Projektarbeit Mobile & Ubiquitous Computing

Sommersemester 2023

Ausgabe: 01.06.2023

Ostbayerische Technische Hochschule Amberg-Weiden

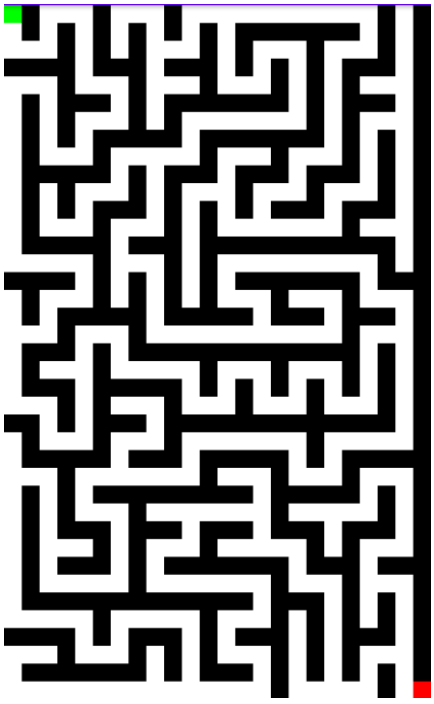
Fakultät Elektrotechnik, Medien und Informatik - Prof. Dr. Ulrich Schäfer

Labyrinth-Geschicklichkeitsspiel mit Lagesensoren und MQTT

Einführung und Aufgabenstellung

Ziel der Projektarbeit ist die Entwicklung eines Labyrinth-Geschicklichkeitsspiels als Android-App mit Lagesensoren und Möglichkeit zur Fernsteuerung per MQTT (ESP32 mit Beschleunigungssensor MPU6050 als alternativer „Game-Controller“).

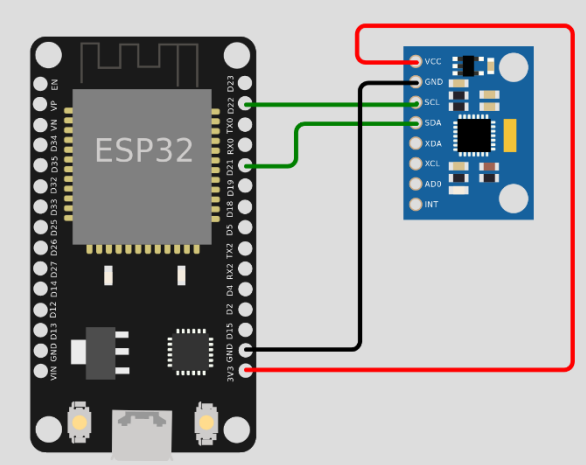
Zu Beginn wird vom Programm ein Objekt (z.B. Figur, Kugel oder Fahrzeug) am Eingang eines Labyrinths auf dem Smartphone-Bildschirm platziert. Der/die Spieler/in soll mit Hilfe der Lagesensoren durch leichtes Kippen des Smartphones (bzw. bei Fernsteuerung per MQTT des Breadboards mit dem ESP32 + Lagesensor MPU6050) das Objekt möglichst schnell bis zum Ziel (Markierung z.B. am Ausgang des Labyrinths) bugsieren. Das Objekt kann sich nur die Gänge entlang bewegen, nicht durch Mauern hindurch. Eine Runde ist zu Ende, wenn das Objekt am Ziel angekommen ist.



Die schnellsten Spieler werden mit Name, ihrer Spielzeit und Level in einer Bestenliste mit SQLite (relationales Datenbankmanagementsystem in Android) persistent gespeichert. Diese Liste soll am Ende sowie über ein Menü angezeigt werden können. Ebenso soll im Menü Sound ein- und ausgeschaltet sowie die Quelle der Lagesensoren (MQTT oder die im Smartphone eingebauten) ausgewählt und die IP-Adresse oder der Name des MQTT-Brokers eingestellt werden können. Beim Finden des Ausgangs soll die (Onboard-)LED des ESP32 per MQTT kurz ein- und wieder ausgeschaltet werden und ein kurzer Klang soll im Android-Gerät ertönen. Nach erfolgreicher Spielrunde soll es mit einem anderen, zufällig erzeugten Labyrinth weitergehen.

Auf dem ESP32 sollen Sie ferner Hardware-basierten Timer programmieren, der alle Sekunde eine MQTT-Nachricht `temp` mit dem Wert des MPU6050-Temperatursensors absetzt, unabhängig vom restlichen Code (funktioniert auch in der wokwi-ESP32-Simulation). Damit soll auf dem Display des Smartphones die seit Spielbeginn vergangene Zeit in Sekunden und die Temperatur dargestellt werden.

Anschluss der ESP-Boards



WEMOS Lolin32 Lite	ESP32 Dev Module (gelb)
`#define I2C_SDA 0`	`#define I2C_SDA 33`
`#define I2C_SCL 4`	`#define I2C_SCL 32`

in setup(): `wire.begin(I2C_SDA, I2C_SCL);`

Bitte beachten Sie, dass als Board-Profil in der Arduino-Entwicklungumgebung für die meisten am 1.6. verteilten Boards **WEMOS Lolin32 Lite** (das schmale ESP32-Board) eingetragen werden muss, nur bei den gelben, breiteren Boards **ESP32 Dev Module**. Falls versehentlich das falsche Board ausgewählt wurde, genügt kurzzeitiges USB-Stecker-Ziehen und Neuprogrammieren mit dem richtigen Profil. Im Wokwi-Beispiel oben sind andere Pins angeschlossen (die von Wokwi vorgeschlagenen „Standard“-Pins für I²C).

Hinweise zur Implementierung

Die Labyrinthe sollen von Ihnen per Java-Code zufällig erzeugt werden. Minimale Ausdehnung: jeweils 10 Gänge horizontal **und** vertikal. Position Start z.B. oben links und Ziel unten rechts. Zeichnen z.B. mit `canvas.drawRect` mit `onDraw` in eine `View`, die ein Layout in der `MainActivity` gelegt wird (vgl. Android-Buch).

Das Verwenden von Labyrinthspiel-(App)-Code aus anderen Quellen ist in dieser Projektarbeit NICHT erlaubt, Sie dürfen aber ChatGPT verwenden, um Code generieren zu lassen, zum Beispiel zum Generieren eines zufälligen Labyrinths mit einem Backtracking-Algorithmus (Achtung: der ist oft nicht auf Anhieb korrekt; Hinweise zur Verwendung von ChatGPT s. weiter unten).

Für die Erzeugung der Grafik bzw. Objekte bitte bei Bedarf in Kapitel 9 im Android-Buch von Louis und Müller nachlesen. Dort steht auch, wie Sie die Auflösung der Zeichenfläche erhalten. Nach den Zeichenanweisungen `invalidate()` nicht vergessen, erst damit wird das Neuzeichnen angestoßen. Ebenfalls im Android-Buch finden Sie Hinweise zu SQLite (Kap. 16) sowie Tonausgabe/Mediaplayer (Kap. 14). Menüs: Kap. 10, Lagesensoren (Kap. 15), Textdatei in assets anlegen und in Java öffnen: Kap 6.4.

Ähnlich wie im Raspberry-Pi-/Python-Kapitel dieser Lehrveranstaltung sollen die Programme auf Basis von Apache Paho (Java-/Android-Client) sowie einem MQTT-Broker (mosquitto) miteinander verbunden werden. Bitte Code zum Auf- und Abbau der MQTT-Verbindung sowie subscriber/publisher-Initiierung passend in die `onResume()` und `onPause()`-Methoden einfügen.

Zum ESP32-Timer: Die Verwendung von Semaphores im ESP32-Code ist nicht notwendig, außer, falls das publishen der Nachrichten aus der ESP32 Hardwaretimer-Callbackfunktion Probleme bereitet (dann

Semaphore in `loop()` abfragen und dort publishen. Falls Sie eine globale Variable in der Interrupt-Routine verwenden bzw. ändern möchten, deklarieren Sie diese bitte `volatile`. MQTT-Bibliothek für ESP32 z.B.: AsyncMQTT_ESP32; Adafruit MPU6050 für den Lagesensor.

MQTT-Bibliothek für Android (Service): https://github.com/eclipse/paho.mqtt.android

Als Broker kann ein Raspberry Pi (z.B im Labor) dienen, sein Name im Netzwerk lautet `broker`. Sie können zu Hause auch einen öffentlichen Broker verwenden oder mosquitto oder HiveMQ auf Ihrem PC oder Laptop installieren (vgl. Moodle zum Thema MQTT), die Broker-IP-Adresse ist dann natürlich die IP-Adresse ihres Rechners und der Rechner muss sich im gleichen WLAN befinden wie ESP32 und das Smartphone.

Die Gesamtlösung besteht aus drei Teilen: 1. Android-App auf Basis des Templates Basic Activity mit MQTT-Java-Client und MQTT-Service, 2. MQTT-Broker (mosquitto) im Labor bzw. auf dem PC/Laptop (hier ist nichts zu programmieren und auch nichts abzugeben) oder public, 3. MQTT-Client auf dem ESP32.

Format der MQTT-Messages: Bitte achten Sie auf das vorgegebene Format für Sensordaten (Topic **`mpu/I00`** mit sechs durch **Kommata** getrennte float-Werte für x, y, z, pitch, roll, yaw in einer Zeile, also z.B. 5.8293, 2.2331, 3.124, usw.), Topic **`temp/I00`** für den Sekundentimer mit Temperatur in Grad Celsius als float-Wert, in Gegenrichtung Topic **`finished/I00`**. Der Bezeichner nach dem Slash **I** oder **M** oder **K** steht für Studiengang II, MI oder KI, und die zweistellige Ziffer für die Moodle-/Projektgruppennummer.

Projektvorlage

In der Datei Abgabe-Vorlage.zip, von Ihnen unter dem neuen Dateinamen **Gruppe_[IMK]\d\d**.zip mit Ihrem Code und Ihrer Dokumentation im Moodle abzugeben, finden Sie die Codestruktur, die Sie verwenden sollen. Verzeichnis `android` für Android, im Verzeichnis `esp32` für den ESP32-MQTT-Client (GUI).

Abgabe

Der Quellcode ist getestet, frei von Fehlern und Warnungen nebst Dokumentation in Moodle abzugeben (Aufgabe: Projektarbeit). Der Quellcode darf keine obsoleten oder auskommentierten Anweisungen, Methoden oder Variablen enthalten. Also vor der Abgabe nochmal checken, ob wirklich alles benötigt wird, was kodiert wurde, Code bereinigen.

Eine ca. 8-seitige Beschreibung Ihrer Projektarbeit ist mit abzugeben, inkl. 2-3 Screenshots, Erklärung des Labyrintherzeugungsalgorithmus, Quellenangaben; bitte die Hinweise zu LaTeX beachten (Moodle).

Bei der Abgabe per Moodle erklären Sie, dass Sie keine Teile der Arbeit (außer angegebenen) von anderen übernommen, sondern das Programm selbst geschrieben haben.

Benutzung von fremdem Labyrinth-App/-Spiel-Code (aus dem Internet, Literatur oder von KommilitonInnen) führt zum Nichtbestehen der Prüfung. Code-Generierung mit ChatGPT ist hingegen gestattet, wenn Sie die Prompts - auch die falschen, die verworfen oder von Ihnen überarbeitet wurden - und die vollständigen Antworten des Systems (z.B. in ein LibreOffice-Dokument kopiert) mit abgeben. Sie dürfen für einzelne Problemstellungen (Netzwerk, MQTT, Lagesensoren, Mediaplayer, Menüs, SharedPreferences etc., außerdem Generierung der Labyrinth-MODELLE in Textform) Code aus dem Internet bzw. dem im Moodle angegebenen Android-Buch von Louis & Müller zu Rate ziehen, allerdings nur, wenn Sie diese Quellen auch angeben. Zur Verwaltung Ihrer Quellen wird die Software Zotero empfohlen.

Abgabe-Checkliste

s. auch Hinweise in der Datei „Checkliste zur Abgabe.txt“

Angabe in einer zip-Datei

Dateiname der hochzuladenden Datei: **Gruppe_[IMK]\d\d.zip** – Inhalt:

Quellcode, sonstige Hilfsdateien; vollständig und kommentiert, sowie die Projektdokumentation.
Verzeichnisstruktur:

```
esp32\          # ESP32-Quellcode
android\        # Android-App-Projektverzeichnis (Android Studio-Projekt),
                  OHNE den Inhalt des Ordners app\build
javadoc\        # mit Android Studio generiertes Javadoc (Verzeichnis mit vielen HTML-Dateien)
Gruppe_[IMK]\d\d.pdf  # Ihre Dokumentation (PDF, KEIN WORD!!!) mit Angabe aller verwendeten Quellen
```

ACHTUNG: Für die Abgabe WLAN-SSID und -Passwort auf Labor-WLAN zurücksetzen (ihr privater WLAN-Zugang ist für mich nicht von Interesse…)

WLAN-Zugang für MQTT im Labor

SSID: `ki-lokal`

Passwort: `dc-ki-2022+`

Hostname des MQTT-Brokers (Raspberry Pi): `broker (IP-Adresse 192.168.1.12 oder .13)`

Bewertungsgrundlagen

- Projektarbeits-Beschreibung, Quellenangaben, Aufbau
- Usability der App/GUI
- Code: Korrektheit, Vollständigkeit, Eleganz, Lösungswege
- Code-Dokumentation: inline (ESP32-Code), inline+Javadoc (Java/Android), – Javadoc-Beispiele s. Quellcode der MQTT-Java-Bibliotheken oder Quellcode von Android
- Kurzvortrag mit Demonstration der Anwendung durch das jeweilige Team am 5./6. Juli 2023

Nach der Abgabe bitte regelmäßig email checken, falls Rückfragen nötig sind.

Viel Spaß & viel Erfolg!