Гимназија „Јован Јовановић Змај”

Нови Сад

Матурски рад из Програмских парадигми
Трилатерација јачином сигнала

Професор ментор
Миодраг Ђукић

Ученик
Марко Кувизић IV-7

Novi Sad, март 2020. год.

Предговор

Није било лако одабрати тему за овај рад, делом због огромног броја крајње интересантних тема, а делом због разлика у комплексности између њих. Циљ је био да се у процесу писања овог рада употреби што шире знање из стручних предмета, из целе средње школе, али и да се научи по нешто ново. Из тог разлога одабрао сам тему која најбоље пристаје предмету „Програмске парадигме" али обухвата и неколико предмета који су на мене лично оставили најјачи утисак током средње школе, као и теме које се у опште не појављују, у оквиру било ког предмета. Један резултат овог приступа јесте да је тема доста комплекснија него што је морала бити, а рад заиста опширан.

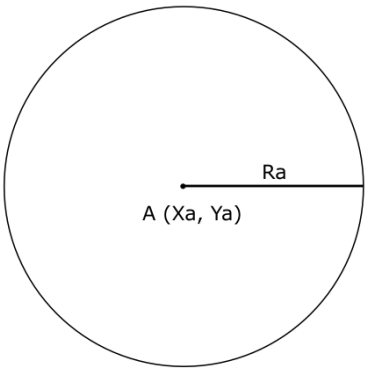
Садржај

1 Увод

Системи за позиционирање су тема која је по правилу резервисана за интернационалне корпорације, са незамисливом количином ресурса. Раде се помоћу сателита и других облика инфраструктуре, а не помоћу *Android* апликација и несавршених података. Осим тога, ова тема, конкретно „Трилатерација јачином сигнала" се не може уредно подвести под један предмет. Потребно је знање физике, програмирања, математике, а може се применити чак и статистика. Ни једна и мало интересантна примена програмирања се не може добро обрадити без знања из области сродних рачунарству, управо математике и физике. Из тог разлога овај рад је рађен у оквиру предмета „Програмске парадигме", тема се не тиче искључиво програмирања, али је та област најбитнија за тему.

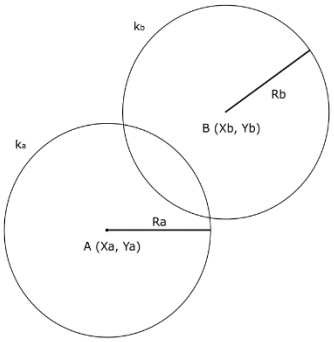
2 Трилатерација

Трилатерација је, укратко, један од процеса којим се може утврдити локација, на основу познате раздаљине од неких тачака. Потребне су локације три тачке, као и позната раздаљина између локације коју утврђујемо и сваке од те три тачке. Следи кратко графичко објашњење трилатерације.



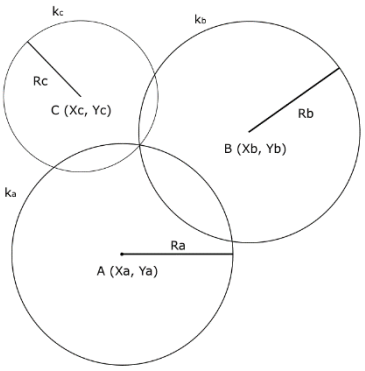
Слика 1

Узмимо познату тачку А, са координатама X_a и Y_a(За сврхе овог објашњења радимо у равни). Пошто је кружница дефинисана својим центром и полупречником, ако узмемо за полупречник познату раздаљину између тачке коју тражимо и тачке А, можемо „нацртати" кружницу k_a(А, R_a). Тачка коју тражимо се сигурно налази на кружници k_a.



Слика 2

Ако поновимо овај поступак за неку тачку В (X_b, Y_b) и раздаљину R_b, добићемо кружницу k_b(В, R_b). У пресеку ове две кружнице ће најчешће бити две тачке. Једна од ове две тачке је она коју тражимо.



Слика 3

Коначно, понављамо овај поступак и трећи пут за тачку С, и добијамо кружницу k_c(С, R_c). У савршеном систему, ова кружница ће садржати једну од тачака које смо добили у другом кораку, то је тачка коју тражимо.

2.1 Проблем раздаљине

Под претпоставком да нам је локација ове три тачке позната (рецимо у виду координата), у овом систему остаје још једна непозната, раздаљина. Да бисмо конструисали три кружнице које су нам потребне за трилатерацију, потребна нам је раздаљина тачке коју тражимо, од центра кружнице. Ако схватимо ове тачке као локације неких одашиљача, а тачку коју тражимо као пријемник, рецимо мобилни телефон, постоји неколико начина да се ова раздаљина израчуна. Следи кратко објашњење два начина која се најчешће користе.

2.1.1 Трилатерација временом пристизања сигнала

(енг. TDOA - time difference of arrival)

Принцип рада утврђивања раздаљине овом методом се ослања на рачунање разлике између времена одашиљања сигнала и времена пристизања сигнала на пријемнику. Ова разлика у принципу представља време које је потребно да би талас прешао раздаљину између одашиљача и пријемника, то јест између познате тачке и тачке коју тражимо.

$$d = c * \sqrt{t_1 - t_2}$$

формула 1 - Раздаљина између одашиљача и пријемника

Формулом 1 се рачуна ова раздаљина, где је t_2 време одашиљања сигнала, t_1 време пристизања сигнала, а с брзина светлости.

$$d = \sqrt{(x_{ref} - x)^2 + (y_{ref} - y)^2}$$

формула 2 - Кружница могућих тачака на основу формуле 1

Онда, на основу формуле за кружницу из аналитичке геометрије, формула 2 представља кружницу свих могућих тачака на основу једне референтне тачке. Додавањем референтних тачака смањујемо број могућих тачака (ово је формула за дводимензионални простор).

Тачност израчунатих раздаљина код овог метода у највећем делу зависи од тачности мерења времена. Грешка у мерењу времена реда величине наносекунде производи грешку у позицији реда величине десет центиметара (*Key & Lemmens*, 2005). У пракси, мерења времена са толико малом грешком се једино могу постићи коришћењем атомског сата, међутим, овакви сатови су прескупи и непрактични за коришћење. Најпознатија примена трилатерације временом пристизања сигнала јесте у GPS (*Global Positioning System*) мрежи, где су одашиљачи сателити. GPS у општем случају користи комбинацију атомских сатова на сателитима, и релативно јефтиних кварцних сатова доступних у паметним телефонима. Ова комбинација производи мерења времена са грешком реда величине десет наносекунди, што се сматра довољно прецизним. Главни проблем са овим приступом, поготово за сврхе

овог рада, јесте цена и приступачност. Као што смо видели овај приступ може да резултује заиста врло прецизним мерењима, али се не може имплементирати једноставно, у виду апликације за рецимо *Android*, и једноставно нам није доступан.

2.1.2 Трилатерација јачином сигнала

Трилатерација јачином сигнала је у најбољем случају компромисно решење. Ослања се на јачину сигнала на пријемнику, такозвану RSSI вредност, да би коришћењем модификоване верзије Фријеве[1] формуле за преношење сигнала у простору, дала донекле прецизну процену раздаљине од одашиљача.

$$P_r = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 R^n}$$

формула 3 - Фријева формула[2]

Где је:

- P_t– Јачина сингала на предаји
- P_r– Јачина сигнала на пријему
- G_t – Добитак предајника
- G_r – Добитак пријемника
- R – Раздаљина
- n – Коефицијент ширења сигнала
- λ – Таласна дужина

Најмања теоретски могућа грешка у позиционирању је реда величине 1 до 2 метра, али у пракси се очекује много већа грешка. Као што јасно видимо, ова формула има много више променљивих од формуле за брзину. То су G_t, G_r, добитак одашиљача и пријемника, n коефицијент ширења сигнала, и P_t, јачина сигнала на предаји. У контролисаном, затвореном простору, можемо контролисати ове варијабле. Кад бисмо, на пример, имали своје одашиљаче, знали бисмо њихов добитак и снагу, и она би била константна. У овим случајевима се може преко Фријеве формуле добити довољно прецизна процена раздаљине (у мојим мерењима са познатом снагом и добитком грешка при позиционирању била је до десет метара). Из ових разлога трилатерација јачином сигнала се примењује углавном у затвореним просторима, са мањим бројем одашиљача, чије су спецификације познате. Међутим, нама није доступна таква мрежа.

За сврхе овог рада постоје у суштини две мреже које можемо користити. Можемо користити кућну бежичну мрежу, под условом да имамо барем три *router*-а, идеално би било

[1] Енг. *Harald Trap Friis, данско-амерички инжењер.*

[2] У неким верзијама ове формуле се узима n=2, и постоји неколико различитих верзија које дају сличне резултате

и више, које ћемо користити као одашиљаче. Ова варијанта би највероватније дала прилично прецизне резултате, али би била доста уско примењива. Друга опција је да користимо мобилну мрежу. Главна предност коришћења мобилне мреже за ове сврхе је у томе што је покривеност ове мреже одлична, до те мере да можемо очекивати да имамо барем три доступна одашиљача, што је минимални број одашиљача потребан за трилатерацију. Осим тога, мобилне GSM, UMTS и LTE мреже су до те мере распрострањене да нам неће бити потребна никаква додатна инфраструктура. На овај начин трилатерација се може имплементирати у виду *Android* апликације. *Android* нам омогућава приступ јачини сигнала према одашиљачима у мобилној мрежи у децибел-миливатима, међутим овај податак, који нам је неопходан за трилатерацију, једино је доступан за сада застарелу GSM 2G мрежу, о овоме ће бити више речи у поглављу 3, које се бави имплементацијом. Постоје и значајни проблеми са коришћењем мобилне мреже за ове сврхе. Ови проблеми су углавном повезани са горе поменутим променљивама у Фријевој формули, у највећем делу са снагом и добитком одашиљача.

2.1.2.1 Мало о коефицијенту ширења сигнала

Коефицијент ширења сигнала је променљива која у Фријевој формули прави разлику у преношењу сигнала у различитим окружењима. Многи извори наводе да се он креће између 2 и 4, и ово јесте тачно у већини свакодневних окружења, међутим овај коефицијент може да достигне и вредности између 4 и 6, рецимо унутар зграда, или у спортским стадионима. Исто тако, постоје јако ретке ситуације у којима се овај број креће између 1.5 и 2, пошто неке јако специфичне врсте архитектуре, рецимо тунели, ублажују слабљење сигнала са раздаљином.

2.1.2.2 Проблем снаге

Дакле, највећи проблем са овом методом трилатерације јесте проблем снаге[3] одашиљача. У принципу, постоји неколико различитих „класа" мобилних одашиљача који се користе у мрежи, у зависности од ситуације. Најснажнији одашиљачи имају горњу границу снаге од 20 W (Напомена - за ове сврхе користимо процене снаге изражене у ватима, због чињенице да се у Фријевој формули коју користимо очекује та мерна јединица, иначе се снага одашиљача може изразити и у децибел-миливатима [Dbm]), али слабији одашиљачи често користе снагу вишеструко мању од ове границе, реда величине 0.1 до 1 W, а у одређеним ситуацијама се појављује и снага мања од 0.1 W. Ово је први проблем снаге, различити одашиљачи који се користе на различитим местима у мрежи имају драстично различите снаге.

Наивна претпоставка би била да постоји нека каталошка вредност која може да представља довољно добар просек снаге свих одашиљача, међутим разлика у снази између 0.1 W и 20 W даје разлику у раздаљини од неколико десетина километара (у зависности од

[3] У овој ситуацији реч „снага" се користи као краћа замена за стручни израз „јачина сигнала на предаји". Ово је у складу са већином извора на страним језицима који обрађују сличне теме, пошто је тај стручни израз заиста непрактичан.

коефицијента ширења сигнала). Ово је наравно екстремни случај, али јасно је да било каква процена раздаљине изведена на овај начин неће бити ни најмање прецизна.

Следећа логична идеја је да можемо за сваки одашиљач на познатој раздаљини измерити јачину примљеног сигнала, и затим за дату раздаљину израчунати снагу за дати одашиљач (поступак се може поновити на више раздаљина). Овом методом бисмо добили добру процену јачине сигнала на предаји сваког од одашиљача. Поступак је доста кратак, и треба га поновити за свега неколико десетина одашиљача да бисмо могли да покријемо цео Нови Сад. Међутим, и овде постоји проблем. По регулацијама за мобилне мреже у Србији, провајдери не би смели да мењају јачину сигнала на предаји базних станица. Свака базна станица има своју каталошку снагу на предаји, и свака станица би морала по правилима да одашиље сигнал те јачине. Мала пословна тајна је да се ова правила не поштују. Провајдери имају своје алгоритме по којима рачунају и мењају снагу на предаји сваке појединачне станице, у циљу смањења трошкова. Овај феномен се често назива „дисање мреже" и за последицу има чињеницу да у различитим тренуцима иста базна станица може имати драстичне варијације у јачини сигнала на предаји, а већ смо закључили да и мале варијације ове променљиве производе огромне разлике у раздаљини.

Овај проблем, барем за сврхе и могућности једног матурског рада, нема савршено решење. Нама нису доступне ни каталошке спецификације појединачних базних станица, ни алгоритам који било који провајдер користи, нити можемо прецизно претпоставити јачину сигнала на предаји на основу времена или датума, пошто она у суштини зависи од оптерећености мреже у датом тренутку, која је некад предвидива у односу на време, али најчешће се не може прецизно претпоставити.

Решење које је примењено у апликацији приложеној уз овај рад представља несавршено, компромисно, програмско решење проблема који је по природи делимично инжењерски а делимично статистички. Као такво ово решење не представља модел корисничке апликације, нити неког глобалног система за позиционирање (то би наравно било немогуће без озбиљне инфраструктуре), већ је, у ограничењима једног матурског рада, најбољи могући доказ концепта трилатерације јачином сигнала.

До сада смо у суштини претпостављали да јачина сигнала опада на у потпуности предвидљив начин у односу на раздаљину (у зависности од коефицијента ширења сигнала n), то јест да за равно, отворено поље јачина сигнала опада са квадратом раздаљине (одавде коефицијент ширења сигнала расте у зависности од окружења). Систем који функционише под овом претпоставком може прецизно да процени локацију само у мрежи где константно „видимо" све одашиљаче. На пример, када бисмо могли да поставимо (најмање) три GSM базне станице, на отвореном равном пољу, овај систем би функционисао за позиционирање у тој мрежи. Ипак, када бисмо могли ово да урадимо, не бисмо имали ни проблем снаге. Ово у ствари представља идеалан систем за трилатерацију јачином сигнала. Нажалост, 2G мрежа није примењена у оваквом систему. Између нас и 2G базне станице у датом тренутку могу бити зграде, зидови, нераван терен, дрвеће, и небројиво много других препрека. У овим случајевима сигнал се често одбија од њих, чиме се продужава његов пређени пут. Такође

се мења функција којом јачина сигнала опада са раздаљином. Један, можда најшире прихваћен начин којим се рачуна ова промена је Рајлијев фејдинг. Рајлијев фејдинг је статистички модел којим се рачуна ефекат средине на преношење сигнала. Овај модел не само да је врло комплексан, већ је статистичко решење, и из ових разлога рачунање Рајлијевог фејда је у потпуности изван опсега овог рада. Генерално, овакви статистички модели су изван опсега знања ученика средње школе, предају се на неколико смерова у каснијим годинама факултета. Стога, наше процене раздаљине биће доста мање прецизне.

2.1.2.3 Примене и прецизност

Једна честа примена трилатерације јачином сигнала се може наћи код хитних служби, поготово код полиције. Често особа која позове полицију не зна, или не може да да своју тренутну локацију. Исто тако често неће бити GPS сигнала. У овим ситуацијама хитне службе у западним државама користе трилатерацију на мобилној мрежи.

Подаци о прецизности у Србији нису толико доступни, али америчка Федерална Комисија за Комуникацију (енгл. FCC) у једном извештају (*Tran*, 2015) тврди да овим системом може да лоцира паметни телефон са грешком од око три четвртине квадратне миље, или око 1.2 квадратна километара. Другим речима могу да нађу област од око 1.2 квадратна километара у којој ће се сигурно налазити телефон. Вреди споменути да и ако се овај систем, конкретно систем којим се бави рад FCC-а, бави углавном трилатерацијом, ипак је примењено још неколико помоћних решења која побољшавају прецизност и покривеност система.

Друга позната примена трилатерације јачином сигнала је помоћни систем *Google* мапа. Ако корисник *Google* мапа покуша да добије локацију са искљученим *GPS*-ом (или у ређем случају без GPS сигнала) и без *wifi* сигнала, *Google* мапе прелазе на систем трилатерације јачином сигнала према мобилној мрежи. Овде прецизност коју *Google* обећава драстично варира. Углавном је грешка између 1600 и 1000 метара, али може бити и око 500.

2.2 Аналитичка геометрија

Сада, под претпоставком да имамо познате раздаљине и центре кружница на основу предајника, потребно је применити аналитичку геометрију да бисмо у дводимензионалном координатном систему одредили локацију пријемника. Почнимо од два предајника.

$$(x - p_1)^2 + (y - q_1)^2 = {r_1}^2$$

$$(x - p_2)^2 + (y - q_2)^2 = {r_2}^2$$

формула 4 - Формула кружнице у дводимензионалном координатном систему

Координате предајника p_1, q_1 і p_2, q_2 користимо као координате центара две кружнице, а раздаљине r_1, r_2 као полупречнике. Сада, две кружнице могу да се односе на три начина:

1) Секу се (у две тачке)

   Ово је очекивани случај, и значи да се пријемник налази у једној од те две тачке.

2) Додирују се (у једној тачки)

   Ово би у суштини значило да се пријемник сигурно налази у тој једној тачки. Овај случај је могућ у теорији, али је у пракси толико редак да га није потребно ни обрађивати.

3) Не додирују се (не секу се ни у једној тачки)

   Овај случај је такорећи случај грешке. За нас ће значити да смо погрешно поставили неку од променљивих, и због тога добили потпуно нетачну раздаљину, или да имамо погрешну локацију неког од предајника. Овај случај ће се неизбежно дешавати у неком проценту мерења и мораћемо да га обрадимо.

Систем једначина из формуле 3 је иначе једноставно решити. Да нам је потребно решење за један систем, могли бисмо да помножимо једну једначину са неким бројем, одузмемо од друге, изразимо једну од две променљиве и тако решимо једначину. Нама је ипак потребно решење које можемо применити за сваки систем једначина. Мораћемо да изведемо формулу.

Корак 1 – распишимо квадрате бинома у оригиналним једначинама:

$$\textit{једначина 1: } x^2 - 2p_1x + {p_1}^2 + y^2 - 2q_1x + {q_1}^2 = {r_1}^2$$

$$\textit{једначина 2: } x^2 - 2p_2x + {p_2}^2 + y^2 - 2q_2x + {q_2}^2 = {r_2}^2$$

Корак 2 – помножимо једну од једначина са -1

$$1: x^2 - 2p_1x + {p_1}^2 + y^2 - 2q_1x + {q_1}^2 = {r_1}^2 / \cdot(-1)$$

$$\downarrow$$

$$1: -x^2 + 2p_1x - {p_1}^2 - y^2 + 2q_1x - {q_1}^2 = -{r_1}^2$$

$$2: x^2 - 2p_2x + {p_2}^2 + y^2 - 2q_2x + {q_2}^2 = {r_2}^2$$

Корак 3 – сабирамо две једначине

$$2x(p_2 - p_2) + {p_2}^2 - {p_1}^2 + 2y(q_1 - q_2) + {q_2}^2 - {q_1}^2 = {r_2}^2 - {r_1}^2$$

Корак 4 – изразимо x

$$x = \frac{{r_2}^2 - {r_1}^2 + {p_1}^2 - {p_2}^2 - 2y(q_1 - q_2) + {q_1}^2 - {q_2}^2}{2(p_1 - p_2)}$$

Корак 5 – ова једначина је једва разумљива, тако да ћемо увести смену на неколико места

$$t = \frac{{r_2}^2 - {r_1}^2 + {p_1}^2 - {p_2}^2 + {q_1}^2 - {q_2}^2}{2(p_2 - p_2)}$$

$$a = -\frac{\mathrm{q1} - \mathrm{q2}}{(p_1 - p_2)}$$

Jednačina sada postaje:

$$x = \mathrm{t} + \mathrm{ay}$$

Корак 6 – коначно, треба да уврстимо x у једну од почетних једначина кружнице. Добијамо квадратну једначину где су решења две могуће вредности за координату y.

$$y^2(a^2 + 1) - y(2ta - 2p_1a - 2q_1) + (t^2 - 2p_1t + {p_1}^2 + {q_1}^2 - {r_1}^2) = 0$$

Пошто су a и t познате константе које смо увели у виду смене, ова квадратна једначина се може решити у сваком случају где постоји пресек између две кружнице. Много важније, ову једначину ћемо моћи да имплементирамо и решавамо програмски.

3 Имплементација

3.1 Циљеви

На основу прецизности два система трилатерације јачином сигнала о којима је било речи у поглављу 2.1 можемо поставити одређене циљеве, што се тиче прецизности и покривености. За почетак, не би било реално очекивати резултате упоредиве имплементацији *Google*-а, или америчке полиције, од матурског рада из средње школе. У суштини постоје две опције, можемо жртвовати покривеност, или прецизност. У овој имплементацији жртвована је покривеност, у смислу да се морају прикупљати одређени подаци о базним станицама у датој области да би апликација радила за ту област. На овај начин се покривеност може проширити на било коју област која је покривена 2G мрежом, али уз та мерења.

3.2 *Android*-ов начин за добављање јачине сигнала

Android представља базне станице мобилне телефоније кроз класе које наслеђују класу *CellInfo* (*Anrdoid developers*, 2021), конкретно у зависности од технологије мобилне мреже то су:

- *CellInfoCdma*
- *CellInfoGsm*
- *CellInfoLte*
- *CellInfoNr*
- *CellInfoTdscdma*
- *CellInfoWcdma*

Ова класа има методу *getCellSignalStrength*, која у зависности од подкласе класе *CellInfo* враћа објекат одговарајуће подкласе класе *CellSignalStrength*, на пример *CellSignalStrengthGsm*. Објекти ове класе имају методу *getDbm* која враћа јачину сигнала према датој базној станици у децибел-миливатима. Једна мања препрека је што у Android-овом *api*-у метода *getDbm*, када је позвана над објектом типа *CellInfoLte*, који одговара верзији мобилне мреже са којом бисмо најрадије радили, увек враћа вредност *UNDEFINED*, то јест *int* вредност 2147483647 или *Integer.MAX_VALUE*. Из овог разлога ћемо се ограничити на GSM mrežu.

3.3 Класа *CellTower*

Ипак, већину података који су неопходни за ову имплементацију не можемо да добијемо од *Android*-а. Из овог разлога уводимо класу *CellTower*, која између осталог садржи поље типа *CellInfoGsm*.

```
protected double phoneGain = 2.2; //dbi
protected double towerGain = 11.3; //dbi
protected List<Double> powers = new ArrayList<Double>();
protected List<Double> distances = new ArrayList<Double>();
protected CellInfoGsm cellInfo;
protected WGS84 coords;
protected boolean accurate = true;
```

Код 1– Поља класе CellTower

Поља типа *double*, по имену *phoneGain* и *towerGain* представљају добре претпоставке за вредности добитка предајника и пријемника изражене на *dbi* скали. Од овога, 2.2 dbi (0 dbd) је опште прихваћена вредност за добитак међу произвођачима мобилних телефона, а 11.3 dbi (9.1 dbd) је средња вредност која ће произвести довољно добре резултате.

3.3.1 Проблем координата

Локација базних станица је једна од информација које не можемо да добијемо од *Android*-а. Ипак, постоје многобројне базе података које нам могу послужити. Идеално решење би било да користимо нечији *api*, и да једноставним захтевом добијамо локацију. Међутим, базе података које се баве Србијом су ипак доста ређе. Овде је коришћен скуп података у виду *csv* датотеке, састављен као део *open-source* пројекта *OpenCellId*. Табела садржи око 60000 редова где сваки представља једну базну станицу у Србији, и има колоне које садрже податке о географској ширини и дужини сваке од њих. Прецизност ових података је још један од ограничавајућих фактора за прецизност имплементације.

Једна базна станица је јединствено одређена скупом следећих вредности:

- Mcc – енг. *Mobile Country Code* – Одређује државу
- Mnc – енг. *Mobile Network Code* – Одређује мрежу, то јест мобилног оператера
- Lac – енг. *Location area code* – Одређује област (базне станице се групишу у области ради оптимизације)
- CiD – енг. *Cell ID* – Идентификатор

Парсирање ове csv датотеке је, наравно, тривијално. Тај проблем је решен у оквиру класе *CsvFile*, конкретно у њеној методи *read*, која враћа листу низова (матрицу) елемената типа *String*. Пошто горе поменута класа *CellTower* има поље типа *CellInfoGsm*, за сваки објекат типа *CellTower* конструктор ове класе може да прође кроз податке које враћа парсер *csv*-а и да на основу горе поменутих вредности нађе одговарајуће координате.

```
double latitude;
double longitude;
for(String[] row : res){
if(row[0].contains("GSM") &&
row[1].contains(String.valueOf(this.cellInfo.getCellIdentity().getMcc())) &&
row[2].contains(String.valueOf(this.cellInfo.getCellIdentity().getMnc())) &&
row[3].contains(String.valueOf(this.cellInfo.getCellIdentity().getLac())) &&
row[4].contains(String.valueOf(this.cellInfo.getCellIdentity().getCid()))){
                    latitude = Double.valueOf(row[7]);
                    longitude = Double.valueOf(row[6]);
                    this.coords = new WGS84(latitude, longitude);
                    Log.d("Tower Constructor Csv: ", "cellTower: Found");
            }
        }
```

Код 2– Део конструктора класе CellTower који се бави координатама

Променљива *res* представља горе поменуту листу низова коју враћа парсер *csv*-а. Овом петљом пролазимо кроз ставке у тој листи и када наиђемо ставку (ред из *csv*-а) која одговара базној станици, по 5 идентификатора *mcc*, *mnc*, *lac*, *cid* и мрежи којој припада, чувамо те координате. У следећем поглављу ће бити речи о класи *WGS84*.

3.3.2 Различите пројекције и конверзија

Пројекција округле Земље на равну површину је изненађујуће стар проблем. Могло би се рећи да нема савршеног решења. Најчешће примењена пројекција је WGS 84, са познатим степенима географске ширине и дужине. У овој пројекцији су нам дате координате базних станица, али не можемо на те вредности применити формуле из поглавља 2.2 које се бави аналитичком геометријом. Те формуле захтевају јединице које представљају међусобно једнаке раздаљине у дводимензионалном координатном систему, као на пример метре. За то се користи UTM пројекција, која дели Земљу на правоугаоне зоне означене словима абецеде и бројевима, унутар којих се локације описују помоћу две вредности, аналогне x и y вредностима у дводимензионалном координатном систему.

Постоје многе библиотеке за програмски језик *Java* које служе за конвертовање између различитих пројекција, а најпознатија међу њима је *Javaproject*, порт *Python* библиотеке. Проблем је што *Javaproject* не ради на *Android-y*. За ову имплементацију коришћене су класе *WGS84* и *UTM* са *Github*-а (Ansell, 2020), слично начину на који би био коришћен рецимо *Javaproject*.

3.3.3 Рачунање раздаљине

Због проблема описаних у поглављу 2.1, у највећем делу због проблема дисања мреже (алгоритмичких и у потпуности непредвидивих промена у јачини сигнала на предаји), а у мањем делу због непредвидиве природе преношења сигнала у окружењима која нису отворена поља, не можемо са великом сигурношћу одредити једну раздаљину између базне станице и пријемника. Прецизније речено, можемо одредити једну тачну раздаљину за један дати тренутак, међутим параметри које смо користили за тај тачан прорачун ће се врло брзо променити, најчешће због тога што је једна од базних станица чију јачину сигнала користимо променила јачину сигнала на предаји. Вреди споменути, са одређеном количином фрустрације, да овај проблем не би постојао да се провајдери мобилних мрежа придржавају правила споменутог у поглављу 2.1.2.

Несавршено програмско решење овог нерешивог инжењерског проблема је да у суштини израчунамо раздаљину за сваку могућу јачину сигнала на предаји. Да би се ово имплементирало уводимо променљиву која се може звати модификованом јачином сигнала на предаји. Ова променљива се добија на основу мерења јачине сигнала на пријему, на познатој раздаљини. Када се ове две променљиве уврсте у Фријеву формулу из поглавља 1.3, добија се променљива која представља јачину сигнала на предаји модификовану начином на који се опадање јачине сигнала у зависности од раздаљине убрзава или успорава у тој области (одатле назив модификована јачина сигнала на предаји). Вреди споменути да овај корак прикупљања података у опште не би био потребан у већини западних држава. За Сједињене Државе, Канаду, већину Европе, па чак и неколико напреднијих афричких држава, постоје скупови података који се састоје од мерења јачине сигнала на познатој раздаљини од базних станица. Често су то *open-source* пројекти, и тачно ти подаци су потребни за ову имплементацију. Нажалост, они за Србију или не постоје, или нису лако доступни. Из тог разлога ова имплементација ће се ослањати на податке прикупљене специфично за те сврхе, са ограниченим ресурсима и за кратко време, што представља највеће ограничење за покривеност и прецизност апликације.

Резултати овог поступка се чувају у другој *csv* датотеци, једноставније структуре. Пошто радимо на једној мрежи у једном граду биће довољно да базне станице једнозначно одређујемо њиховом *cell id* вредности. Ову *csv* датотеку парсира класа *CsvPower*, и она враћа листу низова (матрицу), где је сваки низ у листи скуп података за једну базну станицу. Конструктор класе *CellTower* онда налази одговарајуће податке за своју *cell id* вредност.

```
for(String[] row : powers){
    if(row[0].contains(String.valueOf(this.cellInfo.getCellIdentity().getCid()))){
                for(String cell : row){
                    if(cell.contains("0")){
                        this.powers.add(Double.valueOf(cell));
                    }
                }
            }
        }
        if(this.powers.size()!=0){
            this.powers.remove(0);
        }else{
            for(String cell : powers.get(powers.size() - 1)){
                if(cell.contains("0")){
                    this.powers.add(Double.valueOf(cell));//default vrednost
                }
            }
            if(this.powers.size()!=0){
                this.powers.remove(0);
                this.accurate = false;
            }
        }
```

Код 3 – Део конструктора класе CellTower који се бави модификованом јачином сигнала на предаји

Ова петља ће једноставно пролазити кроз листу низова, док не наиђе на низ који одговара *cell id* вредности објекта који се конструише. Онда ће додати сваки појединачни елемент низа у поље *powers*. Коначно, ако у *csv* датотеци нема података о базној станици на којој радимо, том објекту ће бити додељене такозване *default* вредности, које представљају средње вредности података о свим базним станицама. Такви објекти ће бити означени вредношћу *boolean-a accurate* (*false*), да би функције које ће ове податке користити за трилатерацију знале да дају већи приоритет објектима за који су модификоване јачине сигнала на предаји познате.

Сам процес рачунања раздаљине на основу ових вредности подељен је у две методе, *distanceCalculator* и *distanceCalcWrapper*.

```
public double distanceCalculator(double modifiedPower){

double squared = (modifiedPower * this.phoneGain * this.towerGain *
0.11095742873)/(Math.pow(4, 2) * Math.pow(Math.PI, 2) *
CellTower.dbmToWattConverter(this.cellInfo.getCellSignalStrength().getDbm()));
        this.distances.add(Math.pow(squared, 1/2.6));
        return this.distances.get(this.distances.size()-1);
    }

protected List<Double> distanceCalcWrapper(){

        for(Double modifiedPower : this.powers){
            this.distanceCalculator(modifiedPower);
        }
        return this.distances;
    }
```

Код 4- Методе које се баве рачунањем раздаљине у оквиру класе CellTower

Прва метода, *distanceCalculator*, је само имплементација Фријеве формуле из поглавља 1.1, она прима један параметар типа *double*, који представља модификовану јачину сигнала на предаји, и на основу тог параметра и јачине сигнала на пријему рачуна раздаљину, коју додаје у листу *distances*[4]. Друга метода, *distanceCalcWrapper*, позива методу *distanceCalculator* за сваку могућу модификовану јачину сигнала на предаји.

3.4 Класа *MainActivity*

Класа *MainActivity* је главна класа апликације. Она садржи промењиве које прате стање апликације, као и помоћне и главне методе које се баве самом трилатерацијом. Ова класа такође садржи поље типа *TelephonyManager* и преко њега добавља објекте типа *CellInfoGsm*, да би се могли конструисати објекти типа *CellTower*.

[4] Помоћна метода *dbmToWattConverter* конвертује вредност јачине сигнала на пријему из децибел-миливата у вате.

3.4.1 Поља класе *MainActivity*

```
protected TelephonyManager tel;
protected List<CellTower> towers = new ArrayList<CellTower>();
protected List<Double[]> points = new ArrayList<Double[]>();
protected GoogleMap map;
protected SupportMapFragment mapFragment;
protected WGS84 avgCoords;
protected Double[] sumAvg = {0.0, 0.0};
protected int numOfAvgs = 1;

private InputStream inputStream;
private InputStream powerInput;
private CsvFile csvFile;
private CsvPower csvPower;
private List<String[]> res;
private List<String[]> powers;

private PhoneStateListen phoneSTL = new PhoneStateListen(this);
private ImageView signalIndicator;
private int signalStr = 3;
private boolean failTracker = true;
```

Код 5 – Поља класе MainActivity

- Поље *tel* типа *TelephonyManager* се користи за добављање објеката типа *CellInfoGsm*, на начин објашњен у поглављу 3.2.
- Поље *towers* чува објекте типа *CellTower*, који представљају 2G базне станице.
- Поља *map* и *mapFragment* се баве радом са *Google* мапама.
- Поља *sumAvg*, *numOfAvgs* и *avgCoords* прате до сад израчунате локације ради рачунања просека
- Поља *inputStream*, *powerInput*, *csvFile*, *csvPower*, *res* и *powers* парсирају *csv* датотеке і чувају резултате тог парсирања
- Поља *signalIndicator*, *signalStr* и *failTracker* прате прецизност израчунате локације на основу броја познатих базних станица. За сваку базну станицу која није позната вредност поља *signalStr* се умањује за 1, и на основу те вредности користи се једна од 4 иконице које означавају јачину сигнала.

3.4.2 Почетно стање апликације

Одређене радње у овој апликацији треба обавити само једном, на самом почетку рада. Овим радњама се бави функција *locationSetup*, која се позива једном, приликом конструкије класе *MainActivity*. Ова метода тражи потребне дозволе, иницијализује променљиве класе *MainActivity*, по потреби их поставља на почетне вредности, и позива методе потребне за рад са мапом.

3.4.3 Рачунање локације

Рачунање локације у оквиру класе *MainActivity* је имплементирано у три функције. Функција *intersect* је само имплементација формуле изведене у поглављу 2.2. Прима два објекта типа *CellTower*, и раздаљине сваке од тих базних станица, од пријемника (корисника). Ова функција чува резултат у пољу *points*, и такође га враћа у виду матрице. У случају ове апликације могућности су да ће функција вратити две тачке пресека, или ни једну. У случају да не постоји ни једна тачка пресека функција враћа матрицу испуњену са *NaN* (енг. *not a number*). Функција *intersectCaller* позива функцију *intersect* за сваку могућу комбинацију раздаљина, за два објекта типа *CellTower*. За сваки од та два објекта који је означен као непознат, односно непрецизан, смањује вредност поља *signalStr* за 1. Овај поступак за резултат има листу тачака, од којих сваки пар (рецимо тачке на нултом и првом индексу), представља тачке пресека два круга. То значи да за сваки пар тачака постоји једна „тачна" и једна „нетачна", и ту листу треба филтрирати и избацити сваку „нетачну тачку".

```
public boolean intersectCaller(CellTower ct1, CellTower ct2){
        Log.d("TAG", "intersectCallerACC: " + ct1.accurate);
        for(int i = 0; i < ct1.distances.size(); i++){
            for(int j = 0; j < ct2.distances.size(); j++){
             this.intersect(ct1, ct2, ct1.distances.get(i), ct2.distances.get(j));
            }
        }
        if(ct1.accurate == false){
            this.signalStr-=1;
        }
        if(ct2.accurate == false){
            this.signalStr-=1;
        }
        this.filter();
        for(Double[] a : this.points){
           if(!Double.isNaN(a[0]) && !Double.isNaN(a[1])){
               return true;
           }
        }
        return false;
    }
```

Код 6 – Функција intersectCaller

Ово филтрирање се ради у односу на локацију трећег *CellTower* објекта, и њиме се бави помоћна функција *minimumDistance* коју позива функција *filter*. Филтрира се по раздаљини од најмањег могућег круга око трећег *CellTower* објекта, то јест из сваког пара тачака се избацује она која је даље од тог круга.

```
for(int i = 1; i < this.points.size(); i+=2){
  if(Math.abs(distanceTwoPoints(this.points.get(i), utmDouble)
   - ct3.distances.get(0)) < Math.abs(distanceTwoPoints(this.points.get(i - 1),
   utmDouble) - ct3.distances.get(0))){
    pointsToRemove.add(this.points.get(i - 1));
}else{
      pointsToRemove.add(this.points.get(i));
            }
        }
        this.points.removeAll(pointsToRemove);
    }
```

Код 7 – Функција minimumDistance

Коначно, функција *intersectCallerWrapper* позива функцију *intersectCaller* за сваки могући пар објеката типа *CellTower*, док не наиђе на пар који производи барем једну валидну тачку. У ситуацијама када имамо везу са више од три базне станице, али само за три од њих постоји тачка пресека, ова функција осигурава да ће апликација вратити пресек између три круга којима одговарају управо те три базне станице.

```
public void intersectCallerWrapper(){
    for(int i = 0; i < this.towers.size(); i++){
       for(int j = 1; j < this.towers.size(); j++){
        if(i != j){
         if(this.intersectCaller(this.towers.get(i), this.towers.get(j)) == true){
                        return;
                    }
                }
            }
        }
        return;
    }
```

Код 8 - Функција intersectCallerWrapper

3.4.4 Позивање функција за трилатерацију

Три методе за трилатерацију, заједно са додатним петљама које добијају потребне податке везане за базне станице, запаковане су у методу *locationSetup*.

```
for (CellInfo ci : cellsTemp) {
            if (ci instanceof CellInfoGsm) {
                ctTemp = new CellTower((CellInfoGsm) ci, res, powers);
                if(ctTemp.getCoords() != null){
                    CellTower c = new CellTower((CellInfoGsm) ci, res, powers);
                    this.towers.add(c);
                    if(c.getLongitude() != 0.0 && c.getLatitude() != 0.0){
                       Log.d("DISTANCE", String.valueOf(c.distanceCalcWrapper()));
                    }
                    if(c.distances.size() == 0){
                        ctsToRemove.add(c);
                    }
                }
                this.cellsLte.add((CellInfoGsm) ci);

            }
        }
        this.towers.removeAll(ctsToRemove);
```

Код 9- Први део функције locationUpdate

Први део функције *locationUpdate* пролази кроз листу објеката типа *CellInfoGSM*, на основу њих конструише објекте типа *CellTower*, и проверава да ли је за све позната локација. За оне за које је позната локација позива се метода за рачунање раздаљине. За случај да после позива ове методе није успешно израчуната ни једна раздаљина, тај објекат се брише из листе *towers*, ма да то није очекивано понашање.

```
if(this.towers.size() >= 2){
            sortByAvgDistance(this.towers);
            this.intersectCallerWrapper();
        }else{
            this.locationUpdate();
            return;
        }
Double[] avg = averageCalc(this.points);
UTM avgMeters = new UTM(34, 'T', avg[0], avg[1]);
```

Код 10- Позив функција за трилатерацију и рачунање просека свих добијених тачака

Функција за трилатерацију се позива само у случају где су познате барем две базне станице. Иначе је за трилатерацију потребно најмање три тачке, али у овој имплементацији могуће је добити просечну тачку са само две базне станице, са много мањом прецизношћу. Ако нису познате барем две базне станице, функција *locationUpdate* се позива рекурзивно.

Ово покрива случајеве где се сигнал према једној станици губи привремено, на део секунде. Коначно, добијена тачка се додаје на просек осталих добијених тачака, и мапа се ажурира.

```
this.numOfAvgs +=1;
if(avgMeters.getEasting()!=0.0 && !Double.isNaN(avgMeters.getEasting()) &&
   avgMeters.getNorthing()!=0.0 && !Double.isNaN(avgMeters.getNorthing())) {
    this.sumAvg[0] += avgMeters.getEasting();
    this.sumAvg[1] += avgMeters.getNorthing();
    UTM pinLocation = new UTM(34, 'T', this.sumAvg[0] / (this.numOfAvgs),
         this.sumAvg[1] / (this.numOfAvgs));
    this.avgCoords = new WGS84(pinLocation);
    if (this.map != null) {
         LatLng avgC = new LatLng(this.avgCoords.getLatitude(),
                             this.avgCoords.getLongitude());
          this.map.clear();
          this.map.addMarker(new MarkerOptions().position(avgC));
            }
        }
```

Код 11 – Ажурирање мапе

3.5 Класа PhoneStateListen

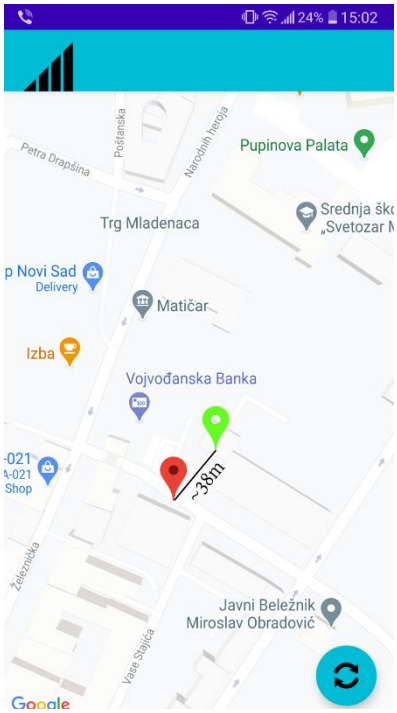
Једини детаљ који је остао недоречен јесте питање када треба позвати методу за ажурирање локације. *Android* има класу *PhoneStateListener*, која има неколико опција за „ослушкивање”. За ову имплементацију, најкориснија од ових опција је *PhoneStateListener.LISTEN_SIGNAL_STRENGTHS*, која служи за ослушкивање промена у јачини сигнала на мобилној мрежи. Ипак, овај *PhoneStateListener* се позива превише често, неколико десетина пута у секунди. Класа *PhoneStateListen* наслеђује класу *PhoneStateListener* и додаје функционалност одбројавања, тако да се локација ажурира на сваки n-ти позив *callback* методе ослушкивача.

```
public void onSignalStrengthsChanged(SignalStrength ss) {
        if(t < 0) {
            this.m.locationUpdate();
            t = 500;
        }else{
            t-=1;
        }
        m.tel.listen(this, PhoneStateListener.LISTEN_SIGNAL_STRENGTHS);
    }
```

Код 12 - Callback metoda klase PhoneStateListen

Callback метода класе *PhoneStateListen* се позива сваки пут када се јачина сигнала према некој базној станици промени. Поље t прати колико пута се то десило, ако вредност поља t падне испод нуле, локација се ажурира. Ово у суштини значи да ће се локација ажурирати само ако се јачина сигнала према некој базној станици променила, али не сваки пут када се то деси. Учесталост ажурирања локације се мења са почетном вредношћу поља t.

3.6 Кориснички интерфејс и примери рада апликације



Слика 4 - Очекивано понашање апликације

Главни део апликације јесте мапа, учитана преко Google maps *api*-а, на којој се налази иконица која представља процену локације корисника. На слици 4 то је црвена иконица. На исту слику додата је зелена иконица која представља праву локацију користника, да би се илустровала прецизност при позиционирању. Грешка у позиционирању за овај пример је приближно 38 метара. Осим мапе пропратни елементи корисничког интерфејса су дугме у доњем десном углу екрана, и индикатор јачине сигнала у горњем левом углу екрана. Индикатор јачине сигнала мери прецизност процене локације која зависи од броја познатих базних станица. За сваку непознату базну станицу коју апликација користи, индикатор се смањује за 1. У примеру са слике 4 све базне станице које се користе су познате и стога је индикатор постављен на највиши ниво. Дугме у доњем десном углу брише сва досадашња мерења из просека и започиње процес трилатерације из почетка, чиме се добија „свежа" локација.

Слика 5 – Понашање апликације са мањком података

На слици 5 је приказан пример рада апликације са само једном познатом базном станицом. У овом случају, трилатерација се и даље може обавити, тиме што се за непознате базне станице узимају такозване *default* вредности, просеци свих других вредности. У овом случају, користе се 2 непознате базне станице, па се индикатор јачине сигнала умањује за 2.

4 Закључак

Резултати, прецизност и покривеност

У првом делу рада било је речи о две комерцијалне примене система за трилатерацију јачином сигнала, код хитних служби, и помоћни систем *Google* мапа[5]. Оба Система тврде да постижу прецизност са грешком између 1600 метара, и 500 метара. Систем *Google* мапа има широку покривеност, а системи хитних служби су углавном ограничени на један град, или чак један део града.

Имплементација у овом раду је наравно рађена са много мање ресурса, података, и за много мање времена од тих примена, и стога не треба очекивати сличан ниво функционалности. Ипак, пошто не постоје имплементације ових система рађене на нижем нивоу, то ће бити једино поређење.

Ова имплементација има грешку у лоцирању која у општем случају није већа од 250 – 300 метара. Најчешће се грешка креће у простору 100 – 200 метара, а може бити чак и око двадесетак метара. Вреди споменути да аномалије на мрежи, као што су привремене драстичне промене у јачини сигнала према једној базној станици, значајно утичу на ову прецизност, као и код сваког система за трилатерацију јачином сигнала. Ово се углавном испољава у виду једне значајно мање прецизне процене локације, која се најчешће губи у рачунању просека. Прецизност такође опада са бројем познатих базних станица. Тачније, ако апликација има податке о све три базне станице које користи, та мерења дају најбољу могућу прецизност. Одатле прецизност опада са сваком базном станицом коју апликација користи, а нема податке о њој. Коначно, ако нема података ни о једној од три базне станице, постоји добра шанса да апликација у тој ситуацији неће успети да израчуна локацију (зато што у пресеку кружница нема тачака). У том случају се узима просечна локација три базне станице, што даје најлошију могућу прецизност[6]. Ови нивои прецизности су графички приказани стандардним индикатором јачине сигнала. У просеку, прецизност је углавном значајно боља од других примена трилатерације јачином сигнала.

[5] Ово је помоћни систем, када нема GPS сигнала

[6] Ово није очекивано понашање апликације, већ је само помоћни систем за ситуације где се на кратко губи сигнал према познатим базним станицама, и постоји да апликација не би падала када се изађе из познатих области.

Једна слабост ове имплементације је покривеност. Не у смислу максималне могуће покривености, покривеност ове апликације се може проширити у теорији и на цео свет, већ у смислу мањка података. У државама где су скупови података мерења јачине сигнала на познатој раздаљини од *2G* базних станица лако доступни, проширење покривености би био тривијалан проблем. Само би требало израчунати модификовану јачину сигнала за сваку базну станицу, а тај задатак би се наравно, аутоматизовао. Нажалост, у нашој држави ти подаци нису доступни. Из тог разлога ова имплементација се ослања на податке скупљене управо за те сврхе, што значајно ограничава покривеност. У суштини, једино што ограничава покривеност и конзистентност рада апликације су кратко време и непоуздани начини на које су ти подаци прикупљени, и покривеност се повећава само прикупљањем веће количине података.

Литература

[1] – Документација *Android-a*

Anrdoid developers. (2021, 02. 24.). *Android documentation*. Android developers: https://developer.android.com/reference/android/telephony/CellInfo

[2] - Библиотека за конвертовање координата

Ansell, P. (2020, 5. 18.). *utm2wgs*. Github:

https://github.com/ansell/utm2wgs

[3] - Чланак о *GPS* позиционирању

Key, H., & Lemmens, M. (2005, 6. 5.). *GPS: Position, time and distance*. GIM International: https://www.gim-international.com/content/article/gps-position-time-and-distance

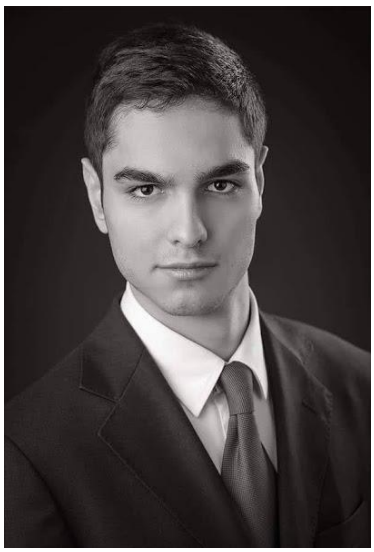
[4] - Извештај FCC-а о системима за трилатерацију јачином сигнала

Tran, M. (2015). *Accurate Location Detection - System and method that allows for cost effective location detection accuracy that exceeds current FCC standards.* Springfield, VA: Federal Communications Commision https://transition.fcc.gov/pshs/911/Apps%20Wrkshp%202015/911_Help_SMS_WhitePaper0515.pdf

Садржај – формуле, делови кода и слике

БИОГРАФИЈА МАТУРАНТА

Марко Кувизић је рођен 17.6.2002. Уписао је Основну школу „Ђорђе Натошевић” у Новом Саду, а завршио је Основну школу при Гимназији „Јован Јовановић Змај”. Гимназију „Јован Јовановић Змај”, смер „Ученици са посебним способностима за рачунарство и информатику” уписао је 2017. године. Почео је да се бави програмирањем током основне школе, али ван наставе. Планира да упише један од смерова сродних рачунарству и програмирању на Факултету техничких наука. Велики је љубитељ шаха, а и других друштвених игара. Осим тога, у слободно време се забавља рачунарским играма.



Фотографија матуранта

Датум предаје матурског рада: _____

Комисија:

Председник _____

Испитивач _____

Члан _____

Коментар:

Датум одбране: _____ Оцена_____ (___)