn.uns.ac.rs/et-mikroracunarska-elektronika/wp-content/uploads/sites/138/2018/03/SSK.pdf (21.5.2024.)

.

POPIS SLIKA

POPIS TABLICA

PRILOZI

Popis kratica

OTAA	Over the Air Activation
APB	Activation By Personalization
IoT	Internet of Things
NWKSKEY	Network Session Key
APPSKEY	Application Session Key
DEVADDR	device address
TTN	The Things Network
M2M	Machine to Machine
CSS	Chirp Spread Spectrum
SoC	System of Chip
ISM	The Industrial, Scientific, and Medical