Internet of Things

Studienjahr 2023, Prüfungsaufgaben

Prüfungsformalitäten

Bearbeitung in Eigenleistung

Alle Aufgaben sind in Eigenleistung zu bearbeiten. Die abgegebene Lösung muss den persönlichen Wissensstand widerspiegeln und soweit wie möglich einen individuellen Lösungsweg erkennen lassen. Zwar ist es durchaus erlaubt, sich in Form von Rückfragen oder Internetrecherche Hilfe zu holen. Die Lösung muss aber dennoch selbst ausgearbeitet worden sein, um gewertet werden zu können. Fremde Quellen müssen kenntlich gemacht werden (gilt auch für Google-Suchen, Wikipedia-Recherche, … wobei ein Link-genügt).

Abgabe

Die Abgabe erfolgt in Moodle bis zum Ende des Semesters bis zum Tag vor Beginn der nächsten Praxisphase. Abweichend davon kann eventuell über Moodle ein anderer Stichtag kommuniziert werden.

Aufgabenstellung

Aufgabe 1: Nebenläufigkeit in Python (6 + 4 Punkte)

a) Mit Hilfe der Module `multiprocessing`, `threading` und `asyncio` der Python-Standardbibliothek können nebenläufige Aufgaben auf unterschiedliche Weise realisiert werden. Beschreiben Sie für jedes der drei Module kurz, ob die Nebenläufigkeit durch Prozesse, Threads oder Co-Routinen erfolgt und welche Art der Parallelausführung sie deshalb unterstützen. Nennen Sie zusätzlich jeweils einen typischen Anwendungsfall.

b) Die interne Steuerung eines IoT-Devices wurde mit Co-Routinen programmiert. Innerhalb einer Co-Routine kommt es jedoch immer mal wieder vor, dass die Methode zum Auslesen eines Sensors drei Sekunden benötigt, um den Sensorwert zu ermitteln. Welches Problem ergibt sich daraus, wenn es sich bei der Methode um eine so genannte „blockierende Methode“ handelt und wie könnte es gelöst werden?

Aufgabe 2: Hardwarenahe Programmierung (20 Punkte)

In der Vorlesung haben wir die Bibliotheken `gpiozero`, `tkgpio` und `asyncio` genutzt, um ein selbstfahrendes Fahrzeug zu programmieren. In mehreren, quasi-parallel ablaufenden Tasks werden dabei verschiedene Sensoren genutzt, um die physische Umwelt zu überwachen und daraus Aktionen abzuleiten. In dieser Aufgabe soll in Anlehnung daran ein anderes Fallbeispiel implementiert werden: Die Steuerelemente, Sensorik und Aktorik eines einfachen PKW: Blinker, Fahrtlicht, Scheibenwischer und Hupe.

Am einfachsten funktioniert dabei die Hupe, simuliert durch einen Button und einen Buzzer:

- **Button nicht gedrückt:** Hupe aus
- **Button gedrückt:** Hupe ein

Der Blinker soll hingegen durch zwei LEDs sowie drei Buttons simuliert werden:

- **Kein Button gedrückt:** Blinker aus
- **Button 1 gedrückt:** Links blinken
- **Button 2 gedrückt:** Rechts blinken
- **Button 3 gedrückt:** Warnblinker (beide LEDs)

Das Fahrtlicht soll durch zwei LEDs, pulsweitenmoduliert jedoch nicht getrennt schaltbar, sowie einen Lichtsensor simuliert werden. Hinzu kommt ein Wahlschalter, der mangels eines entsprechenden Elements in `tkgpio` als Potentiometer an einem MCP3008 A/D-Wandler nachgebaut werden kann. Der A/D-Wandler liefert hierfür je nach Potistellung eine Kommazahl zwischen null und eins zurück, so dass der Wertebereich einfach in gleiche Intervalle eingeteilt werden kann:

- **Nullstellung:** Licht aus
- **Intervall 1:** Abblendlicht (schwache Beleuchtung)
- **Intervall 2:** Fernlicht (helle Beleuchtung)
- **Intervall 3:** Lichtsensor schaltet das Abblendlicht ein oder aus

Nach dem gleichen Prinzip sollen zwei Servomotoren zur Simulation der vorderen Scheibenwischer sowie zwei Potis zur Simulation des Wahlschalters und eines Regensensors genutzt werden:

- **Nullstellung:** Scheibenwischer aus
- **Intervall 1:** Intervallwischen langsam
- **Intervall 2:** Intervallwischen schnell
- **Intervall 3:** Wischgeschwindigkeit durch Regensensor gesteuert

Beispiele zur Simulation der benötigten Hardwareelemente finden Sie an folgender Stelle im Quellcode der Bibliothek `tkgpio`: https://github.com/wallysalami/tkgpio/tree/master/docs/examples. Sie müssen dabei lediglich beachten, dass die meisten Beispiele statt einer JSON-Datei zur Konfiguration der Hardwareelemente diese als JSON-String direkt im Pythoncode definieren.

Die Klasse `gpiozero.MCP3008` wird an folgender Stelle beschrieben: https://gpiozero.readthedocs.io/en/stable/api_spi.html#mcp3008

a) Zeichnen Sie zunächst eine Skizze, aus der die nebenläufigen Tasks der Anwendung hervorgehen. Erläutern Sie dabei zu jedem Task in ein/zwei Sätzen seine Funktion.

b) Legen Sie ein neues Python-Projekt an oder kopieren Sie den Quellcode aus der Vorlesung und entfernen dabei alle Quellcodes, die für diese Aufgabe nicht benötigt werden. Das neue Projekt sollte zunächst nur eine Datei `requirements.txt` zur Installation der Bibliotheken, ein Startskript (im Beispiel `carbot_sim.py` genannt), ein Hauptmodul (im Beispiel `carbot_sim/__init__.py` und `carbot_sim/main.py` genannt) sowie eine JSON-Datei zur Beschreibung der Hardwarebausteine (im Beispiel `tksim_windows.json` bzw. `tksim_linux.json` genannt) beinhalten.

c) Schauen Sie sich die oben verlinkten Beispiele im `tkgpio`-Quellcode an und versuchen Sie damit, die Hardwarebausteine in der JSON-Datei zu beschreiben.

d) Programmieren Sie nun die einzelnen Tasks aus und rufen diese im Hauptmodul auf.

Anhand der Bildschirmausgabe sollte die korrekte Funktionsweise des Programms beobachtet werden können. Die Benotung erfolgt hierbei nach folgenden Kriterien:

- Vollständige Umsetzung der Funktionsbeschreibung
- Fehlerfreiheit durch manuellen Test aller Funktionen (keine fachlichen Fehler)
- Fehlerfreiheit durch Betrachtung des Quellcodes (keine technischen Fehler)
- Lesbarkeit, Strukturierung, Formatierung und Kommentierung des Quellcodes (orientieren Sie sich hierfür einfach an den Beispielquellcodes zur Vorlesung in GitHub)