

**NAČRTOVANJE RAČUNALNIŠKIH SISTEMOV**

**Tehnična dokumentacija projekta**

Maribor, januar 2020

**Avtorji**:

Alen Poklič

Domen Osojnik

Gal Glogovšek

Vsebina

[1. Načrtovanje sistema 3](#_Toc61693590)

[1.1. Zasnova rešitve na podlagi obstoječega sistema 3](#_Toc61693591)

[1.2. Izdelava projektne sheme 4](#_Toc61693592)

[2. Implementacija 5](#_Toc61693593)

[2.1. Kompas 5](#_Toc61693594)

[2.2. .NET aplikacija 6](#_Toc61693595)

[2.3. Zaledni del 7](#_Toc61693596)

# Načrtovanje sistema

## Zasnova rešitve na podlagi obstoječega sistema

Končni cilj projekta je bila nadgradnja obstoječega sistema, katerega smo implementirali tekom lanskega semestra kot del projekta **Analize masivnih podatkov za aplikacije** **v** **realnem svetu**, modul **Infrastrultura**, z uporabo znanja, ki smo ga pridobili letos pri predmetu **Načrtovanje računalniških sistemov**.

Obstoječa rešitev je bila zgrajena iz štirih delov:

* mobilne aplikacije, ki zajema senzorske podatke pametnega telefona,
* zalednega dela (Node), strežnika, ki prejema, pošilja, ter obdeluje vse zajete podatke,
* podatkovne baze (MongoDB),
* osprednjega dela (Angular), odjemalca – spletna stran, ki prikazuje rezultate vseh meritev/izračunov.

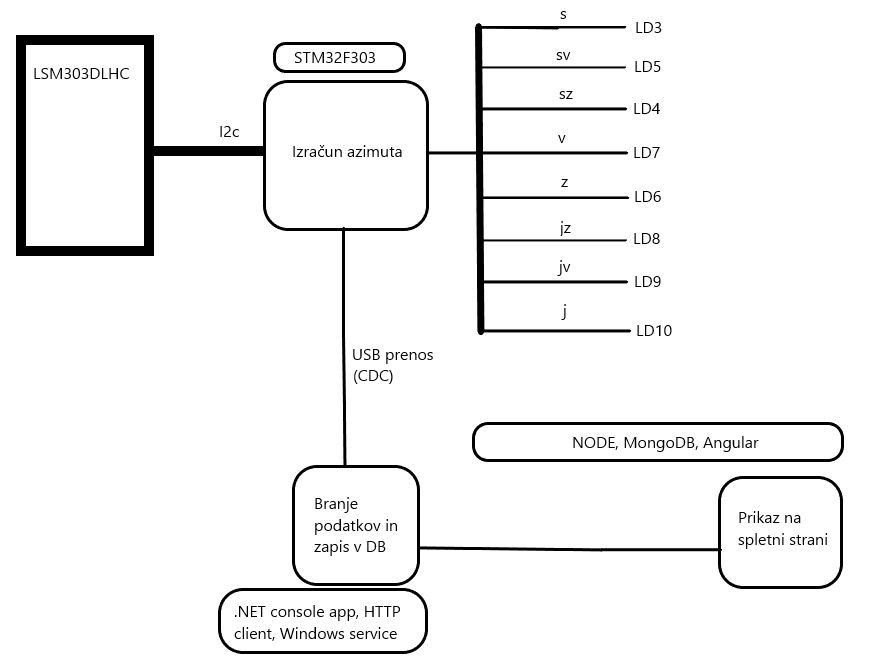
Rešitev nam omogoča pregled morebitnih trenutnih dogodkov na slovenskih cestah in statistike na avtocestah, ki jih beležijo na spletni strani promet.si. Spremljamo lahko tudi statistike naših voženj (zaznava prometnih znakov s pomočjo računalniškega vida ter stanja vozišča s senzorji telefona): telefon s pogledom na cesto pred vožnjo fiksno namestimo v našem vozilu, zaženemo aplikacijo ter omogočimo zajem podatkov.

V sistem je bilo potrebno dodati nov del: mikrokrmilnik, ki, podobno kot pametni telefon, zajete senzorske podatke posreduje v zaledje. Senzorji, ki jih pri tem uporabimo naj bi bili **pospeškometer**, **giroskop** ter **magnetometer**, ki skupaj gradijo **IMU** sistem.

Odločili smo se izdelati kompas, torej na osnovi senzorskih podatkov izračunati trenutno usmerjenost naše ploščice in s tem tudi usmerjenost vozila med vožnjo. V ta namen je bilo najlažje uporabiti izposojeni univerzitetni mikrokrmilnik SM32 F3 Discovery, ki pa sam po sebi ne omogoča brezžične komunikacije. Za potrebe projekta smo se odločili ploščico žično povezati kar na PC, ki pa nato dalje podatke posreduje na zaledni del, torej je bilo potrebno izdelati še programsko opremo, ki bo omogočila to vmesno komunikacijo.

## Izdelava projektne sheme

Naslednji korak je bila izdelava projektne sheme na podlagi zasnovanega načrta, po kateri se bomo ob implementaciji lahko dalje zgledovali. V njej so prikazani vsi posamezni deli, združeni v celoto s prikazom medsebojnih odvisnosti in načini komunikacije med temi.



Slika 1: Shema projekta

# Implementacija

## Kompas

Za implementacijo kompasa smo uporabili podatke magnetometra in pospeškometra. Same podatke je bilo potrebno pretvoriti v ustrezne enote (G, nato v mikro tesle), ter nato še kompenzirati nagib.

Text

Description automatically generated

Slika 2: Prebiranje podatkov magnetometra in kalibracija

Graphical user interface, text, application, email

Description automatically generated

Slika 3: Primer prižiga luči za smer sever

Da je kompas deloval tudi ob nagibu ploščice, smo dodali še kompenzacijo nagiba iz meritev pospeškometra:

Graphical user interface, text, application

Description automatically generated

Slika 4: Kompenzacija nagiba s pomočjo pospeškometra

## .NET aplikacija

Za prenos rezultatov na PC, ki jih daje mikrokrmilnik na podlagi stanj njegovih senzorjev, smo napisali .net konzolno aplikacijo. V produkciji bi lahko program v sistem namestili kot servis (topshelf-project.com) ki se avtomatično zbudi ob zagou računalnika, med razvojem pa smo ga zaganjali ročno.

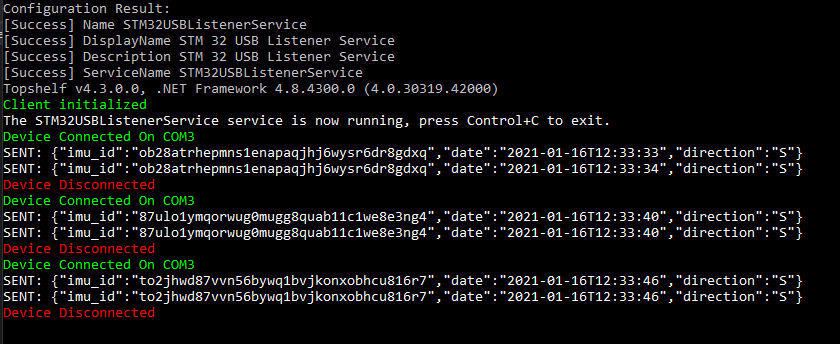
V začetni fazi delovanja program vzpostavi vse potrebno za delovanje v obliki servisa, nato pa v novi niti začne začne poslušati na vseh priklopih računalnika za našo ploščico. To prepozna s pomočjo »product string« konfiguracije USB na ploščici (product string = myIMU).

Ko identificira napravo, blokira glavno nit tako dolgo, da ne prejme podatkov. Po prejemu podatkov jih najprej validira in formatira (zapiše v instanco razreda Packet, ki predstavlja strukturo naše zahteve).

Sporočila pričakuje v 4B obliki, kjer sta prva dva byta kot header namenjena validaciji paketa (AB), ostala dva byta pa predstavljata sam podatek. Možna stanja podatka vključujejo: 0J, 0V, 0Z, 0S, SV, SZ, JV, JZ.

Če so vsi podatki validni, jih s POST zahtevo posreduje na zaledni del (serializacija Packet objekta razreda). Ploščico lahko v katerikoli fazi delovanja programa izklopimo in ponovno priklopimo (ujete so vse izjeme), pri čemer se vsak ponovni priklop šteje kot ločeno »zajemanje« s skupnimi id-ji.

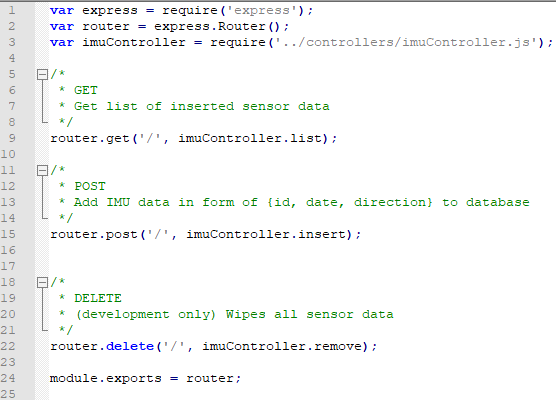
Vse nastavitve (URL strežnika, Pot na strežniku, dolžina sporočila) lahko tudi po namestitvi spreminjamo v App.config.



Slika 4 Delovanje konzolne aplikacije

## Zaledni del

Na strežnik smo dodali novo pot (/imu), ki pričakuje POST, GET in DELETE zahteve.



Vsako senzorsko zajemanje vsebuje:

* imu\_id – id zajemanja (ta se spreminja ob vsakem ponovnem zagonu mikrokrmilnika),
* date – čas in datum zajetega podatka,
* direction – smer neba, kot jo je senzor v tem časovnem okvirju zaznal

POST zahteve se poslužuje ploščica, ko podatke zapisuje na strežnik (ti se preprosto zapišejo v podatkovno bazo, GET uprablja osprednji del pri urejenem prikazovanju, DELETE pa je namenjen striktno samo ob razvoju (kasneje bi ga bilo potrebno odstraniti ali ob klicu preverjati pravice....).

Text

Description automatically generated

Slika 5: Zahteva za pridobitev seznama podatkov

## Čelni del

Na sprednjem delu smo pridobljene rezultate senzorja prikazali. Podatki sestojijo iz podatka o identifikaciji seanse, datuma zapisa podatka, ter sama smer, ki je preračunana na mikrokrmilniku.

Graphical user interface, application, Teams

Description automatically generated

Slika 6: Prikaz podatkov senzorja na spletni strani