2024

Communicatieprotocol

Onderzoek & Advies

PROJECT

TADRAŁA,PIOTR P.P.

O-PP-CMK

I. MANAGEMENT SAMENVATTING

Dit document richt zich op het onderzoek naar een communicatieprotocol voor een modulair systeem dit uit een desktopapplicatie en een microcontroller bestaat. Het systeem is bedoeld voor een kunstinstallatie die gebruik maakt van Plant-E technologie. Ons concept omvat het spel Pong dat wordt afgespeeld op een display van E-Planten.

Het systeem bestaat uit twee modules: een desktopapplicatie waarop het spel wordt gesimuleerd, en een microcontroller die visuele data ontvangt vanuit de desktopapplicatie en deze visualiseert. Hiervoor moet een geschikt communicatieprotocol worden gekozen.

De potentiële opties zijn: Serial Ports, REST API en Bluetooth Low Energy. Om de geschikte optie te kiezen, zijn de requirements geïdentificeerd: low latency, robust en energy eficient. Door voor elk protocol een testimplementatie te maken, was het mogelijk om een duidelijke vergelijking te maken.

Door de voor- en nadelen van elke implementatie in kaart te brengen, ben ik tot de conclusie gekomen dat Serial Ports het meest geschikte communicatieprotocol is voor ons systeem.

II. VERSIEBEHEER EN DISTRIBUTIE

VERSIEBEHEER

Versie	**Omschrijving**	**Auteurs**
1.0	Eerste opzet van het document	Piotr T.
1.1	Feedback verwerkt	Piotr T.

DISTRIBUTIEBEHHER

Versie	**Ontvanger**
1.0	Monique V. & Berry S.

III. WOORDENLIJST

Begrip	**Omschrijving**
Modulair	Opgebouwd uit meerdere componenten
Portal	Desktop-applicatie waarop Plant-E apps zullen draaien
ESP32	Een microcontroller zoals een Raspberry pi of Arduino
Display Controller	ESP32 controller die als prodxy tussen de planten en de portal zal functioneren
Latency	Latency is vertraging in dataoverdracht over een communicatienetwerk
Robust	Een robust oplossing wilt zeggen dat die niet snel zal falen

CONTENTS

1.0 INLEIDING

Dit semester zijn we aan de slag gegaan met het ontwerpen en realiseren van een kunstinstallatie die gebruikmaakt van de Plant-E technologie. Deze zou eventueel op Glow gepresenteerd kunnen worden. In de eerste sprint hebben we ervoor gekozen om met behulp van E-Planten een grid te creëren waarop het spel Pong gespeeld kan worden. Dit willen we bereiken door ledjes te gebruiken die gevoed worden door E-Planten en deze in een grid plaatsen. Hierop willen we vervolgens het spel visualiseren.

Om dit te realiseren hebben we twee prototypes bedacht. In dit document zal ik een daarvan toelichten, namelijk een modulaire opzet waarbij het spel wordt gesimuleerd op een desktopapplicatie en vervolgens wordt weergegeven op het display met een microcontroller als proxy. Een belangrijke onderdeel hiervan is hoe we de communicatie tussen de twee modules willen laten verlopen. Dit document beschrijft mijn onderzoek en beslissing over het beste communicatieprotocol voor ons concept.

2.0 CONTEXT

2.1 INLEIDING

Voor het concept van een modulaire opzet waarbij de logica op een desktopapplicatie wordt uitgevoerd en de visuele data naar een display-microcontroller wordt doorgestuurd, moet er een beslissing worden gemaakt over hoe de data getransfered zal worden. In dit document zal ik mijn onderzoek presenteren naar de potentiële communicatieprotocollen die we kunnen gebruiken en een advies geven over welke het beste past bij de opgestelde architectuur.

2.2 SITUATIE

Chris G. en Sandra V. hebben het project voorgesteld, waarbij zij een concept inclusief prototype willen ontvangen dat gebruikmaakt van de Plant-E technologie. Dit concept zou mogelijk gepresenteerd kunnen worden tijdens Glow.

2.3 PROBLEEMSTELLING

Voor het creëren van een prototype dat uit twee modules bestaat moet je een beslissing maken over hoe je deze met elkaar wilt laten communiceren. Het is voor ons voornamelijk van belang dat de twee modules real-time met elkaar moeten kunnen communiceren met minimale delay. Daarnaast is het belangrijk dat het communicatieprotocol robust is, we niet willen dat de verbinding tussen de modules tijdens een Pong spel zomaar verbreekt.

2.4 DOELSTELLING

Met dit document wil ik tot de conclusie komen wat het beste communicatieprotocol voor ons modulaire applicatie is.

2.5 CONCLUSIE

Dit document onderzoekt welk communicatieprotocol geschikt is voor ons prototype. Het volgende hoofdstuk beschrijft de aanpak hiervan en hoe ik tot een conclusie wil komen.

3.0 ONDERZOEKSAANPAK

3.1 INLEIDING

Om tot een conclusie te komen, is het in eerste instantie belangrijk om duidelijk te maken wat de hoofdvraag van je onderzoek is. In dit hoofdstuk zal ik uitleggen waar mijn onderzoek zich op richt en welke methodes ik heb gebruikt.

3.2 HOOFDVRAAG

Welke communicatieprotocol is het meest geschikt voor een modulair applicatie die uit een desktop app en microcontroller bestaat?

3.3 DEELVRAGEN

Index	**Deelvraag**
D1	Welke protocollen kun je gebruiken voor communicatie tussen software en hardware?
D2	Wat zijn de belangrijkste eigenschappen van een protocol?
D3	Wat voor invloed hebben de eigenschappen op de algemene werking van het prototype?

3.4 ONDERZOEKSMETHODEN

Deelvraag	Strategie	Methode	Verwachte uitkomst
D1	Internet research	Literatuuronderzoek	Lijst met potentiële opties
D2	Internet research	Literatuuronderzoek	Lijst met de belangrijkste eigenschappen
D3	Experiments & Internet research	Tests uitvoeren	Een vergelijking tabel

3.5 SCOPE

De scope van mijn onderzoek richt zich op het onderzoeken van potentiële protocollen en deze vervolgens te vergelijken om de verschillende eigenschappen in kaart te brengen.

3.6 CONCLUSIE

Nu de deelvragen en het plan van aanpak duidelijk zijn, zal ik in het volgende hoofdstuk mijn onderzoek presenteren. Door de onderzoeksvragen te beantwoorden, zal er een duidelijk antwoord op de hoofdvraag ontstaan.

4.0 WELKE PROTOCOLLEN KUN JE GEBRUIKEN VOOR COMMUNICATIE TUSSEN HARDWARE EN SOFTWARE?

4.1 INLEIDING

In dit hoofdstuk richt ik me op het onderzoek naar wat de beschikbare opties zijn voor communicatie tussen software en hardware. Zodra we een lijstje met mogelijke protocollen hebben kunnen we naar de volgende deelvraag kijken over wat de belangrijkste eigenschappen zijn in de context van ons concept.

4.2 SERIAL PORTS

Als je een communicatieprotocol tussen software en hawdware opzoekt, krijg je meteen de meest voorkomen protocol tegen, namelijk serial ports. Wat is serial ports? Serial Ports is een interface voor het uitwisselen van data via een fysieke verbinding, zoals een USB-kabel. In de context van dit concept zouden we data

kunnen uitwisselen via een fysieke kabel die de desktop applicatie verbindt met de microcontroller.

• *Purushotam Kumar. (2019). Serial Communication between node js and Arduino (read and write data)*
• *Javatpoint. (Onbekend). What is a Serial Port*

4.3 REST API

Een REST API is een manier waarop software via het netwerk kan communiceren met andere software. RESTful API's worden vaak gebruikt voor het bouwen van webservices. In ons concept zou de desktopapplicatie een verzoek sturen naar een specifieke poort op het netwerk, dat vervolgens door de microcontroller wordt ontvangen.

• *Uptrends. (Onbekend). Wat is REST API*

4.4 BLEUTOOTH LOW ENERGY

BLE (Bluetooth Low Energy) is een draadloze communicatieprotocol die ontworpen is voor apparaten met beperkte energiebronnen, zoals sensoren, wearables en IoT-apparaten. Het biedt een efficiënte manier om kleine hoeveelheden data over te dragen met een lager stroomverbruik dan traditionele Bluetooth.

• *Doojin Kang. (2020). Arduino BLE as IoT*

4.5 CONCLUSIE

Nu we een kort lijstje hebben met mogelijke oplossingen en een antwoord op de eerste deelvraag **D1**, kunnen we in het volgende hoofdstuk kijken wat de belangrijkste eigenschappen zijn van een communicatieprotocol binnen ons concept.

5.0 WAT ZIJN DE BELANGRIJKSTE EIGENSCHAPPEN VAN EEN PROTOCOL?

5.1 INLEIDING

Elk protocol heeft zijn sterke en zwakke punten. Daarom is het belangrijk een protocol te kiezen dat het meest geschikt is voor jouw concept. Door eerst de requirements in ons concept te identificeren, kunnen we vervolgens bepalen welke belangrijke eigenschappen een protocol moet hebben waarop we ons moeten

richten.

5.2 REQUIREMENTS

Ons concept richt zich op het ontwikkelen van een spel dat in real-time door gebruikers gespeeld kan worden. Het is daarom belangrijk dat er zo min mogelijk delay zit tussen de modules. Daarnaast is het belangrijk dat de verbinding stabiel blijft, we willen niet dat een verbroken vebinding de spelervaring negatief beïnvloedt. Tot slot moet het systeem energiezuinig zijn. Aangezien we gebruik maken van Plant-E technologie, is het esentieel dat het systeem zo min mogelijk stroom verbruikt.

5.3 BELANGRIJKSTE EIGENSCHAPPEN

- **Low Latency**
  De belangrijkste requirement is om de latency zo laag mogelijk te houden. Dit heeft er voornamelijk mee te maken dat het spel in real-time user inputs moet doorsturen naar de desktopapplicatie en vervolgens de visuele data naar de controller moet upsturen. Bij een hoge latency zal het spel niet synchroon lopen met de user inputs.
- **Robust**
  Als tweede moet de verbinding robust zijn, wat betekent dat deze niet zomaar verbroken kan worden en dat requests consistent aankomen zonder onderbrekingen.
- **Energy eficient**
  De microcontroller zal op Plant-e technologie runnen, en dus is het belangrijk dat deze zo energiezuinig mogelijk is.

5.4 CONCLUSIE

Het in kaart brengen van de requirements heeft ons mogelijk gemaakt de belangrijkste eigenschappen van een communicatieprotocol voor ons concept te identificeren en daarmee deelvraag **2D** te beantwoorden. Nu het duidelijk is waarop we ons moeten focussen, kunnen we de drie genoemde protocollen onderzoeken, met als focus de genoemde eigenschappen.

6.0 WAT VOOR INVLOED HEBBEN DE EIGENSCHAPPEN OP DE ALGEMENE WERKING VAN HET PROTOTYPE?

6.1 INLEIDING

Nu het duidelijk is aan welke requirements het protocol voor ons concept moet voldoen, kunnen we de invloed van elke eigenschap op het systeem gaan onderzoeken. Dit doe ik door elk protocol te implementeren en ze vervolgens met elkaar te vergelijken. Op basis hiervan kunnen we een matrix opstellen waaruit we gemakkelijk conclusies kunnen trekken.

6.2 TEST IMPLEMENTATIE

Voordat ik een vergelijking matrix kan creëren, moet ik eerst de genoemde protocollen implementeren. Test implementaties zijn hier terug te vinden:

- **9.1 Serial ports implementatie**
- **9.2 REST API implementatie**
- **9.3 BLE implementatie**

6.3 VERGELIJKING MATRIX

	Latency	Energieverbruik	Stability	Range
Serial Ports	5	5	5	3
REST API	3	2	3	5
BLE	5	5	4	2

6.4 VOOR EN NADELEN

Protocol	Voordelen	Nadelen
Serial Ports	• Lage latency • Eenvoudig te implementeren	• Langzaam data-transfer speed • Fysieke verbinding vereist.
REST API	• Scalability • Draadloos verbinding	• Latency en speed is variabel obv netwerkcondities • Kan energy-intensief zijn
BLE	• Energizuinig • Lage latency • Draadloos	• Beperkte data transfer snelheid • Beperkte afstand van werking • Complexere implementatie

• *Kingswaysoft. (2024).* *Pros and Cons of SOAP and REST API Integrations*
• *Electronicsweekly. (2014).* *The prons and cons of BLE*
• Analog. (Onbekend). What are the advantages and disadvantages of serial and parallel mode?

6.5 CONCLUSIE

Door voor alle protocollen een test implementatie te creëren en de resultaten daarvan in kaart te brengen, hebben we een duidelijk beeld gekregen van de eigenschappen van elk protocol. Op basis van deze vergelijking is duidelijk geworden dat Serial ports het meest optimale protocol is voor ons systeem. Hiermee zijn deelvraag **D3** en de **hoofdvraag** beantwoord.

7.0 CONCLUSIE

Door eerst naar de potentiële opties te kijken, vervolgens alle requirements in kaart te brengen, en tot slot test implementaties voor alle genoemde opties te creëren, is er een overzichtelijke matrix ontstaan. Hieruit kunnen we duidelijk concluderen dat Serial ports het meest optimale protocol is voor ons concept.

8.0 BRONNEN

• *Purushotam Kumar. (2019).* *Serial Communication between node js and Arduino (read and write data)*
• *Assistant. (2024).* *What is the disadvantage of serial port*

- *Javatpoint. (Onbekend).* *What is a Serial Port*
- *Uptrends. (Onbekend).* *Wat is REST API*
- *Doojin Kang. (2020).* *Arduino BLE as IoT*
- *kingswaysoft. (2024).* *Pros and Cons of SOAP and REST API Integrations*
- *Electronicsweekly. (2014).* *The prons and cons of BLE*
- Analog. (Onbekend). What are the advantages and disadvantages of serial and parallel mode?

9.0 BIJLAGEN

9.1 SERIAL PORTS IMPLEMENTATIE

Desktopapplicatie:

```
const { usb :  } = require('usb');
const {SerialPort} = require("serialport");
const {ReadlineParser} = require("@serialport/parser-readline");
const {ipcMain : IpcMain } = require("electron");

function sendData(data) : undefined | Promise<...> {
  if(!this.serialport || !this.parser){ return }

  this.parser.removeAllListeners();

  return new Promise( executor: resolve => {
    this.serialport.write(JSON.stringify(data));
    this.parser.on('data', response => {
      resolve(response);
    });
  })

}
```

Microcontroller:

```
void listenToCommands(){
  if(Serial.available() <= 0){ return; }

  String data = Serial.readString();

  StaticJsonDocument<200> json;
  DeserializationError error = deserializeJson(json, data);

  if (error) {
    Serial.println("FAIL");
    return;
  }

  String command = json["command"];
  if(command == "INIT"){
    init();
  }
  else if(command == "VISUALIZE"){
    visualize(json["image"]);
  }

  blink();
}
```

9.2 REST API IMPLEMENTATIE

Desktopapplicatie

```
const express : e | () => core.Express  = require('express');
const app : any | Express  = express();

app.get('/data', (req : Request<P, ResBody, ReqBody, ReqQuery, LocalsObj> , res : Response<ResBody, LocalsObj> ) : void  => {
  res.json( body: { message: 'Hello from Node.js' });
});

app.listen( port: 3000,  hostname: () : void  => {
  console.log('Server running on port 3000');
});
```

Microcontroller:

```
#include <WiFi.h>
#include <HTTPClient.h>

const char* ssid = "TEST_SSID";
const char* password = "1234";

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected to WiFi");
}

void loop() {
  if ((WiFi.status() == WL_CONNECTED)) {
    http.begin("http://127.0.0.1:3000/plant-e");
    int httpCode = http.GET();
    if (httpCode > 0) {
      String payload = http.getString();
      Serial.println(payload);
    }
    http.end();
  }
  delay(10000);
}
```

9.3 BLE IMPLEMENTATIE

Desktopapplicatie

```
const noble = require('@abandonware/noble');

noble.on('stateChange', (state) : void  => {
  if (state === 'poweredOn') {
    noble.startScanning();
  } else {
    noble.stopScanning();
  }
});

noble.on('discover', (peripheral) : void  => {
  if (peripheral.advertisement.localName === 'ESP32_BLE_Device') {
    noble.stopScanning();
    connectAndSendData(peripheral);
  }
});

function connectAndSendData(peripheral) : void  {
  peripheral.connect(() : void  => {

    peripheral.discoverSomeServicesAndCharacteristics(['service_uuid'], ['characteristic_uuid'], (error, services, characteristics) : void  => {
      if (characteristics.length > 0) {
        const characteristic = characteristics[0];

        const dataToSend : Buffer  = Buffer.from( str: 'TEST',  encoding: 'utf-8');
        characteristic.write(dataToSend, true, () : void  => {
          peripheral.disconnect();
        });
      }
      else {
        peripheral.disconnect();
      }
    });
  });
}
```

Microcontroller

```
#include <BLEDevice.h>
#include <BLEServer.h>
#include <BLEUtils.h>
#include <BLE2902.h>

BLEServer *pServer = NULL;
BLECharacteristic *pCharacteristic = NULL;

// BLE service en karakteristiek UUID's
#define SERVICE_UUID "4fafc201-1fb5-459e-8fcc-c5c9c331914b"
#define CHARACTERISTIC_UUID "beb5483e-36e1-4688-b7f5-ea07361b26a8"

class MyServerCallbacks : public BLEServerCallbacks {
  void onConnect(BLEServer *pServer) {
    Serial.println("Connected to central device");
  }

  void onDisconnect(BLEServer *pServer) {
    Serial.println("Disconnected from central device");
  }
};

void setup() {
  Serial.begin(115200);
  BLEDevice::init("ESP32_BLE_Device");
  pServer = BLEDevice::createServer();
  pServer->setCallbacks(new MyServerCallbacks());

  BLEService *pService = pServer->createService(SERVICE_UUID);

  pCharacteristic = pService->createCharacteristic( CHARACTERISTIC_UUID, BLECharacteristic::PROPERTY_READ | BLECharacteristic::PROPERTY_WRITE);

  pCharacteristic->setCallbacks(new BLECharacteristicCallbacks());

  pService->start();
  BLEAdvertising *pAdvertising = pServer->getAdvertising();
  pAdvertising->start();
}

void loop() {
}
```