

5. Geolokácia

Geolokácia je určenie geografickej polohy. Mnohé mobilné zariadenia vedia určiť svoju polohu pomocou geolokačného senzora a/alebo pripojenia na internet. Údaje o geografickej polohe využívajú mobilné aplikácie rôznym spôsobom, napríklad na navigovanie do cieľa, zaznamenanie trasy, označenie fotografie údajom, kde bola zhotovená, hranie exteriérových hier, zdieľanie polohy s priateľmi a podobne. V tejto kapitole uvádzame námety na dva projekty, ktorých cieľom je vytvorenie aplikácie využívajúcej údaje o polohe.

5.1. Reverse caching

Kľúčové slová

GPS, trilaterácia, emulovanie GPS, digitálna mapa, kreslenie do mapy, skupina blokov Any component

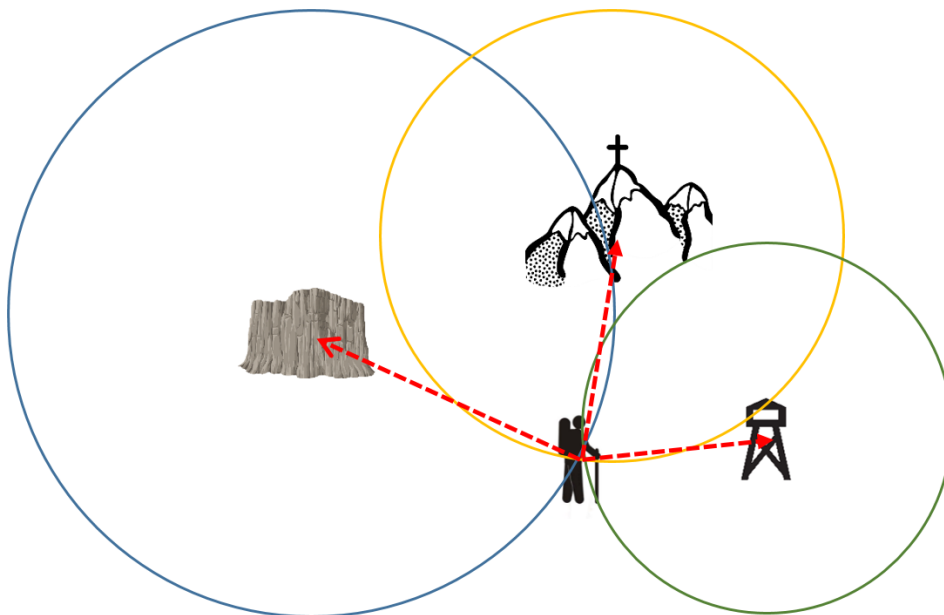
Čo sa naučíme a čo si precvičíme

- princíp GPS – určenie polohy na Zemi pomocou trilaterácie,
- ladiť aplikáciu pomocou emulovania vstupov z geolokačného senzora,
- spracovávať údaje o polohe z geolokačného senzora,
- vypočítať vzdialenosť aktuálnej polohy od danej geografickej polohy,
- zobrazovať v aplikácii svoju polohu na mape a kresliť do mapy,
- používať bloky zo skupiny Any component.

Čo zaujímavé môžeme zistiť?

Predstavme si človeka strateného niekde v horách, ktorý vysielaczkou nadviazať spojenie so záchranármi. Zo svojej pozície vidí tri výrazné objekty: rozhľadňu, vrchol kopca s krížom a výraznú skalu. Záchranárom nahlási odhadom vzdialenosť od týchto troch miest. Ako ho záchranári nájdu?

Záchranári si na mape nájdu tri spomínané objekty. Stratený turista sa bude nachádzať niekde na kružniciach okolo objektov v danej vzdialenosti. Ak turista záchranárom správne odhadne svoju vzdialenosť od troch objektov, kružnice sa pretnú v jednom bode, na ktorom sa nachádza. Tento princíp určovania polohy sa volá *trilaterácia* (obr. 5.1.1). Ak turista neudá vzdialenosti celkom presne, kružnice sa nepretnú v jednom bode, ale aspoň bližšie vymedzia priestor, na prehľadávanie ktorého sa záchranári zamerajú.

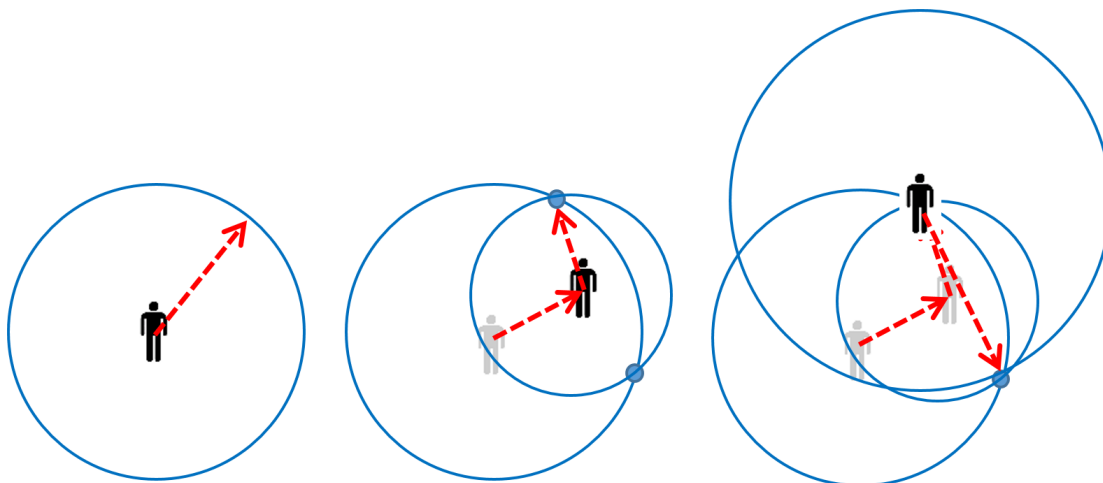


Obr. 5.1.1 Princíp trilaterácie

Podobne funguje princíp určovania polohy na Zemi pomocou satelitov. Pomocou mobilného zariadenia s prijímačom signálu satelitnej navigácie, napr. GPS (Geographical Positioning System), náš prístroj zameria niekoľko satelitov a odmeria svoju vzdialenosť od nich. Na základe vzdialeností od aspoň troch satelitov (aspoň štyroch aj pre určenie nadmorskej výšky) vie určiť svoju polohu na Zemi.

Akú zaujímavú aplikáciu môžeme vytvoriť?

Mobilné aplikácie na navigáciu do cieľa nám ukazujú smer, ktorým sa nachádza náš cieľ, a vzdialenosť, ktorá nás od neho delí. Z princípu trilaterácie vieme, že aj bez udania smeru vieme nájsť cieľ na tri merania vzdialenosti (obr. 5.1.2).



Obr. 5.1.2 Hľadanie cieľa meraním vzdialenosti

My si urobíme aplikáciu, ktorá nám umožní zahrať sa exteriérovú hru „reverse caching“, pri ktorej sa hľadá miesto s ukrytým „pokladom“ na základe informácie o vzdialenosti k cieľu bez

udania smeru. Jednoduchá verzia tejto hry je známa ako „Zima–teplo“, v ktorej sa neudáva konkrétna vzdialenosť, len relatívna blízkosť k cieľu. Naša aplikácia:

- bude určovať našu polohu a zobrazovať ju výpisom súradníc a graficky na mape,
- umožní zadať a uložiť do pamäte súradnice cieľa,
- na požiadanie (stlačenie tlačidla) vypočíta vzdialenosť k cieľu,
- vypíše vzdialenosť a farebne odliší, či sme ďaleko, blízko, alebo veľmi blízko k cieľu,
- na mape vyznačí okolo aktuálnej pozície kružnicu, na ktorej sa cieľ môže nachádzať,
- bude počítat počet meraní vzdialenosti.

Ako budeme postupovať pri tvorbe aplikácie?

V Etude 2.9 sme sa zoznámili s blokmi na prácu s údajmi z geolokačného senzora a na zobrazovanie geolokačných údajov v digitálnej mape. Použili sme a aj v tomto projekte použijeme tieto prvky:

- Komponenty:
 - LocationSensor
 - Map
- Udalosti:
 - LocationSensor.Changed
- Vlastnosti:
 - LocationSensor.Latitude
 - LocationSensor.Longitude
 - LocationSensor.DistanceInterval
 - LocationSensor.TimeInterval
 - Map.LocationSensor,
 - Map.ShowUser
 - Map.EnableZoom
 - Map.ZoomLevel
 - Map.EnablePan

Pri ladení a testovaní aplikácií, ktoré spracovávajú údaje o polohe zariadenia, musíme zabezpečiť vstup týchto údajov buď premiestnením sa v teréne, čo je nepraktické, alebo emulovaním (napodobňovaním) zmeny polohy softvérovo. Počas ladenia aplikácie je pohodlnejšie emulovanie zmeny polohy bez presúvania sa v teréne.

Vysvetlíme si

Emulovanie (napodobňovanie) určovania polohy je nahradenie údajov o skutočnej polohe zariadenia falošnými údajmi, ktoré sú generované softvérovo. V aplikačnom obchode nájdeme aplikácie, ktoré emulujú zmenu polohy zariadenia, zadáním kľúčového slova *GPS Emulator*. Emulátor napodobňuje fyzickú zmenu polohy zariadenia vyznačením polohy na mape. Používanie emulovaných údajov v aplikáciách namiesto skutočných údajov o polohe povolíme v nastaveniach operačného systému v možnostiach pre vývojára.

Úloha 1

Otestujte aplikáciu Kde som? z etudy 2.9 pomocou emulátora GPS vstupov.

Úloha 2

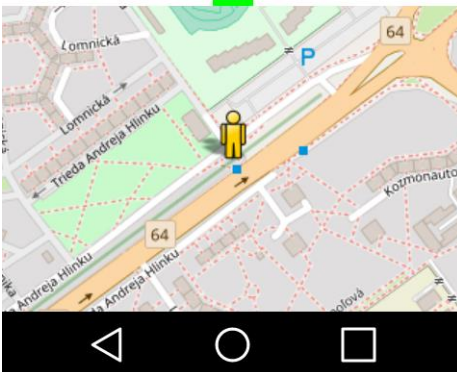
Vytvorte používateľské rozhranie aplikácie Reverse caching podľa obr. 5.1.3. Na obrazovke sa majú zobrazovať:

- aktuálne geografické súradnice označené príslušnými popismi,
- editovacie polia na zadanie súradníc cieľa označené popismi,
- tlačidlo s nápisom *Prevezmi* na prevzatie aktuálnych súradníc do cieľových súradníc (vyplní polia súradníc cieľa súradnicami aktuálnej polohy),
- tlačidlo s nápisom *Ulož* na uloženie cieľových súradníc do premenných,
- tlačidlo s nápisom *Meranie vzdialenosti (v metroch)* na výpočet vzdialenosti,
- veľký farebný nápis na zobrazenie vzdialenosti,
- mapa so zobrazením pozície používateľa.

Reverse caching			
Pozícia	Prevezmi	Cieľ	Ulož
Lat	48.31338	Lat	48.31338
Lon	18.10345	Lon	18.10345

Meranie vzdialenosti (v metroch)

0



Obr. 5.1.3 Dizajn aplikácie

Úloha 3

Naprogramujte zobrazovanie súradníc aktuálnej polohy a funkcie tlačidiel `Button_Prevezmi` a `Button_Ulož`. Aplikáciu testujte so vstupmi z emulátora. Sledujte, ako sa mení poloha používateľa na mape.

Úloha 4

Naprogramujte výpočet vzdialenosti od aktuálnej pozície k cieľu a výsledok zobrazte po stlačení tlačidla *Meranie vzdialenosti* v nápisu pod tlačidlom s farebným pozadím:

- zeleným, ak je vzdialenosť menšia ako 10 m,
- žltým, ak je vzdialenosť menšia ako 50 m a nie menšia ako 10 m,
- červeným, ak je vzdialenosť 50 a viac metrov.

Vypočítať najkratšiu vzdušnú vzdialenosť medzi dvomi bodmi na Zemi určenými geografickými súradnicami znamená určiť dĺžku oblúka na povrchu gule sploštenej na póloch. Nasledujúci vzorec počíta dĺžku oblúka na guli bez sploštenia, teda s určitou chybou:

$$d = \arccos(\sin \varphi_1 \cdot \sin \varphi_2 + \cos \varphi_1 \cdot \cos \varphi_2 \cdot \cos(\lambda_2 - \lambda_1)) \cdot R$$

φ_1 , φ_2 sú geografické šírky aktuálnej pozície a cieľa (latitude), λ_1 , λ_2 sú ich geografické dĺžky (longitude), R je polomer Zeme (asi 6371 km).¹

Pri malých vzdialenostiach chyba výpočtu nie je veľká a môžeme ju zanedbať. Oveľa väčšiu chybu spôsobuje nepresnosť vstupných údajov z lokalizácie.

Vysvetlíme si

Jednotkou veľkosti uhla je *radián*. V praxi sa však častejšie na meranie uhlov používajú *stupne*. Aj geografické súradnice sa udávajú v stupňoch.

V App Inventore sa používa ako jednotka uhla stupeň. To znamená, že všetky funkcie, ktorých parametrom sú uhly, alebo vracajú ako výsledok uhol, počítajú s údajmi v stupňoch. Bloky na prácu s uhlami nájdeme v skupine Math:

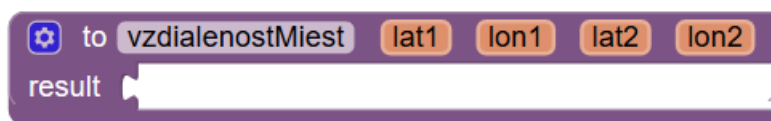
Bloky pre goniometrické funkcie `sin`, `cos`, `tan` majú vstupný parameter uhol v stupňoch a vracajú číslo.

Ich inverzné funkcie `asin`, `acos`, `atan` majú vstupný parameter číslo a vracajú uhol v stupňoch.

Na prevod stupňov na radiány a naopak slúži funkcia `convert radians to degrees`, `convert degrees to radians`.

V matematickom vzorci na výpočet vzdialenosti bodov na povrchu gule sú všetky uhly v radiánoch. Funkcie `sin`, `cos` však v App Inventore očakávajú vstupný uhol v stupňoch. To je výhodné, pretože nemusíme geografické šírky a dĺžky prevádzať v App Inventore na radiány. Výsledkom funkcie `acos` je v App Inventore tiež uhol v stupňoch. Aby sme ho mohli použiť vo vzorci na výpočet vzdialenosti dvoch bodov, musíme ho previesť na radiány.

Na sprehľadnenie kódu pri programovaní vzorca na výpočet vzdialenosti dvoch miest je užitočné naprogramovať výpočet ako funkciu so štyrmi parametrami (súradnice dvoch miest) a s návratovou hodnotou (vzdialenosťou vypočítanou podľa vzorca). Blok s hlavičkou funkcie:



¹ <https://www.movable-type.co.uk/scripts/latlong.html>

Úloha 5

Do mapy zakreslite kružnicu so stredom v aktuálnej polohe používateľa s polomerom vzdialenosti od cieľa.

Na kreslenie kružnice do mapy so stredom v daných geografických súradniciach a s polomerom v metroch použijeme komponent `Circle`. Komponent sa nachádza v skupine `Maps`. V režime návrhu umiestnime kruh `Circle1` kdekoľvek do komponentu `Map1`, nastavíme farbu výplne na `None` (žiadna), zvolíme farbu a hrúbku obrysu – zostane kružnica. Kružnicu zatiaľ ponecháme neviditeľnou, zobrazovať ju budeme až počas behu programu. Nastavenie aktuálneho stredu a polomeru a následné zobrazenie kružnice naprogramujeme v reakcii na stlačenie tlačidla na výpočet vzdialenosti.

Vysvetlíme si

Vybrané vlastnosti komponentu `Circle`:

`Latitude` – geografická šírka stredu kruhu

`Longitude` – geografická dĺžka stredu kruhu

`Radius` – polomer kruhu v metroch

`FillColor` – farba výplne kruhu, 0 pre žiadnu farbu

`StrokeColor` – farba obrysu kruhu

`StrokeWidth` – hrúbka obrysu kruhu

`Visible` – viditeľnosť (true alebo false)

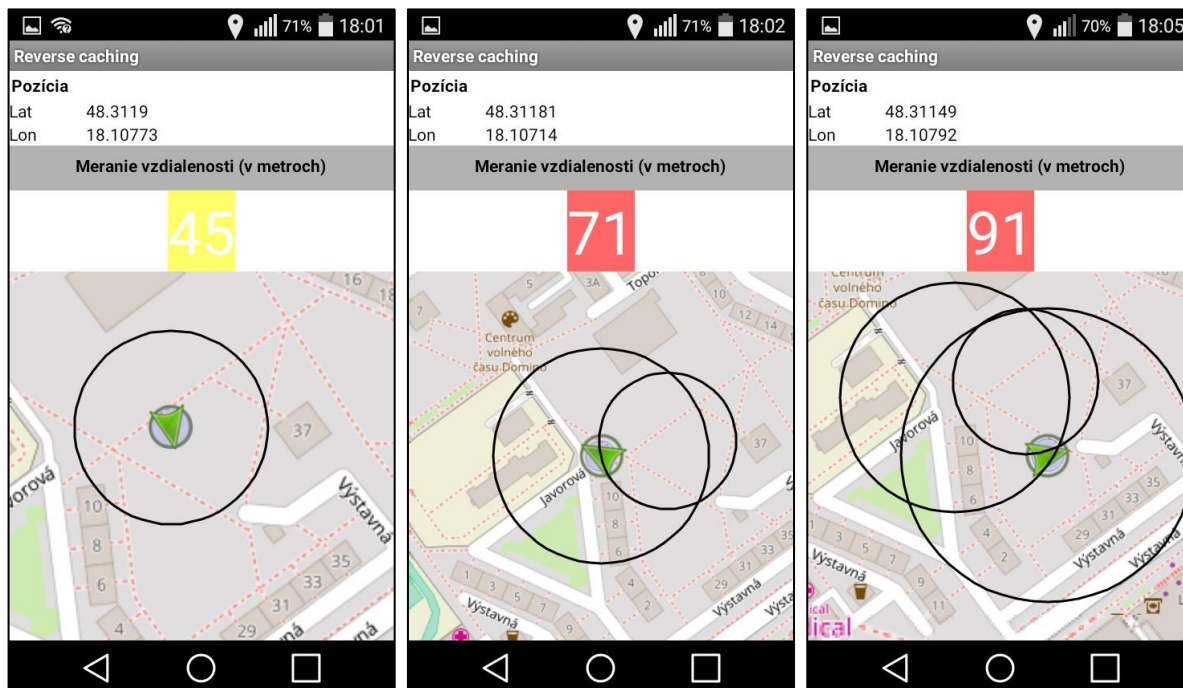
Metóda:

`SetLocation(number latitude, number longitude)` – nastaví pozíciu stredu kruhu na dané súradnice

Kreslenie kružnice do mapy nám pomáha pri hľadaní cieľa metódou trilaterácie. Na jednoznačné určenie polohy cieľa však potrebujeme odmerať vzdialenosť z aspoň troch rôznych miest a nakresliť aspoň tri kružnice. Do projektu preto pridajme viac komponentov `Circle`. Komponenty sa nedajú v App Inventore vytvárať počas behu programu, musíme si ich vložiť do projektu vopred v režime návrhu. Počet kružníc teda musíme obmedziť na konkrétny počet (aspoň 3, ale lepšie je pridať viac). Na začiatku ich zneviditeľníme.

Úloha 6

Pridajte do projektu počítadlo meraní vzdialenosti a po každom meraní zakreslite do mapy novú kružnicu (obr. 5.1.4). Počet meraní obmedzte a po dosiahnutí maxima neumožnite ďalšie merania.



Obr. 5.1.4 Trilaterácia – po treťom meraní je cieľ síce ďaleko, ale jeho poloha je už známa

Pre počítanie počtu meraní vzdialenosti vytvorme globálnu premennú `cisloMerania`, ktorú budeme pri každom meraní zvyšovať o 1 a ktorá určuje poradové číslo merania. Určme si maximálny počet meraní (napr. 5) a po jeho dosiahnutí nastavíme vlastnosť `Enabled` tlačidla na meranie na `false`, čím znemožníme ďalšie merania.

Do mapy vložme 5 komponentov `Circle1`, `Circle2`, `Circle3`, `Circle4`, `Circle5`. Bolo by nepraktické pre každý z nich písať rovnaký kód pri inicializácii ich vlastností alebo zobrazovaní na mape. Namiesto toho napíšme všeobecný kód pre ľubovoľný komponent z triedy `Circle` (Any Circle – nejaký kruh), v ktorom špecifikujeme adresáta (konkrétny kruh) až počas behu programu.

Vysvetlíme si

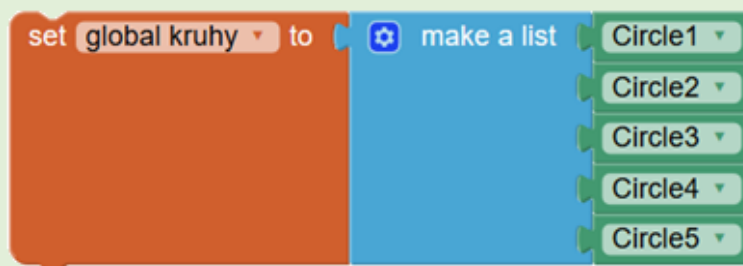
Bloky zo skupiny *Any component* (napr. Any Button, Any Label, Any Circle atď.) sú podobné, ako bloky konkrétnych komponentov s tým rozdielom, že umožňujú určiť adresáta (konkrétny komponent) až počas behu programu. Napríklad:

Blok pre komponent `Circle1`: a pre Any Circle:



Ak máme viac komponentov z rovnakej triedy, uložíme si ich do zoznamu (list). V cykle `for each item in list` vykonáme rovnaký kód zapísaný pomocou blokov zo skupiny *Any component* pre každý komponent zo zoznamu.

Napríklad vytvoríme zoznam kruhy komponentov Circle1 až Circle5:



Nastavenie vlastnosti Visible postupne vo všetkých komponentoch Circle v zozname kruhy vykonáme v cykle:



Nastavenie vlastnosti Visible jedného komponentu, ktorý sa nachádza v zozname kruhy na indexe uloženom v premennej cisloMerania, urobíme takto:



Rovnaké vlastnosti pre každý kruh (Visible, FillColor, StrokeWidth) nastavíme v cykle pri inicializácii aplikácie. Pri kreslení kružnice do mapy upravíme pozíciu (metódou SetLocation), polomer (Radius) a viditeľnosť (Visible) toho kruhu, ktorý je v zozname kruhov na indexe zodpovedajúcom číslu merania.

Úloha 7

Otestujte aplikáciu v teréne so skutočnými vstupmi zo senzora GPS.

Ako vylepšiť či rozšíriť našu aplikáciu?

Tu je niekoľko námetov na ďalšie rozšírenia aplikácie:

- Centrovať mapu podľa polohy používateľa (s použitím metódy Map.PanTo).
- Vložiť do mapy značky na miesta meraní vzdialenosti od cieľa (komponent Marker) v stredoch kružníc.
- Spojiť v mape lomenou čiarou miesta meraní (komponent LineString).
- Pri priblížení sa k cieľu na veľmi malú vzdialenosť (napr. menšiu ako 10 m) automaticky vypísať oznam.
- Po piatich meraniach vypísať správu o ukončení meraní.

- Prerobiť dizajn na aplikáciu s dvomi obrazovkami – s prvou na zadávanie a uloženie súradníc cieľa, s druhou na hľadanie cieľa pomocou určovania vzdialenosti a zobrazovania kružníc na mape.
- Pridať tlačidlo na ukončenie aplikácie.

Ako sa zahrať s našou aplikáciou?

Reverse caching je hra, v ktorej sa hľadá neznáme miesto na Zemi (môže byť na ňom niečo ukryté) pomocou určovania vzdialenosti. S našou aplikáciou sa takúto hru môžeme zahrať.

V prvej fáze hry označíme cieľové miesto v teréne nejakou nenápadnou, ale viditeľnou značkou (napr. nálepkou) alebo na ňom niečo ukryjeme. Súradnice cieľa uložíme v našej aplikácii.

V druhej fáze odovzdáme mobilné zariadenie s našou aplikáciou niekomu, kto bude náš označený cieľ hľadať. Na nájdenie cieľa má najviac 5 pokusov s meraním vzdialenosti.

Čo sme sa naučili

- ✓ Použiť emulátor na generovanie falošnej polohy mobilného zariadenia,
- ✓ počítať vzdialenosť dvoch bodov určených geografickými súradnicami,
- ✓ zobrazovať v aplikácii polohu na mape a kresliť do mapy,
- ✓ vytvárať hromadné algoritmy pre komponenty pomocou blokov zo skupiny Any Component.

Bibliografia

Michaličková, V. (2016). *Programovanie mobilných aplikácií v prostredí MIT App Inventor 2*. Nitra: Univerzita Konštatína Filozofa v Nitre.

MIT. (2018). . Retrieved from: <http://ai2.appinventor.mit.edu/>

Wolber, D., Abelson, H., Spertus, E., & Liz, L. (2014). *App Inventor 2 - Create Your Own Android Apps*. O'Reilly.

Register pojmov

A

Any component, 7

E

emulovanie, 3

F

funkcia, 5

L

list, 7

LocationSensor, 3

M

Map, 3

R

radián, 5

reverse caching, 2

Z

T

zoznam, 7

trilaterácia, 1