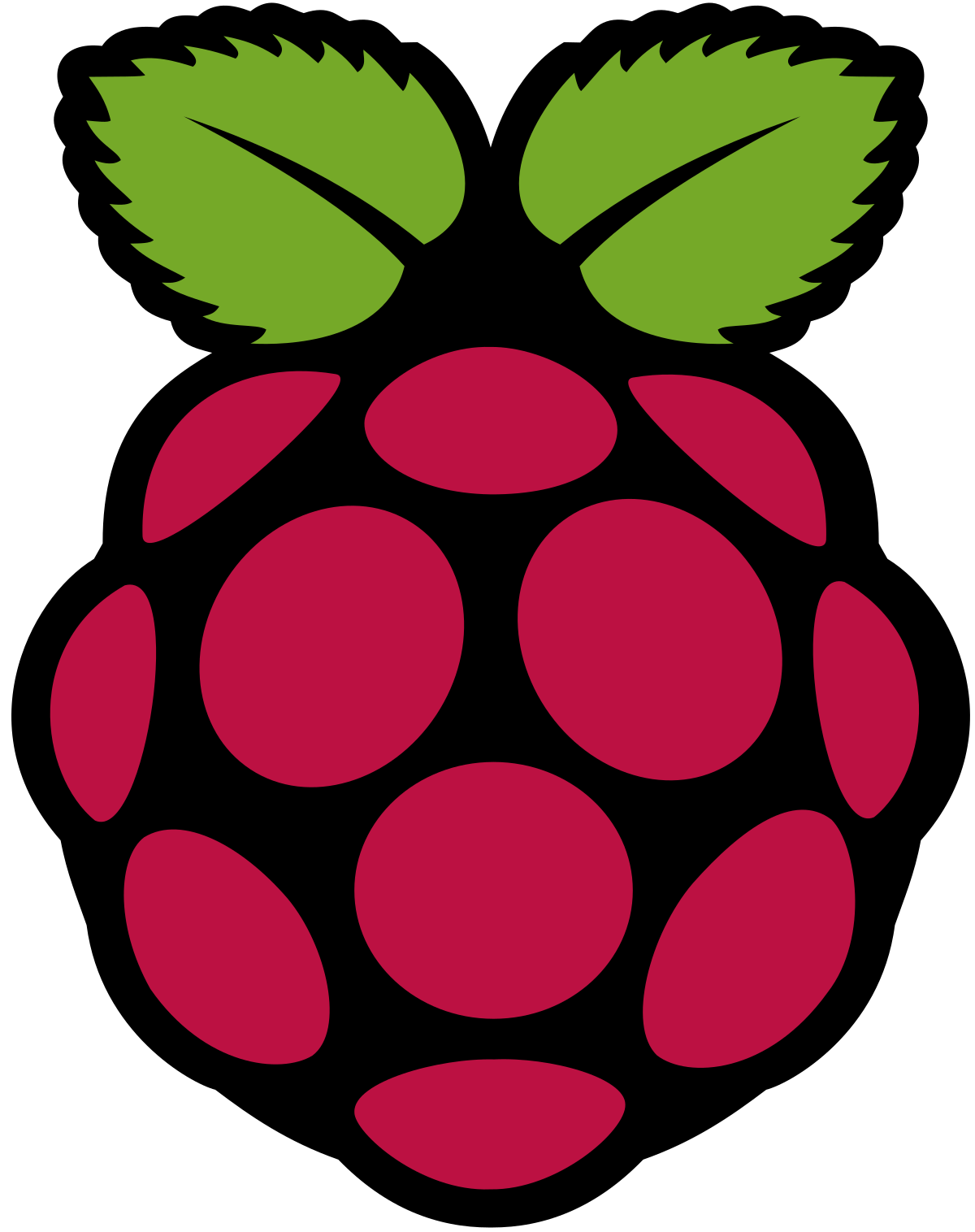
Heinrich-Hertz-Schule Karlsruhe

RFID-projekt

Pascal L, Nico



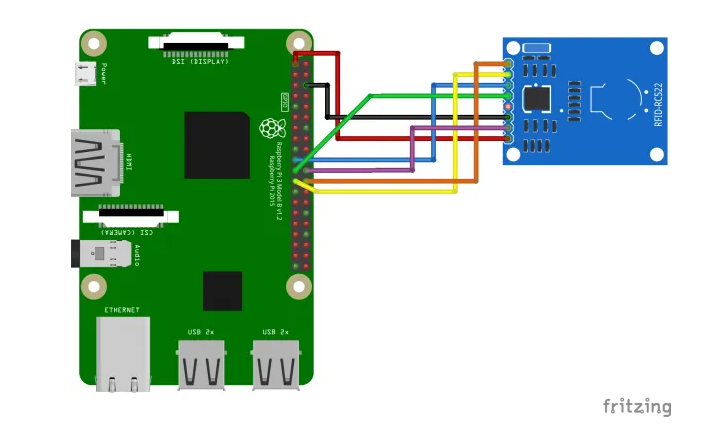
Projekt Ziel

Unser Ziel am Ende der Projektzeit ist es, eine Art Rätselspiel programmiert zu haben. Der Anwender bekommt durch Starten der Anwendung ein Rätsel angezeigt, seine Antwort zu diesem Rätsel wird dann als Eingabe von dem Programm erwartet. Die Eingabe wird auf den RFID-Chip/-karte geschrieben. Danach wird die Antwort auf dem/der Chip/-karte mit der in dem Programm hinterlegten korrekten Antwort auf dieses Rätsel abgeglichen. Wenn die Antwort korrekt angegeben wurde bekommt der Anwender eine entsprechende Ausgabe angezeigt, dasselbe gilt dann auch für eine inkorrekte Eingabe. Die Eingabe des Anwenders wird mit Hilfe des MQTT-Protokolls an den Raspberry Pi geschickt, dort wird die Antwort auf den/die Chip/-karte geschrieben. Die Antwort wird dann auf dem PI eingelesen und per MQTT wieder zurück an die Anwendung geschickt.

Verwendet Hardware

Zur Umsetzung dieses Projektes benötigen Sie einen funktionstüchtigen Raspberry Pi. In der eigenen Durchführung wurde von uns ein Raspberry Pi 3, Model B verwendet.

Zusätzlich wird ein RFID-Scanner mit Chip/-karte benötigt. Wie der RFID-Scanner korrekt angeschlossen werden kann siehe Bild:



Durchgeführte Arbeiten

Benötigte zusätzliche Software wurde installiert: python3 und eine MFRC522 Bibliothek:

sudo apt-get install python3.6

Pip3 install mfrc522 --upgrade

Im ersten Schritt muss SPI auf dem Pi aktiviert werden. Hierzu kommt man über sudo raspi-config in das Konfigurationsmenü des Pi’s. Nun auf den Punkt *3 Interface Options Configure connections to peripherals*, daraufhin *P4 SPI Enable/Disable automatic loading of SPI kernel module*. Es erscheint ein neues Fenster, hier mit Enter <Ja> bestätigen. Die folgende Bestätigung einfach mit Enter wegdrücken. Danach kann über <Finish> das Konfigurationsmenü wieder geschlossen werden.

Daraufhin haben wir zwei Python Skripte geschrieben um die Funktionalität des RFID-Scanners und des/der Chip/-karte zu testen. Zum einen haben wir das Skript write.py geschrieben (siehe Anlagen).

Mit diesem Skript wird getestet ob man Chip/-karte mit Daten beschreiben kann.

Zusätzlich haben wir das Skript read.py geschrieben (siehe Anlagen)

Mit diesem Skript wird die vorherig eingegeben Eingabe aus Chip/-karte ausgelesen und dann auf der Konsole angezeigt.

Mit diesen Skripts konnten wir testen, ob unser RFID-Device funktioniert. Diese Tests waren dann auch erfolgreich. Wir können Chip und Karte nach Belieben beschreiben und auslesen.

Als nächstes haben wir uns darangesetzt, das Beschreiben von Chip/-karte auszulagern. Wir wollen an einem anderen System einen Text eingeben und diesen mittels MQTT an den Pi schicken. Um das zu realisieren haben wir angefangen ein Programm in JAVA über die Eclipse IDE zu schreiben. Wir haben ein neues Projekt samt Package erstellt. In diesem Package haben wir dann schon mal drei Klassen angelegt: Write.java, Read.java und Main.java. Die Namen sollten für sich sprechen: Write.java und Read.java sollten die Funktionalitäten von Ihren Namensvettern in Python übernehmen. Dann haben wir uns überlegt, dass wir ja auch noch eine Verbindung zu dem Pi mittels MQTT herstellen müssen. Dafür haben wir angefangen in der Write.java Klasse eine Publish-Connection zu erstellen. Nach ein paar oberflächlichen Tests ist es uns allerdings nicht gelungen, die Verbindung erfolgreich herzustellen. Da wir allerdings keine Zeit verschwenden wollten haben wir uns dann dazu entschieden mit der Bearbeitung der Read.java Klasse fortzufahren. Wir haben in der Klasse die nötigen Schritte vorgenommen, um ein Topic eines MQTT-Brokers subscriben zu können. Die Connection hat auch funktioniert was uns motivierte den Counterpart auf dem Pi zu implementieren: Ein Skript welches den Inhalt des/der Chip/-karte ausliest und über das Topic published. Dafür haben wir das Skript sendInputTo.py erstellt (siehe Anlagen).

Nach 6-Wöchiger Abstinenz aus der Schule mussten wir uns erst mal wieder mit den gegebenheiten unseres Projektes im Unterricht vertraut machen. Wir haben also den Pi angeschlossen und SPI in den Interface Options enabled (s.o.). Nachdem der erste Test unseres damals erstellten sendInputTo.py Skriptes nun fehlschlug, haben wir uns daran gesetzt sowohl die Python Skripte auf dem Pi als auch den Java Quellcode zu bug-fixen und debuggen. Wir konnten einige kleine Fehler beheben und das Fehlerhandling generell verbessern.

Nun wollten wir der Identität des ‘Rätsel-Guesser‘ näher kommen und haben uns daran gesetzt den String, welchen wir über MQTT vermittelt bekommen, mit einem in Java hinterlegten String zu vergleichen. Die ersten Tests waren sehr simpel: Wenn der übermittelte String mit dem hinterlegten String übereinstimmte, solle eine entsprechende Ausgabe erfolgen. Wir sind dabei allerdings auf unvorhergesehene Probleme gestoßen, was die Verarbeitung des übermittelten Strings angeht. Glücklicherweise konnten wir diese Probleme dann doch noch lösen und der Vergleich der beiden Strings funktionierte. Der nächste Schritt für uns war nun tatsächlich Fragen zu generieren und zu diesen Fragen antworten zu hinterlegen. Dafür haben wir die Questions.java Klasse erstellt. In dieser Klasse befindet sich eine Methode welche mehrere Fragen hinterlegt hat, und zu jeder Frage die zugehörige Antwort speichert. In der Read.java wird diese Klasse mit einer neuen Methode aufgerufen, welche eine zufällige Zahl generiert. Je nachdem welche Zahl generiert und zur Initialisierung der Questions.java Klasse verwendet wird, wird eine andere Frage ausgewählt. Der hinterlegt String welcher die Lösung darstellt wird weiterhin mit der Eingabe des Users verglichen und man bekommt eine entsprechende Ausgabe in der Konsole ausgegeben. Die genaue Implementierung kann natürlich in den Klassen genauer angesehen werden.

Da wir nun allerdings wieder fast die Deadline für Arbeiten an dem Projekt erreicht hatten, haben wir uns dazu entschieden als letzte Arbeiten das Fehlerhandling weiter zu verbessern, den Java-Code gemäß clean-code anzupassen und ein paar ‘Quality-of-Life‘ Anpassungen vorzunehmen. Letzteres umfasst unter anderem, dass das Java-Programm sich nun beendet, nachdem die Eingabe des Users erfolgreich verglichen wurde. Dadurch entfällt das ständige Terminieren der Anwendung was bis dahin noch nötig war. Weiterhin haben wir ein bash-Script geschrieben, welches auf dem Pi ausgeführt wird und die write.py und sendInputTo.py (s.o.) nacheinander aufruft, sodass der Benutzer die beiden Skripte nicht mehr selbst anstoßen muss. Sondern nur noch das neue runProject.sh Skript.

Zukünftige Features

1. Java-Code:
   1. Publisher-Connection fertig programmieren

Ein zukünftiges Feature wäre, die Eingabe des Users auch in Java erfolgen zu lassen, sodass man nicht ständig zwischen den zwei Systemen wechseln muss.

* 1. Als weiteres zukünftiges Feature könnte man einbauen, dass der User automatisch eine neue Frage bekommt ,so dass das Programm nicht jedes Mal neu gestartet werden muss und man immer weiter spielen kann. Das Ende des Programms kann man dann z.B. an einer gewissen Punktzahl festmachen oder an einer bestimmten Anzahl an Versuchen die dem Anwender zur Verfügung stehen.
  2. Code-Polishing

Das weitere verfeinern und anpassen des Quellcodes wäre auch defintiv noch ein TODO. Während Fehlerhandling, clean-code, etc. ausreichend implementiert worden ist, gibt es in diesen Feldern klaren Platz zur Verbesserung.

Protokollblatt

Gruppenmitglieder: Pascal L, Nico

|  |  |  |
| --- | --- | --- |
| **Datum** | **Name** | **Aufgabe / Erledigtes / Probleme** |
| 26.04.2022 | Pascal L, Nico | Wir haben uns Gedanken über unser Projekt gemacht und Überlegt was wir machen könnten bzw. wir haben uns auf ein Thema geeinigt. |
| 27.04.2022 | Pascal L | Ich habe im Internet nach Beispielen recherchiert |
| 27.04.2022 | Nico | Ich habe mir ähnliche GitHub Projekte angeschaut |
| 28.04.2022 | Pascal L | PI Updates gemacht sowie die benötigen Software Installiert |
| 28.04.2022 | Nico | PI mit dem RFID-Scanner aufgebaut |
| 28.04.2022 | Pascal L, Nico | Python Skripte geschrieben sowie Fehlerbehebungen gemacht, danach haben wir die Funktion getestet |
| 03.05.2022 | Pascal L, Nico | Als erstes haben wir etwas mit dem RFID-Scanner getestet danach haben wir haben angefangen die Dokumentation zu schreiben. |
| 04.05.2022 | Pascal L, Nico | Dokumentation weiter geschrieben |
| 05.05.2022 | Pascal L | sendInputTo.py erarbeitet |
| 05.05.2022 | Nico | Java Klassen erarbeitet |
| 21.06.2022 | Pascal L, Nico | Bug-Fixing, Fehlerhandling verbessert |
| 22.06.2022 | Pascal L, Nico | Questions.java implementiert und erste Tests vorgenommen |
| 23.06.2022 | Nico | Code entsprechend Clean-Code angepasst, Fehlerhandling weiter verbessert, weitere Fragen ausgedacht |
| 23.06.2022 | Pascal L | Weitere Fragen ausgedacht, python Skripte angepasst, shell Skript geschrieben |
| 25.06.2022 | Pascal L, Nico | Dokumentation fertig gestellt |

Quellen

<https://upload.wikimedia.org/wikipedia/de/thumb/c/cb/Raspberry_Pi_Logo.svg/1200px-Raspberry_Pi_Logo.svg.png>

<https://tutorials-raspberrypi.de/raspberry-pi-rfid-rc522-tueroeffner-nfc/>