Sistem de casă inteligentă folosind dispozitive IoT

Vasilovici Maria, 345C1

Cuprins

1. Introducere

Proiectul propus reprezintă dezvoltarea unui sistem de casă inteligentă, care integrează un set de senzori și actuatori pentru monitorizarea și controlul condițiilor din locuință. Acest sistem va oferi utilizatorilor informații valoroase în timp real prin intermediul unei aplicații pe telefon (Blue Alert), având ca scop creșterea siguranței și confortului acestora.

Sistemul se bazează pe patru senzori: un senzor de temperatură, un senzor PIR (Passive Infrared), un senzor de gaz metan și un senzor de puls. Pe lângă aceștia, proiectul include și un actuator, respectiv un LED care poate fi controlat de la distanță prin aplicație, permițând utilizatorului să-l aprindă sau să-l stingă după necesitate.

Datele colectate de la senzori vor fi transmise prin protocolul MQTT către un broker local (laptop-ul utilizatorului), iar acestea vor fi vizualizate sub forma unor grafice interactive în aplicație. În plus, sistemul va trimite notificări utilizatorilor în cazurile de detectare a gazului metan sau de mișcare, în funcție de setările din aplicație, sporind astfel securitatea locuinței. Acest proiect își propune să îmbunătățească atât siguranța cât și confortul utilizatorilor prin integrarea eficientă a tehnologiilor IoT (Internet of Things) într-un sistem accesibil și ușor de utilizat.

Mai jos sunt detaliate scopurile fiecărei funcționalități a sistemului:

Monitorizarea temperaturii ambientale: Senzorul de temperatură va măsura constant temperatura din locuință. Cunoașterea acesteia poate avea mai multe beneficii. În primul rând, ajută la menținerea unui mediu confortabil pentru utilizatori, având în vedere că temperatura influențează starea de bine a acestora. De asemenea, acest senzor poate contribui la economisirea energiei, prin optimizarea utilizării sistemelor de climatizare, întrucât utilizatorul poate ajusta temperatura în funcție de nevoile sale, pe baza datelor furnizate de senzor.

Detectarea mișcării prin senzorul PIR (alarmă antifurt): Senzorul PIR va detecta mișcările din interiorul casei, având un rol crucial în securitatea locuinței. Acest senzor va funcționa ca o alarmă antifurt, semnalând orice mișcare neautorizată în zonele de interes (de exemplu, în cazurile în care casa este goală). Atunci când este activată opțiunea în aplicație, utilizatorul va fi notificat imediat printr-o alertă dacă se detectează mișcare, ceea ce îl ajută să reacționeze rapid în cazul unei intruziuni. Acest senzor poate fi util în alte contexte, cum ar fi monitorizarea prezenței persoanelor în locuință, fiind de ajutor în gospodării cu copii mici sau vârstnici care necesită supraveghere.

Detectarea gazului metan pentru siguranță: Unul dintre cele mai importante obiective ale acestui proiect este siguranța locuinței, iar senzorul de gaz metan va juca un

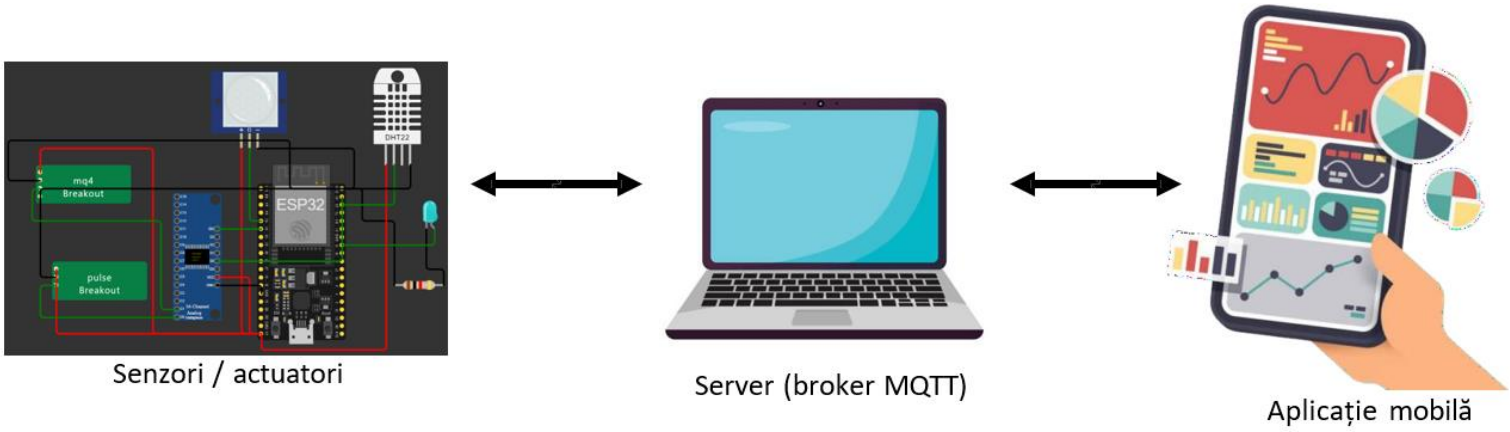
rol esențial în acest sens. Acest senzor va detecta orice scurgere de gaz metan, un pericol semnificativ care poate cauza explozii sau intoxicații. În cazul în care gazul metan este detectat, utilizatorul va fi alertat imediat printr-o notificare, ceea ce îi va permite să ia măsuri rapide, cum ar fi oprirea alimentării cu gaz sau evacuarea locuinței.

Monitorizarea pulsului pentru sănătate: Senzorul de puls va fi un element esențial pentru persoanele care suferă de afecțiuni cardiovasculare sau respiratorii, precum și pentru persoanele vârstnice sau cele care au nevoie de monitorizare constantă a stării lor de sănătate. Acesta va măsura ritmul cardiac al utilizatorilor, iar modificările semnificative ale pulsului se pot observa în aplicație, permițându-i persoanei respective să ia măsuri rapide, cum ar fi consultarea unui medic, sau alte metode de a preveni complicațiile.

Controlul LED-ului prin aplicație: Actuatorul, în acest caz un LED controlat prin aplicație, permite utilizatorului să gestioneze iluminatul locuinței dintr-o locație la distanță. Aceasta poate fi util în diverse scenarii, cum ar fi aprinderea luminii atunci când utilizatorul se întoarce acasă sau controlarea iluminatului pe timp de noapte fără a fi necesar să se deplaseze pentru a aprinde un întrerupător.

2. Arhitectură

Arhitectura sistemului propus pentru casa inteligentă este bazată pe o rețea IoT în care fiecare componentă este conectată la unitatea centrală de control (topologia fizică Star) prin intermediul unui protocol de comunicație eficient.



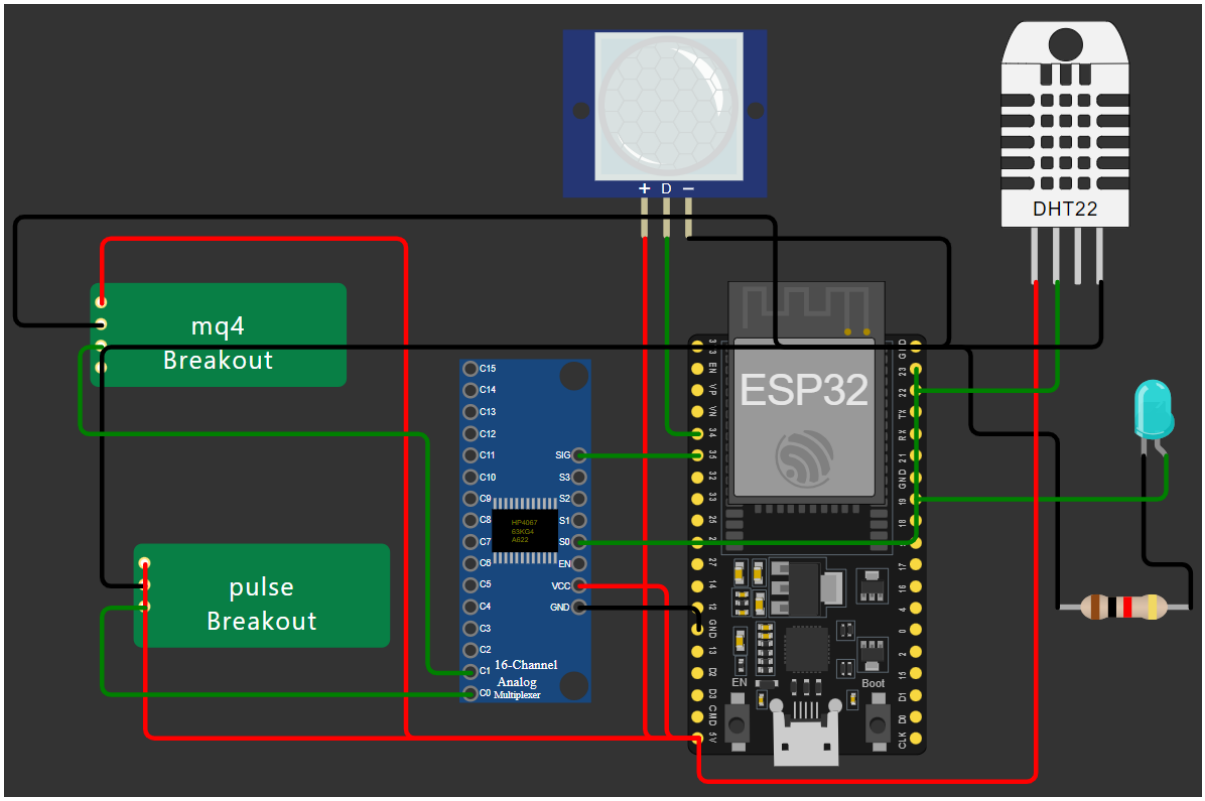
Topologia conceptuală utilizată este Fog Computing deoarece procesarea are loc în apropierea locului în care sunt colectate datele. Brokerul MQTT va fi găzduit pe un laptop local. Acesta va primi și transmite mesajele de la senzori și va direcționa datele către

aplicația mobilă a utilizatorului. Totodată, va trimite informatii de la aplicatie catre actuatori (aprinderea becului).

În cadrul acestui sistem, se utilizează următorul protocol de comunicație, respectiv protocol de rețea:

- **MQTT (Message Queuing Telemetry Transport)**: Acesta este un protocol de mesagerie ușor și eficient, utilizat pentru transmiterea datelor între dispozitivele IoT. MQTT este ideal pentru sisteme în care lățimea de bandă și resursele sunt limitate, având un consum redus de energie și resurse. Senzorii vor trimite datele către brokerul MQTT, iar aplicația mobilă va prelua aceste date pentru a le afișa utilizatorului. De asemenea, în cazul unor evenimente importante (de exemplu, detecția gazului sau mișcării), se vor trimite notificări prin MQTT către aplicație.
- **Wi-Fi**: Toate dispozitivele vor comunica prin rețeaua Wi-Fi locală, oferind o conexiune rapidă și fiabilă pentru transferul datelor între microcontroller și brokerul MQTT.

Descrierea componentelor:



- Senzorii:
  - **Senzor de Temperatură DHT22**: Monitorizează temperatura din locuință, trimițând datele către microcontroller pentru a fi procesate și apoi transmise brokerului MQTT.
  - **Modul Senzor PIR HC-SR501 (Senzor de Mișcare)**: Detectează mișcările din locuință și activează alarmele în cazul unui comportament suspect. Atunci când se detectează mișcare, senzorul va trimite o notificare aplicației mobile.
  - **Modul Senzor de Gaz Metan CNG MQ-4**: Detectează orice scurgere de gaz metan periculos. Dacă detectează o concentrație de gaz care depășește limita sigură, va trimite o alertă imediată utilizatorului prin aplicație.
  - **Modul Senzor Puls**: Măsoară ritmul cardiac al utilizatorului, iar datele sunt trimise către aplicația mobilă pentru monitorizare continuă.
- Actuatorul:
  - **LED controlat prin aplicație**: Acest LED poate fi aprins sau stins de la distanță prin intermediul aplicației, oferind utilizatorului posibilitatea de a controla iluminatul din locuință în mod convenabil.
- Altele:
  - **Microcontroller ESP32:** Este centrul de control al sistemului, care colectează datele de la senzori, procesează informațiile și le transmite către brokerul MQTT. Include conectivitate Wi-Fi și Bluetooth, facilitând comunicația în timp real.
  - **Multiplexor analogic/digital CD74HC4067:** Permite conectarea mai multor semnale de intrare către un singur canal de ieșire. ESP32 include două convertoare ADC, iar unul dintre convertoare este folosit parțial pentru subsistemul Wi-Fi. Multiplexorul este utilizat pentru a conecta mai multe surse de semnal analogic (senzorii conectați la chipurile MQ-4 și Pulse) la un singur canal ADC al ESP32.

- Aplicația mobilă Blue Alert: Aplicația joacă un rol esențial în interacțiunea utilizatorului cu sistemul. Ea permite vizualizarea datelor colectate de la senzori, afișarea acestora sub formă de grafice și trimiterea de notificări în caz de evenimente importante (de exemplu, detectarea gazului metan sau mișcarea într-o zonă securizată). De asemenea, utilizatorul poate controla LED-ul din aplicație și poate modifica setările de notificare sau activarea senzorilor.

3. Implementare

- Simularea componentelor hardware (senzori, actuator si microcontroller ESP32)

Pentru simularea sistemului hardware al proiectului, am utilizat platforma Wokwi, care permite testarea și validarea funcționalităților fără a necesita componente fizice. Configurația include un microcontroller ESP32 conectat la diverși senzori și un actuator, conform diagramei proiectate.

În această simulare, majoritatea componentelor, precum ESP32, LED-ul, senzorul PIR și senzorul DHT22, au fost selectate direct din biblioteca integrată a platformei. Totuși, pentru senzorul de gaz MQ-4 și senzorul de puls, care nu erau disponibile, a fost necesar să implementez propriile simulări utilizând limbajul C. După scrierea codului, l-am compilat împreună cu fișierul JSON al fiecărei componente într-un fișier .wasm (WebAssembly). Aceste fișiere au fost apoi încărcate în simularea Wokwi, permițând utilizarea senzorilor personalizați ca și cum ar fi parte integrantă a bibliotecii platformei.

Odată cu adăugarea fișierelor .wasm și configurarea completă a ESP32 (incluzând scrierea de cod pentru a gestiona conexiunea cu brokerul MQTT), întreaga simulare a fost integrată în Wokwi. Am testat funcționalitatea pentru a verifica:

- Generarea corectă a datelor de către senzorii personalizați.
- Transmiterea datelor de la ESP32 către brokerul MQTT.
- Recepția comenzilor de la aplicația mobilă pentru controlul LED-ului.

Această abordare a permis simularea completă a întregului sistem hardware, inclusiv componente personalizate, și a demonstrat compatibilitatea cu aplicația software dezvoltată.

- Configurarea brokerului MQTT

Urmărind explicațiile din laboratorul 5, am instalat Mosquitto pe laptop utilizând pachetul disponibil pe pagina oficială. După instalare, am configurat Mosquitto pentru a permite conexiuni externe. Am editat fișierul mosquitto.conf, adăugând liniile necesare pentru a activa portul standard MQTT (1883) și a permite conexiuni fără autentificare.

Pentru a porni brokerul, am utilizat comanda: mosquitto -v -c "C:\Program Files\mosquitto\mosquitto.conf" . Această configurare permite dispozitivelor din rețea să comunice prin protocolul MQTT, iar modul detaliat (-v) oferă informații despre conexiunile și mesajele gestionate (de exemplu "New client connected from 192.168.10.3:38046 as android_app", "Sending PUBLISH to android_app (d0, q0, r0, m0, 'home/sensors/temperature', ... (2 bytes))").

- Dezvoltarea aplicației mobile

Pentru realizarea aplicației Blue Alert, am utilizat Android Studio, implementând funcționalitățile principale printr-o structură modulară. În MainActivity.java, am implementat logica principală a aplicației, gestionând legătura dintre elementele vizuale definite în fișierul activity_main.xml și funcționalitatea acestora. Codul include metode pentru afișarea și actualizarea graficelor (pentru temperatură și puls), gestionarea notificărilor, precum și logica pentru manipularea mesajelor primite pe topicurile MQTT la care aplicația este abonată. De asemenea, aici sunt tratate interacțiunile utilizatorului cu switch-urile pentru activarea/dezactivarea LED-ului și a notificărilor pentru senzorul de mișcare.

În MqttHandler.java, am centralizat funcționalitățile legate de protocolul MQTT. Acest fișier conține metode pentru inițierea conexiunii la brokerul MQTT, publicarea de mesaje către diverse topicuri, abonarea la topicuri relevante, precum și dezabonarea. Codul din acest fișier este structurat pentru a fi reutilizabil și eficient, facilitând gestionarea comunicației dintre aplicație și dispozitivele conectate.

4. Vizualizare și Procesare de Date

Pentru ca utilizatorii să beneficieze de o interfață intuitivă și funcțională, aplicația mobilă Blue Alert oferă o experiență care combină controlul sistemului cu monitorizarea datelor senzorilor în timp real și analizarea istoricului acestora. Aceasta este realizată printr-o interfață intuitivă, bine organizată și metode eficiente de procesare și afișare a datelor.

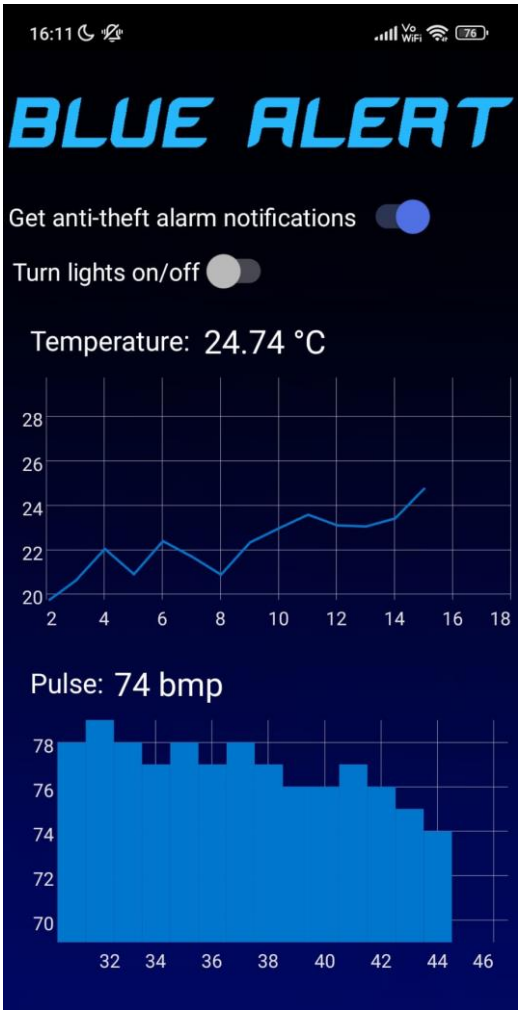
- Funcționalități de control prin switch-uri

Controlul actuatorului (LED-ul):
Utilizatorii au posibilitatea de a controla LED-ul din aplicație printr-un switch dedicat. Activarea acestui switch trimite un mesaj către brokerul MQTT pe un topic specific (home/actuators/led), care este interpretat de microcontroller pentru a aprinde sau stinge LED-ul. Acest control simplu și intuitiv permite utilizatorilor să interacționeze cu sistemul direct din aplicație.

Gestionarea notificărilor pentru senzorul PIR (mișcare):
Un al doilea switch permite utilizatorilor să activeze sau să dezactiveze notificările generate de

senzorul de mișcare. Această funcționalitate este utilă pentru a preveni notificările inutile în situațiile în care detectarea mișcării nu este relevantă (de exemplu, atunci când utilizatorul este acasă). Mesajele MQTT sunt folosite pentru a informa microcontrollerul despre starea acestui switch.

Notificările pentru senzorul de gaz:
Spre deosebire de senzorul de mișcare, notificările pentru gazul metan sunt critice și nu pot fi dezactivate. Acest lucru asigură un nivel ridicat de siguranță, alertând utilizatorii în caz de scurgeri de gaz indiferent de preferințele lor.



- Afișarea datelor senzorilor

Temperatura curentă și istoricul temperaturilor:
În aplicație, temperatura curentă (în grade Celsius) este afișată într-un format ușor de citit, oferind utilizatorului o imagine clară asupra condițiilor din locuință.
Sub valoarea curentă, un grafic afișează valorile istorice ale temperaturii, permițând utilizatorului să observe variațiile de-a lungul timpului. Graficul este generat prin librăria GraphView și este actualizat automat pe baza datelor primite de la brokerul MQTT.

Pulsul curent și evoluția acestuia:
Similar, pulsul utilizatorului este afișat în timp real în bătăi pe minut (bpm), oferind informații critice pentru monitorizarea sănătății, în special pentru persoanele cu afecțiuni cardiovasculare. Graficul afișat sub valoarea curentă permite utilizatorului să analizeze tendințele pulsului, identificând creșteri sau scăderi care ar putea indica situații anormale.

- Procesarea datelor

Datele de la senzori sunt transmise microcontrollerului, care le publică pe topicurile corespunzătoare prin MQTT. Aplicația mobilă se abonează la aceste topicuri și primește datele în timp real. De exemplu: temperatura este publicată pe topicul home/sensors/temperature, iar pulsul este publicat pe topicul home/sensors/pulse.

Acest sistem de vizualizare și procesare a datelor senzorilor oferă utilizatorului acces ușor la informații esențiale, alături de un nivel ridicat de control, contribuind astfel la o experiență completă și satisfăcătoare.

5. Securitate

Pentru a asigura securitatea transmisiei de date între brokerul MQTT și aplicația mobilă, am configurat comunicarea să utilizeze protocolul SSL (Secure Sockets Layer) prin portul 8883. Această configurare adaugă un strat de protecție esențial împotriva interceptării și manipulării datelor.

Protocolul SSL criptează datele transmise între broker și aplicația mobilă, împiedicând atacatorii să vizualizeze sau să modifice informațiile sensibile. Aplicația utilizează o conexiune SSLSocketFactory, care stabilește sesiuni securizate și gestionează schimbul de chei pentru criptare și decriptare. Un alt avantaj al utilizării SSL este autentificarea pe bază de certificate. Brokerul MQTT utilizează un certificat SSL pentru a-și dovedi identitatea către client (aplicația mobilă). Prin aceste măsuri, sistemul asigură o transmisie de date sigură, protejând informațiile utilizatorului și garantând integritatea și confidențialitatea comunicației dintre aplicația mobilă și brokerul MQTT.

```
BROKER_URL = "ssl://192.168.68.192:8883";
```

6. Provocări și Soluții

Una dintre cele mai mari dificultăți a fost configurarea conexiunii MQTT în aplicația mobilă. Am avut inițial probleme în a înțelege cum să implementez funcțiile necesare pentru conectare, publicare și abonare la topicuri. Am căutat informații din mai multe surse și am întâlnit multe exemple care erau fie prea complexe, fie incomplete. În cele din urmă, am găsit pe GitHub un proiect simplu și ușor de înțeles (https://github.com/ashenishanka/MQTT-Android-Java-Example), pe baza căruia am reușit să pornesc. Acest exemplu mi-a oferit o bază solidă, pe care am adaptat-o la nevoile proiectului meu.

O altă provocare majoră a fost să mă familiarizez cu Android Studio, aceasta fiind prima dată când am lucrat cu acest mediu de dezvoltare și, în general, cu programarea aplicațiilor mobile. Fiind complet nouă în acest domeniu, a trebuit să mă documentez intens folosind tutoriale introductive pentru Android Studio, învățând să creez interfețe simple, să gestionez activitățile și să implementez funcționalitatea graficelor și notificărilor.

În ceea ce privește brokerul MQTT, am avut relativ puține probleme tehnice, dar am întâmpinat o dificultate practică legată de schimbarea adresei IP a laptopului, care varia în funcție de rețeaua Wi-Fi utilizată. Trebuia să fiu mereu atentă să configurez telefonul și laptopul pe

aceeași rețea pentru ca transmisia de date să funcționeze corect. Această problemă a fost mai degrabă una de organizare, dar a fost necesar să îmi amintesc constant să verific aceste detalii.

Simularea hardware a fost un alt punct provocator. Inițial, pentru testarea aplicației, am creat un program simplu în Python care trimitea date simulate prin MQTT către broker. Această soluție a fost relativ ușor de implementat și m-a ajutat să mă asigur că funcționalitatea aplicației era corectă. Cu toate acestea, am dorit o simulare hardware mai realistă, motiv pentru care am trecut la platforma Wokwi. Aici am întâlnit probleme, mai ales că unele componente necesare, precum senzorul MQ-4 și senzorul de puls, nu erau disponibile. După multe încercări și căutări, am primit ajutor de la un coleg care mi-a explicat cum să creez componente personalizate folosind cod C și să le integrez sub formă de fișiere .wasm.