



PLÁN [OBNOVY]

Nitrianske kompetenčné centrum kybernetickej bezpečnosti
Fakulta prírodných vied a informatiky
Univerzita Konštantína Filozofa v Nitre
Tr. A. Hlinku 1, 949 01 Nitra

OVLÁDANIE SERVO MOTORA CEZ WEBOVÝ SERVER S VYUŽITÍM FIREBASE DATABÁZY

Predmet: PMS/22 – Programovanie mikroprocesorových
systémov

Téma

- inštalácia a nastavenie Firebase
- hardvérové zapojenie Arduino a servomotor
 - bezpečnostné hrozby a ich eliminácie

Martin Magdin
mmagdin@ukf.sk

ZAPOJENIE A OVLÁDANIE SERVO MOTORA CEZ WEBOVÝ SERVER S VYUŽITÍM FIREBASE	
DATABÁZY	4
NASTAVENIE DATABÁZY	5
ARDUINO KÓD	6
SERVER KÓD	8
AKO VYTVORIŤ INDEX.HTML	8
V ČOM SÚ SKRYTÉ HROZBY?	10
HROZBA Č. 1 – NEZABEZPEČENÉ FIREBASE PRAVIDLÁ	10
HROZBA Č. 2 - KLIENTSKE API KĹÚČE ULOŽENÉ V KÓDE	11
HROZBA Č. 3 - NEDOSTATOČNÉ OBMEDZENIE FREKVENCIE DOTAZOV (RATE LIMITING)	11
HROZBA Č. 4 - MOŽNOSŤ VZDIALENÉHO VYKONANIA ŠKODLIVÝCH OPERÁCIÍ	12
PRÍKLAD SKRIPTU PRE DOS ÚTOK (IBA NA VÝUČBOVÉ ÚČELY)	12

Abstrakt

O čom je daný materiál – do 100 slov.

Autori

Autor1: Martin Magdin

Email1: mmagdin@ukf.sk

Univerzita: Fakulta prírodných vied a informatiky, Univerzita Konštantína Filozofa v Nitre

Cieľová skupina

Študenti bakalárskeho štúdia odboru Informatika

Vzdelávacie ciele

- Účastníkom osvojované vedomosti a zručnosti
- Účastníkom rozvíjané spôsobilosti

Východiskové predpoklady a požiadavky na účastníkov

- V kontexte predmetu aké témy už musia mať absolvované aby obsah zvládli

Použité metódy

- Prednáška, diskusia, práca v skupinách, simulácia útokov.

Požiadavky

Prístrojové vybavenie

- počítače, laboratórne priestory / učebňa pre realizáciu aktivít v súvislosti s potrebným hardvérom

Priestorové požiadavky

- Učebňa

Veľkosť skupiny

- Min 10, max. 20 študentov

Rozsah a formát

- Trvanie: 3 hodiny
- Forma: prezenčne

Zapojenie a ovládanie servo motora cez webový server s využitím Firebase databázy

Princípom tohto zapojenia je umožniť používateľovi ovládať servo motor na diaľku cez internet pomocou platformy Firebase Realtime Database a dosky Arduino UNO WiFi Rev 2. Servo motor je pripojený na digitálny výstup Arduina a jeho uhol sa automaticky aktualizuje podľa hodnoty uloženej v databáze. Webové rozhranie slúži ako ovládací panel, kde si používateľ môže zvoliť požadovaný uhol natočenia serva, ktorý sa v reálnom čase prenesie do Firebase a následne vykoná fyzické natočenie motora. Projekt predstavuje praktickú ukážku IoT riešenia s obojsmernou komunikáciou medzi hardvérom a webovým používateľským rozhraním. Štandardne by sme pri tak jednoduchom zapojení neočakávali žiadne problémy. Ak však zanedbáme základné bezpečnostné opatrenia, môžeme sa stretnúť s neočakávanými bezpečnostnými rizikami a útokmi. V nasledujúcom príklade si ukážeme ako a kde môžu takéto bezpečnostné diery vzniknúť.



Úloha:

Vytvorte podľa schémy zapojenia hardvérové prepojenie mikrokontroléra Arduino a servo motora. Na ovládanie servo motora použite kombináciu kódu v Arduino IDE a databázu Firebase, v ktorej sa bude aktualizovať hodnota uhlu natočenia servo motora.

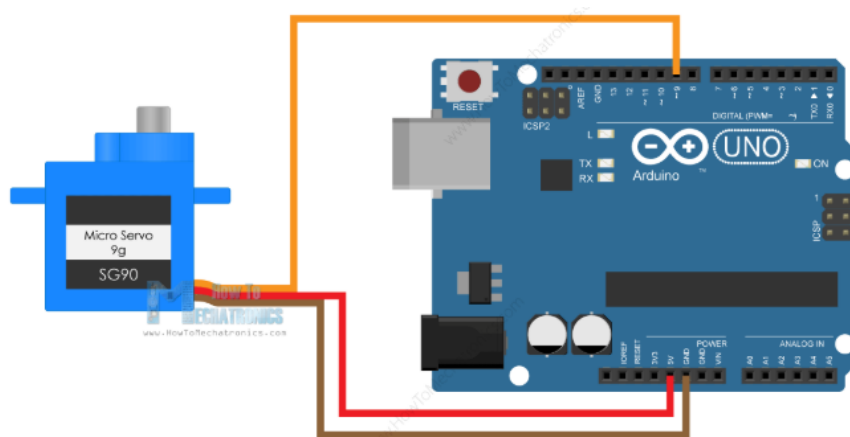
Hardvérové zapojenie

- 1× Arduino UNO WiFi Rev 2
- 1× Servo motor Micro Servo 9g SG90
- 3× Prepojovacie vodiče (male–male alebo male–female)
- 1× Breadboard (voliteľné)

Zapojenie servo motora:

Servo pin	Pripojenie na Arduino	Popis
VCC (červený)	5V (alebo externé 5–6 V)	Napájanie serva
GND (hnedý/čierny)	GND	Zem
SIGNAL (oranžový/žltý)	Digitálny pin D9	Riadiaci signál (PWM výstup)

Príklad zapojenia:



Inštalácia knižníc

V Arduino IDE je potrebné nainštalovať tieto knižnice (cez **Library Manager**):

1. **Servo.h**: Knižnica Servo v Arduino IDE slúži na ovládanie servomotorov, čo sú malé motory s vysokou presnosťou, ktoré sa dajú nastaviť na konkrétny uhol (typicky v rozsahu 0° až 180°).
2. **WiFiNINA**: Knižnica pre prácu s Wi-Fi modulom v doskách ako Arduino Uno WiFi Rev 2, umožňuje pripojenie k internetu.

Firebase_Arduino_WiFiNINA: Zabezpečuje komunikáciu s Firebase

Nastavenie databázy

Je potrebné upraviť bezpečnostné pravidlá v službe Firebase Realtime Database preto, aby Vaša Arduino doska mohla čítať a zapisovať údaje do databázy. V predvolenom nastavení je prístup obmedzený, aby bola databáza chránená pred neoprávneným prístupom.

Ak chcete počas vývoja jednoducho testovať funkčnosť komunikácie medzi webovým rozhraním, Firebase a hardvérom, odporúča sa dočasne povoliť úplný prístup (`.read` a `.write` nastaviť na `true`). Tým umožníte, aby Vaše zariadenie aj webová stránka mohli bez obmedzení pracovať s dátami v databáze.

Postup na úpravu bezpečnostných pravidiel v Firebase RTDB

Firebase Console

Prejdi na: <https://console.firebase.google.com>

Vyber svoj projekt.

Chod' do časti „Realtime Database“

V ľavom menu klikni na „Build“ → „Realtime Database“.

Vyber svoju databázu.

Klikni na kartu „Rules“ (Pravidlá)

Tá sa nachádza hore – vedľa „Data“ a „Usage“.

Uprav pravidlá podľa potreby

Príklad:

```
{
  "rules": {
    ".read": true,
    ".write": true
  }
}
```

Po dokončení vývoja je však veľmi dôležité tieto pravidlá sprísniť, aby nedošlo k zneužitiu alebo poškodeniu dát.

Arduino kód

Arduino kód slúži na ovládanie servo motora na základe údajov uložených vo Firebase Realtime Database. Po pripojení na Wi-Fi a nadviazaní spojenia s databázou Arduino každých 200 ms kontroluje hodnotu v uzle servo/angle. Ak zistí zmenu, nastaví servo motor na nový uhol v rozsahu 0–180°. Zároveň pri spustení zapíše počiatočný uhol (0°) do databázy. Kód využíva knižnice Servo.h, WiFinINA a Firebase_Arduino_WiFinINA.

JE POTREBNÉ VYPLNIŤ ÚDAJE:

- **WIFI_SSID:** názov Vašej WiFi siete
- **WIFI_PASS:** heslo Vašej WiFi siete
- **DATABASE_HOST:** tvoj-projekt-default-rtdb.firebaseio.com (bez https://)

DATABASE_SECR:

"Settings -> Project settings -> Service accounts -> Database secrets -> Secrets"

```
#include <Servo.h>
#include <WiFinINA.h>
#include <Firebase_Arduino_WiFinINA.h>

#define WIFI_SSID      "YOUR_SSID"
#define WIFI_PASS      "YOUR_PASS"
#define DATABASE_HOST  "your-project-id.firebaseio.com"    // bez
https://
#define DATABASE_SECR  "Settings -> Project settings -> Service accounts ->
Database secrets -> Secrets"

#define SERVO_PIN      9
#define POLL_MS        200

Servo      myServo;
FirebaseData fbdo;

int currentAngle = 0; // Počiatočný uhol 0°
bool isInitialized = false; // Príznak inicializácie
uint32_t lastPoll = 0;

void setup() {
```

```

Serial.begin(9600);

/* Servo inicializácia */
myServo.attach(SERVO_PIN);
myServo.write(currentAngle);

/* Wi-Fi pripojenie */
Serial.print("Pripájam sa na Wi-Fi");
WiFi.begin(WIFI_SSID, WIFI_PASS);
while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(300);
}
Serial.println(" OK");

/* Firebase pripojenie */
Firebase.begin(DATABASE_HOST, DATABASE_SECR, WIFI_SSID, WIFI_PASS);
Firebase.reconnectWiFi(true);

/* Počiatočný záznam do Firebase */
if (Firebase.setInt(fbdo, "servo/angle", currentAngle)) {
    Serial.println("Počiatočný uhol 0° uložený do Firebase");
} else {
    Serial.print("Chyba pri vytváraní počiatočnej hodnoty: ");
    Serial.println(fbdo.errorReason());
}

Serial.println("Inicializácia hotová.");
}

void loop() {
    /* Pravidelné čítanie z Firebase */
    if (millis() - lastPoll >= POLL_MS) {
        lastPoll = millis();

        if (Firebase.getInt(fbdo, "servo/angle")) {
            int target = fbdo.intData();

            /* Kontrola rozsahu */
            if (target < 0) target = 0;
            if (target > 180) target = 180;

            if (target != currentAngle) {
                currentAngle = target;
                myServo.write(currentAngle);

                Serial.print("Servo: ");
                Serial.print(currentAngle);
                Serial.println("°");
            }
        }
        else {
            Serial.print("Firebase get ERR: ");
            Serial.println(fbdo.errorReason());
        }
    }
}

```


Server kód

Súbor `index.html` slúži ako webové ovládacie rozhranie pre servo motor, ktorý je pripojený k Arduino. Po načítaní stránky sa webová aplikácia pripojí k Firebase Realtime Database a zobrazuje aktuálny uhol serva. Používateľ môže jednoducho upraviť uhol pomocou tlačidiel $\pm 10^\circ$ alebo zadáním konkrétnej hodnoty (0–180°), pričom zmeny sa okamžite zapíšu do databázy. Arduino tieto zmeny priebežne sleduje a podľa nich nastavuje polohu serva.

Ako vytvoriť `index.html`

1. Otvorte si **textový editor** (napr. Notepad, VS Code, Sublime Text).
2. Skopírujte doň HTML kód (nachádza sa nižšie).
3. Nahradte vo Firebase konfigurácii v časti `initializeApp` vlastné údaje projektu.

`apiKey`: "Settings -> Project settings -> Service accounts -> Database secrets -> Secrets",
`databaseURL`: "<https://name-of-yourproject-rtdb.firebaseio.com>"

4. Súbor uložte pod názvom: `index.html`
5. Dvojklikom otvorte súbor v prehliadači.

Webová stránka bude fungovať lokálne a komunikovať priamo s Firebase databázou – bez potreby servera.

```
<!doctype html>
<html lang="sk">
<meta charset="utf-8" />
<title>Servo dashboard</title>

<style>
  body{font-family:sans-serif;text-align:center;margin-top:4rem}
  #angle{font-size:4rem;margin:1rem 0}
  button{font-size:1.5rem;padding:.6rem 2rem;margin:.5rem;cursor:pointer}
  .angle-controls {margin: 1rem auto; display: flex; justify-content:
center; align-items: center;}
  #angleInput {font-size: 1.5rem; width: 5rem; text-align: center; padding:
.5rem; margin: 0 .5rem;}
  #setAngleBtn {font-size: 1.5rem; padding: .5rem 1rem; cursor: pointer;}
</style>

<!-- Firebase SDK v11 - modulový import -->
<script type="module">
  import { initializeApp }           from
    "https://www.gstatic.com/firebasejs/11.6.0/firebase-app.js";
  import { getDatabase, ref, onValue,
           runTransaction             } from
    "https://www.gstatic.com/firebasejs/11.6.0/firebase-database.js";

  /* Firebase config */
  const app = initializeApp({
    apiKey:      "Settings -> Project settings -> Service accounts ->
Database secrets -> Secrets",
    databaseURL: "https://name-of-yourproject-rtdb.firebaseio.com",
  });
  const db    = getDatabase(app);
```

```

const angleRef = ref(db, "servo/angle");

/* UI prvky */
const angleEl = document.getElementById("angle");
const btnUp = document.getElementById("btnUp");
const btnDown = document.getElementById("btnDown");
const angleInput = document.getElementById("angleInput");
const setAngleBtn = document.getElementById("setAngleBtn");

/* Live zobrazenie hodnoty */
onValue(angleRef, snap => {
  const val = snap.val();
  angleEl.textContent = (val === null) ? "-" : `${val}°`;
  angleInput.value = (val === null) ? "" : val;
});

/* Bezpečná zmena o  $\pm 10^\circ$  (0-180) */
function changeAngle(delta) {
  runTransaction(angleRef, current => {
    if (current === null || isNaN(current)) current = 90; // default
    let next = current + delta;
    if (next < 0) next = 0;
    if (next > 180) next = 180;
    return next;
  }).catch(err => console.error("Firebase TX error:", err));
}

/* Nastavenie konkrétneho uhla */
function setSpecificAngle() {
  let newAngle = parseInt(angleInput.value);
  if (isNaN(newAngle)) {
    alert("Zadajte platný uhol (0-180)");
    return;
  }

  newAngle = Math.max(0, Math.min(180, newAngle));

  runTransaction(angleRef, () => {
    return newAngle;
  }).catch(err => console.error("Firebase TX error:", err));
}

btnUp.addEventListener("click", () => changeAngle(+10));
btnDown.addEventListener("click", () => changeAngle(-10));
setAngleBtn.addEventListener("click", setSpecificAngle);

// Povolenie nastavenia uhla aj stlačením klávesy Enter
angleInput.addEventListener("keypress", (e) => {
  if (e.key === "Enter") {
    setSpecificAngle();
  }
});
});
</script>

<body>
  <h1>Servo dashboard</h1>
  <div id="angle"></div>
  <button id="btnDown">-10°</button>
  <button id="btnUp">+10°</button>

  <div class="angle-controls">

```

```

<input type="number" id="angleInput" min="0" max="180" placeholder="0-180">
<button id="setAngleBtn">Nastaviť uhol (0 - 180)</button>
</div>
</body>
</html>

```

V čom sú skryté hrozby?

Hrozba č. 1 – Nezabezpečené Firebase pravidlá

V dokumente sa odporúča nastaviť Firebase pravidlá na:

```

{
  "rules": {
    ".read": true,
    ".write": true
  }
}

```

Toto umožňuje **neobmedzený čítací aj zápisový prístup** pre kohokoľvek, kto pozná adresu databázy – **bez autentifikácie**.

Možný útok:

- **DoS útok:** útočník môže nepretržite prepísať hodnotu servo/angle tisíckrát za sekundu (napr. pomocou skriptu), čím zaťaží Firebase a Arduino, ktoré každých 200 ms číta novú hodnotu.
- **Data poisoning:** zapisovanie nesprávnych alebo škodlivých hodnôt (aj mimo rozsahu), čo môže viesť k nesprávnemu chodu zariadenia.
- **Zamrznutie systému:** prepínanie medzi hodnotami (napr. $0^\circ \leftrightarrow 180^\circ$) vo vysokej frekvencii môže motor mechanicky poškodiť alebo zničiť napájanie.

Riešenie:

- Po vývoji **okamžite upraviť Firebase pravidlá**
- Implementovať **autentifikáciu Firebase Auth** (napr. email/password, token)

```

{
  "rules": {
    "servo": {
      "angle": {
        // Povolený prístup len prihláseným používateľom
        ".read": "auth != null",
        ".write": "auth != null && newData.isNumber() && newData.val() >= 0
&& newData.val() <= 180",
        // Detekcia rýchleho zápisu (voliteľné)

```

```

        ".validate": "newData.val() != data.val() && (newData.val() -
data.val()).abs() <= 90"
    },
    "logs": {
        // Povolný zápis logov len prihláseným
        ".write": "auth != null",
        ".read": "auth != null &&
root.child('users').child(auth.uid).child('isAdmin').val() == true"
    }
}
}

```

Vysvetlenie:

- `auth != null`: Iba prihlásený používateľ môže čítať/zapisovať.
- `newData.isNumber() && newData.val() >= 0 && newData.val() <= 180`: Hodnota uhla musí byť číselná a v rozsahu.
- `newData.val() != data.val()`: Zakáže zbytočné opakované zápisy tej istej hodnoty.
- `(newData.val() - data.val()).abs() <= 90`: Voliteľná ochrana pred náhlymi skokmi – zakáže zmeny >90° naraz.

Ak existuje viacero zariadení alebo používateľov, je možné ich ďalej špecifikovať podľa UID alebo „device ID“.

Hrozba č. 2 - Klientske API kľúče uložené v kóde

V HTML a Arduino kóde sú citlivé údaje ako `apiKey`, `databaseURL`, `DATABASE_SECR` uložené natvrdo.

Možný útok:

- Ak je HTML súbor sprístupnený verejne (napr. cez GitHub), kľúče môže získať hocikto a ovládať zariadenie.
- Skriptovanie cez JavaScript na webovej stránke dokáže zneužiť tieto údaje.

Riešenie:

- Nikdy nezverejňovať `DATABASE_SECR`. Namiesto toho použiť **Firestore Authentication + ID tokeny**.
- Služby (napr. REST API) proxyovať cez **bezpečný backend**, kde budú kľúče chránené.

Hrozba č. 3 - Nedostatočné obmedzenie frekvencie dotazov (rate limiting)

- Arduino dotazuje Firebase každých 200 ms.

- Útočník môže posilať tisíce požiadaviek za sekundu (napr. cez skript vo forme curl alebo vlastný bot), čím preťaží Firebase, Arduino alebo pripojenie.

Riešenie:

- Na strane Firebase definovať **security rules s limitáciou prístupov**.
- Na strane Arduino zaviesť **obmedzovač zmien** (napr. ignorovať, ak sa uhol nezmenil o $>5^\circ$, alebo čítať iba každú sekundu).
- Využiť **Cloud Functions** na spracovanie vstupov so server-side kontrolou.

Hrozba č. 4 - Možnosť vzdialeného vykonania škodlivých operácií

Servo motor môže byť ovládaný niekým iným, kto získa prístup k Firebase, čo môže mať **fyzický dopad** (napr. pohyb mechanizmu, otvorenie/zatvorenie brány, apod.).

Riešenie:

- Použiť **overenie pôvodu požiadavky** (napr. token viazaný na zariadenie).
- Implementovať obmedzenie možného počtu požiadaviek za určitý čas (rate limiting).
- Logovať všetky zmeny (napr. do iného Firebase uzla logs/).

Príklad skriptu pre DoS útok (iba na výučbové účely)

UPOZORNENIE: Uvedený skript je len ilustračný – nerealizujte ho v praxi bez výslovného súhlasu vlastníka systému.

```
#!/bin/bash
for i in {1..10000}
do
  curl -X PUT -d "$((RANDOM % 181))" \
    'https://name-of-yourproject-rtdb.firebaseio.com/servo/angle.json'
done
```

Tento skript generuje náhodné hodnoty uhla a posila ich v rýchlej slučke, čo môže:

- Preťažiť Firebase plán (kvóty).
- Zmätok v servopohone.
- Potenciálne poškodiť hardvér (ak nie je mechanicky zabezpečený).

Na základe uvedených hrozieb môžeme definovať nasledovné bezpečnostné opatrenia, ktoré by mali byť pre nás štandardom:

Opatrenie

Popis

Firestore pravidiel	Nastaviť autentifikáciu (napr. auth != null)
Autentifikácia	Firestore Authentication (email/password, token, Google Sign-In...)
Obmedzenie API kľúčov	Obmedziť podľa IP/adresy alebo používať len server-side
Monitoring a logovanie	Logovať zmeny uhla, časy, IP adresy (cez Firestore Functions)
Rate limiting	Na strane klienta aj servera
Fyzické bezpečnostné opatrenia	Zamedziť preťaženiu serva (soft-stop, časové limity)

Ako prevencia k uvedenému DoS útoku je potrebné, aby správca databázy mal prehľad o podozrivých zápisoch. Preto musíme ako základný krok riešiť detekciu podozrivých zápisov pomocou Firestore Cloud Functions.

Firestore Cloud Functions umožňuje **spúšťať serverový kód** pri každej zmene databázy. To nám umožňuje detegovať akúkoľvek zmenu, ktorú môžeme využiť napríklad na:

- detekciu častej alebo extrémnej zmeny uhla
- zalogovanie zmeny do /logs/
- zablokovanie IP alebo na poslanie upozornenia adminovi

Predpoklady:

- Povolené Firestore Functions (Node.js prostredie)
- Hosting aspoň **Spark plan (zadarmo)**

Príklad funkcie (Node.js):

```
const functions = require("firebase-functions");
const admin = require("firebase-admin");
admin.initializeApp();

let lastAngles = {}; // Uchová poslednú hodnotu pre jednoduchú kontrolu

exports.detectAbuse = functions.database
.ref("/servo/angle")
.onWrite(async (change, context) => {
  const newVal = change.after.val();
  const prevVal = change.before.val();
  const now = Date.now();

  if (prevVal === null) return null; // prvý zápis ignorujeme

  const userIP = context.rawRequest?.ip || "unknown";

  // Detekcia extrémnych zmien alebo príliš častého zápisu
  const diff = Math.abs(newVal - prevVal);
  if (diff > 90) {
    await admin.database().ref("/logs").push({
      type: "suspicious_change",
      from: prevVal,
```

```

        to: newVal,
        diff,
        ip: userIP,
        time: now,
        message: "Extrémna zmena uhla detegovaná"
    });
}

// Detekcia rýchleho zápisu (voliteľne podľa časového údaju)
const timeKey = `${context.auth?.uid || "anon"}-${userIP}`;
const last = lastAngles[timeKey] || { time: 0 };
if (now - last.time < 500) {
    await admin.database().ref("/logs").push({
        type: "rate_limit",
        from: prevVal,
        to: newVal,
        time: now,
        ip: userIP,
        message: "Príliš častý zápis detegovaný"
    });
}

lastAngles[timeKey] = { val: newVal, time: now };

return null;
});

```

V súvislosti s detekciou podozrivých zápisov do Firebase databázy je potrebné uvažovať aj o záznamoch logov pre administrátora, aby tieto záznamy boli kedykoľvek kontrolovateľné. Takýto log súbor (logs.json) môže vyzeráť nasledovne:

```

/logs
/-Nl32kdfw...
  type: "suspicious_change"
  message: "Extrémna zmena uhla detegovaná"
  from: 10
  to: 180
  time: 1718550000000
  ip: "192.168.1.25"

```

Admin môže tieto logy prehľadávať cez samostatné webové rozhranie alebo Firebase konzolu. Príklad uvádza, že nastala extrémna zmena uhla, kedy sa zmenil uhol v rozsahu 10-180°, pričom je uvedený čas trvania a detegovaná IP adresa, z ktorej bol vyslaný útok.

Aby sme dokázali bezpečne ochrániť databázu Firebase a zároveň eliminovali možnosť DoS útoku, je nutné oddeliť bezpečnostné pravidlá pre firebase databázu od pôvodného obsahu json. Môžeme to riešiť tak, že umiestnime tento súbor do koreňového priečinka Firebase projektu (napr. vedľa functions/). Príklad firebase.json:

```

{
  "functions": {
    "source": "functions"
  },
  "database": {
    "rules": "database.rules.json"
  },
  "hosting": {
    "public": "public",
    "ignore": ["firebase.json", "**/.*", "**/node_modules/**"]
  }
}

```

Súbor `firebase.json` ak je umiestnený do koreňového adresára (napr. vedľa `functions/`) umožňuje načítavať pravidlá z oddeleného súboru `database.rules.json`, pričom pre detekciu podozrivého správania je potrebné umiestniť súbor `index.js` do adresára `functions`. Súbor `index.js` by mal obsahovať nasledovné:

```
const functions = require("firebase-functions");
const admin = require("firebase-admin");
admin.initializeApp();

const lastWriteTimestamps = {};

exports.detectServoAbuse = functions.database
  .ref("/servo/angle")
  .onWrite(async (change, context) => {
    const after = change.after.val();
    const before = change.before.val();
    const now = Date.now();

    if (before === null) return null; // Prvý zápis ignorujeme

    const uid = context.auth ? context.auth.uid : "anonymous";
    const key = `user-${uid}`;

    const diff = Math.abs(after - before);
    const last = lastWriteTimestamps[key] || 0;
    const interval = now - last;

    const logsRef = admin.database().ref("/logs");

    // Detekcia extrémneho skoku v hodnote uhla (>90°)
    if (diff > 90) {
      await logsRef.push({
        uid,
        type: "large_jump",
        from: before,
        to: after,
        diff,
        time: now,
        message: "Extrémna zmena uhla serva"
      });
    }

    // Detekcia príliš rýchlych zápisov (<500 ms medzi zmenami)
    if (interval < 500) {
      await logsRef.push({
        uid,
        type: "rate_limit_violation",
        from: before,
        to: after,
        interval,
        time: now,
        message: "Zápis uhla príliš rýchlo za sebou"
      });
    }

    lastWriteTimestamps[key] = now;

    return null;
  });
```


Následne súbor obsahujúci bezpečnostné pravidlá `database.rules.json`, ktorý vytvoríme a uložíme ako samostatný súbor, bude obsahovať tieto pravidlá:

```
{
  "rules": {
    "servo": {
      "angle": {
        ".read": "auth != null",
        ".write": "auth != null && newData.isNumber() && newData.val() >= 0
&& newData.val() <= 180",
        ".validate": "newData.val() != data.val() && (newData.val() -
data.val()).abs() <= 90"
      }
    },
    "logs": {
      ".read": "auth != null &&
root.child('users').child(auth.uid).child('isAdmin').val() === true",
      ".write": "auth != null"
    }
  }
}
```

Tento dokument bol vypracovaný pre project Nitrianske kompetenčné centrum pre kybernetickú bezpečnosť (17R05-04-V01-00003)



Nitrianske kompetenčné
centrum kybernetickej
bezpečnosti



PLÁN [OBNOVY]