Technische Universität München
Fakultät für Informatik

Exercise Sheet 1: Hardware Constraints and Setup

Welcome to the course on Advanced Topics of Software Engineering! Throughout the semester, we will learn about the difficulties of software design, coding and deployment and why "good" software engineering is essential.

Therefore, we will develop a web-based Delivery Shipping System, *ASEDelivery*, with a modern microservice architecture, that incorporates aspects of programming scalable distributed as well as embedded systems, i.e., different **contexts of software engineering**. Fig. 1 depicts the business process which our system needs to realize.

For the practical exercises, we will work with the embedded system, the component that brings most of the fun to our project. You will practice how to setup the Raspberry Pi, interact with an LED light as the actuator, and read/write RFID tags with an RFID Scanner as the sensor of the system.

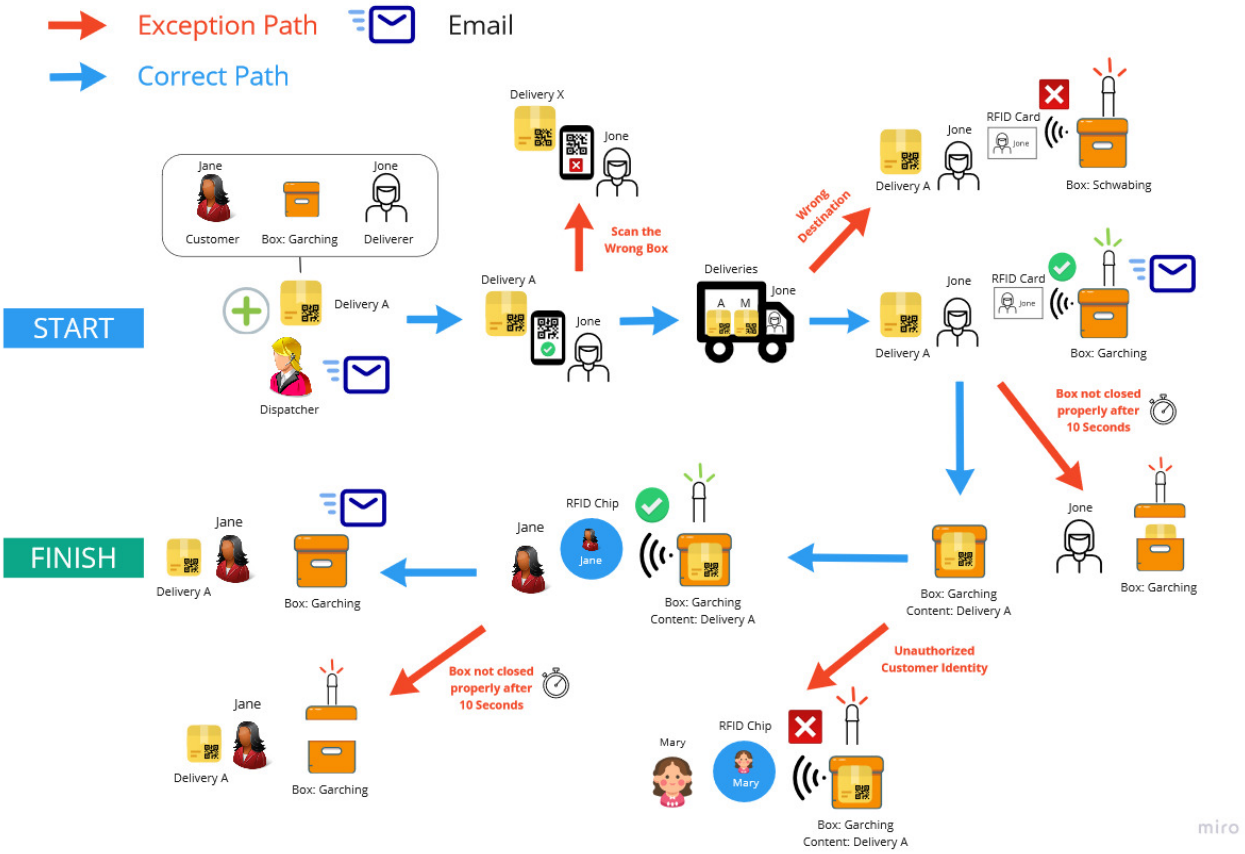

Figure 1: The business process of the ASEDelivery System

Theoretical Exercise

Understanding the requirements, concerns, and goals of stakeholders is a decisive factor in designing software systems. A successful software system is not only functional and user-friendly, but also be highly maintainable, secure, and extensible. Therefore, software architects need to thoroughly analyze the needs, constraints, and visions of the stakeholders before starting to design the foundational structure of the system.

Exercise 1: Identifying Concerns of Stakeholders

Given the descriptions of the course project, identify the stakeholders and their potential concerns (Chapter 2.1.1, Slide 8). How would you address or mitigate these concerns?

Exercise 2: Constraints of Embedded Systems

Developing projects involving embedded systems can be quite fun, until you face their constraints. Given the following hardware components in our systems:

- Raspberry Pi 3 Model B
- RFID Tags

1. Identify the limitations of each hardware component. Briefly explain each limitation in less than four sentences.
2. How do these limitations affect your system?
3. What should you consider when implementing your project, given these constraints?

Exercise 3: Project Management using GitLab Board

Effective project management is crucial to the success of a software system. This exercise describes the key features of the Gitlab Board, which assists your team in planning, assigning, and tracking tasks.

Gitlab Board supports the creation of columns, which represent the stages in your system development process. You can create a Scrum Board if your team decides to apply the Scrum practice or choose a different agile workflow that suits your development process. In this exercise, we will demonstrate how to create a Scrum board for a project in GitLab.

1. Enter the Issues/Boards interface and create a new Board.
2. Specify the name and color for each column of your Board. Click on the *Edit board* button on the top right corner of the board interface, find the **Labels** section, and click on the *Edit* button. Create a new label with color and name.
3. Create a new column from the labels. Click on the *Create list* button on the top right corner. Choose *label* and create.

After creating the labels, you can insert items into each column. Each item can be treated as a task or user story, depending on the organization of your team. You can specify the priority and assigned member(s) for each item, as Figure 2 shows.

We hope the GitLab Board provides convenient support to foster your team communication and project management.

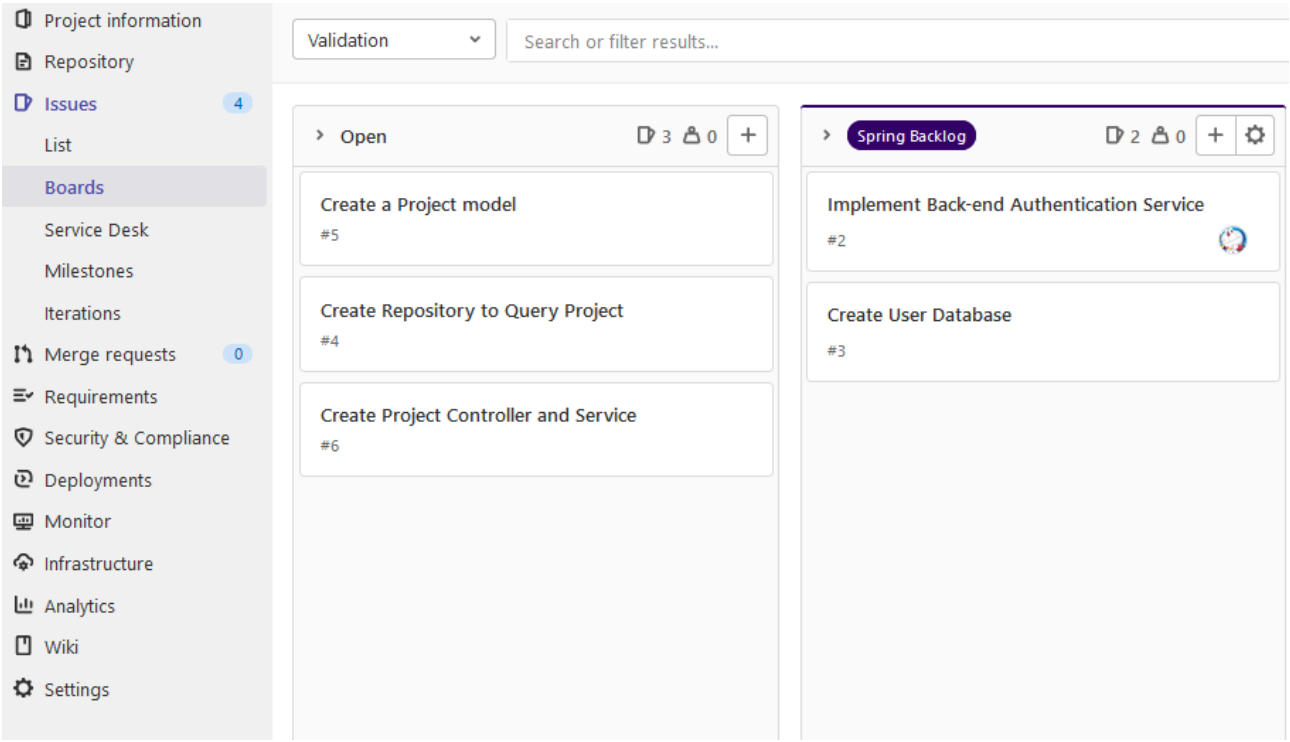

Figure 2: A Kanban board with opened tasks

Practical Exercise

This week's practical exercises will introduce you to how to set up a Raspberry Pi 3 Model B[1] (v1.2) and how to interact with external components such as an LED or an RFID Scanner by controlling the GPIO (general purpose input/output) pins with a Python script. In this course, we are going to be using Pi with the *Raspberry Pi OS* Linux distribution. To install it, you can use the Raspberry Pi Imager (https://www.raspberrypi.org/software/).

Raspberry Pi is a pocket-size computer that gives you the ability to easily interact with the outside world. In your project, you will use the Raspberry Pi to control the interactions on a delivery box. These interactions include reading an RFID token for authentication, lighting a LED to indicate status such as authentication unsuccessful, and checking whether a box is closed properly by measuring the light intensity. To program our Pi for handling such interactions, we will be using Python.

The Python script that will be running inside the Raspberry Pi is responsible for controlling the external components by utilizing the GPIO pins on the Pi. Depending on the model, the GPIO header includes 26 or 40 pins. The Raspberry Pi 3 Model B includes 40 pins and the layout of them can be seen in Figure 3. In this figure, the pins that are marked green are the ones that can be used for general input and output interactions. The rest of the pins have special purposes (see https://projects.raspberrypi.org/en/projects/physical-computing/1).

We will get started with connecting to a Raspberry Pi and then learn how we can interact with it through a Python script. Remember that any Python program that one can execute on their "non-embedded" systems, can be directly ported to Raspberry Pi without compromising the functionality. This is the backbone of *portable* architecture.

Exercise 4: Connect to the Raspberry Pi

1. You can access your Raspberry Pi by either connecting it to a display via an HDMI cable, or you can just SSH (Secure Shell) into it. If you are *not* going to use SSH, click on the command line icon on your Raspberry Pi desktop and skip to step 7.

2. When you want to SSH into the Raspberry Pi, make sure that your personal computer and the Raspberry

[1] https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[2] https://www.raspberrypi-spy.co.uk/2012/06/simple-guide-to-the-rpi-gpio-header-and-pins

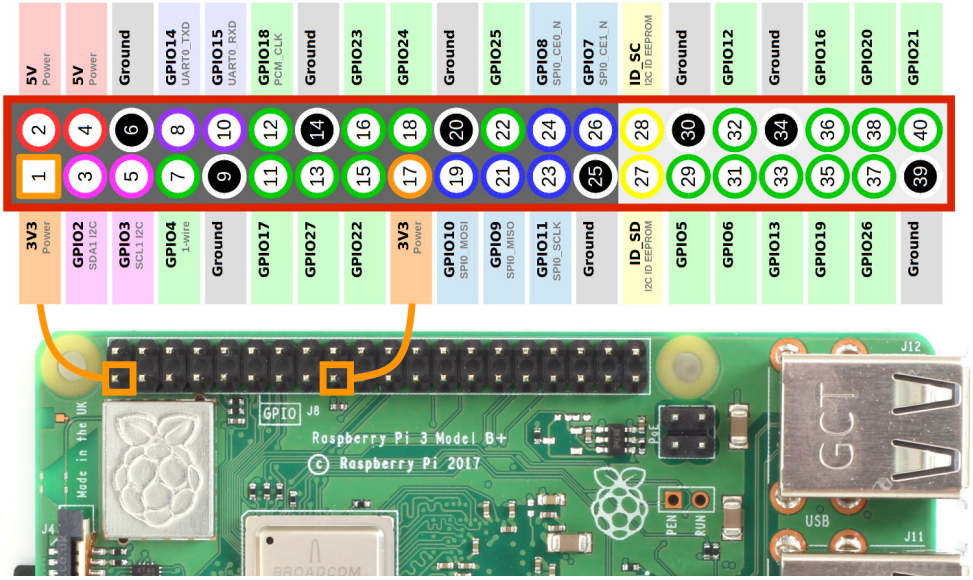

Figure 3: GPIO pins on a Raspberry Pi.[2]

Pi are in the same network.

3. In default settings, the SSH connection is not enabled in Raspberry Pi due to security concerns. To enable it, go to Menu ⟶ Preferences ⟶ Raspberry Pi Configuration. On the Interfaces tab, you will find the SSH option. Go ahead and set it to *Enabled*.

4. To SSH into your Raspberry Pi, you need to know its IP address and the username which is by default *pi*. Run the following command to get the IP address:
   `hostname -I`⟶192.168.x.x

5. You are now ready to connect via SSH. Windows users might want to install the PuTTY SSH client (http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html). Figure 4 shows an example PuTTY configuration. Linux and OS X users can run SSH from their terminals:
   `ssh pi@<IP address>`
   On your first connection attempt, you are going to face a message indicating that the authenticity cannot be established. It will ask you if you would like to continue. Type `yes` and press *Enter*.

6. Upon a successful connection, you will see that your regular command line prompt is replaced with `pi@raspberrypi: ~$`. You are now at a Linux command line!

7. You can use `cd` for changing into a directory, `ls` for listing contents of a directory, `nano` for creating/editing a file, and `cat` for displaying the content of a file.

8. Update the Raspberry Pi to get the latest versions of the software.
   ```
   sudo-apt get update
   sudo-apt get upgrade
   sudo apt-get install python3-dev python3-pip
   ```

9. Now, change to the *Desktop* directory and create a folder named *ASE_Exercises*. Inside this folder, create another folder, *Week_1*, and inside this folder initialize two empty files named *led_exercise.py* and *rfid_exercise.py*. We will be using these files in the following exercises.

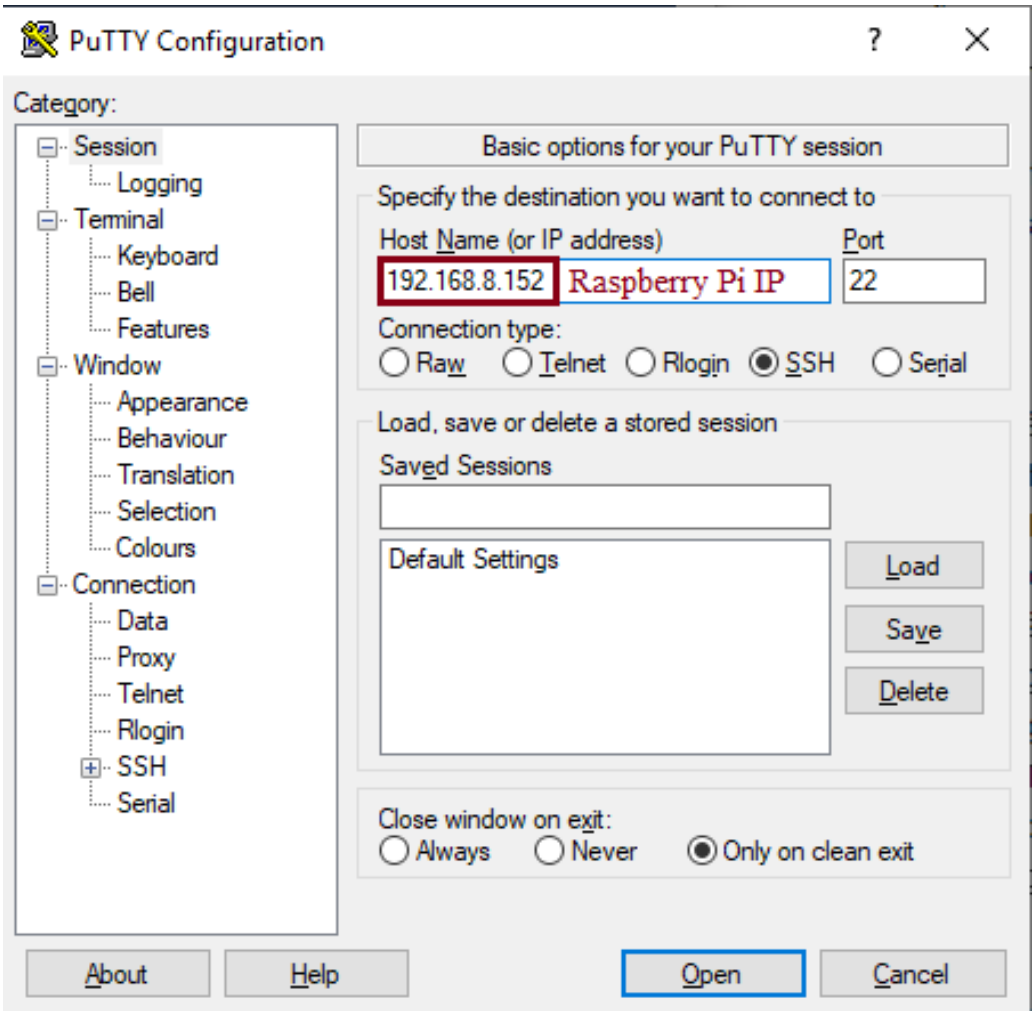

Figure 4: Raspberry Pi SSH Configuration in PuTTY.

Exercise 5: Lighting an LED using Python and Raspberry Pi

Now that you have access to your Raspberry Pi's terminal, we can start working on our first Python program to interact with a GPIO component. In this exercise, you will learn how to switch an LED on and off which is connected to a GPIO pin on our Raspberry Pi. For this matter, please use your *three-legged* LED.

1. To test that the LED is working, try connecting the negative end (middle-long leg) of your LED into a ground pin (e.g. 6) and the positive end (one of the side legs) into a resistor and the resistor to the pin 1 on your Raspberry Pi which provides 3V3 power. **Make sure to use a resistor (at least 50 Ω) in series with the positive end of your LED, otherwise, you may burn it**. Once your setup is ready, the LED should light up. Since the LED is connected to a power source directly (3V3), it will stay lit up unless it is disconnected from power. Figure 5 shows an example setup without using a breadboard. **If you switch the connected positive leg, the color of the LED should change.**

2. To control the LED through a Python script, let's move it to a general-purpose GPIO pin (pins marked in green on Figure 3). For this purpose, disconnect the pin that is connected to power and connect it to a GPIO pin of your choice. Make sure to note down the pin number as you will need it in your Python script. Once you move it, the LED will no longer be lit up.

3. Now that the hardware setup is ready, open the *led_exercise.py* file that you have created. To interact with the GPIO pins on the Raspberry Pi, you have to first import the *RPi.GPIO* library and set the numbering system for our pins. You can either use the actual pin numbers which range from 1-40 by setting the mode to *BOARD* or you can use the Broadcom GPIO numbers by setting it to *BCM*. **If your program is not interacting with the GPIO pins as you expect, the numbering mode could be the reason for it.**

[3]https://projects.raspberrypi.org/en/projects/physical-computing/2

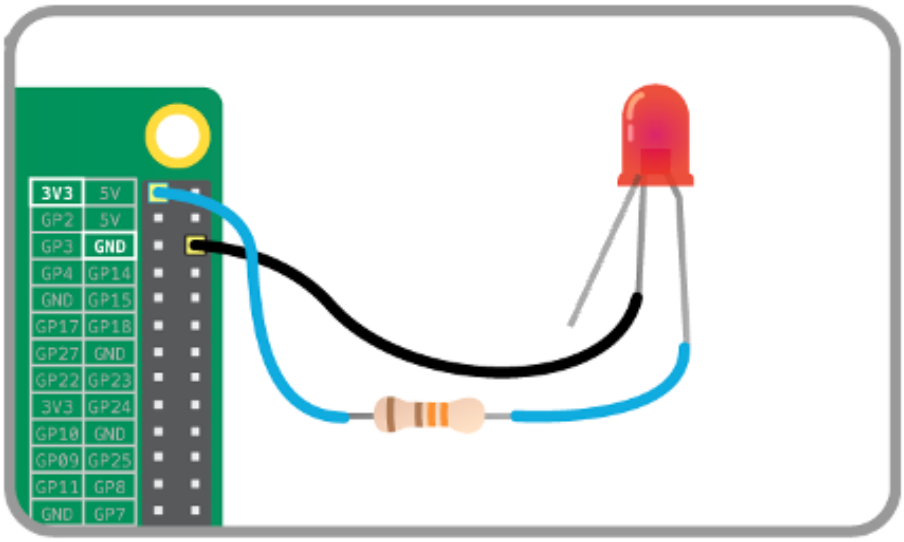

Figure 5: Lighting an LED using a Raspberry Pi.[3]

```
import RPi.GPIO as GPIO

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
```

4. Finally set the GPIO pin to which the LED is connected as an output. For this example, we are going to assume that the pin is connected to GPIO 17 (see Figure 6). You can now control the LED by setting the value of the pin as *LOW* or *HIGH*. In the following example code, we initially set the LED as turned off and after sleeping the program for 3 seconds we turn it on for 5 seconds and turn it off again. Please copy the code into your *led_exercise.py* script and run it with `python led_exercise.py` command.

```
import RPi.GPIO as GPIO
from time import sleep

GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
GPIO.setup(17,GPIO.OUT, initial=GPIO.LOW)

sleep(3)
GPIO.output(17,GPIO.HIGH)
sleep(5)
GPIO.output(17,GPIO.LOW)
```

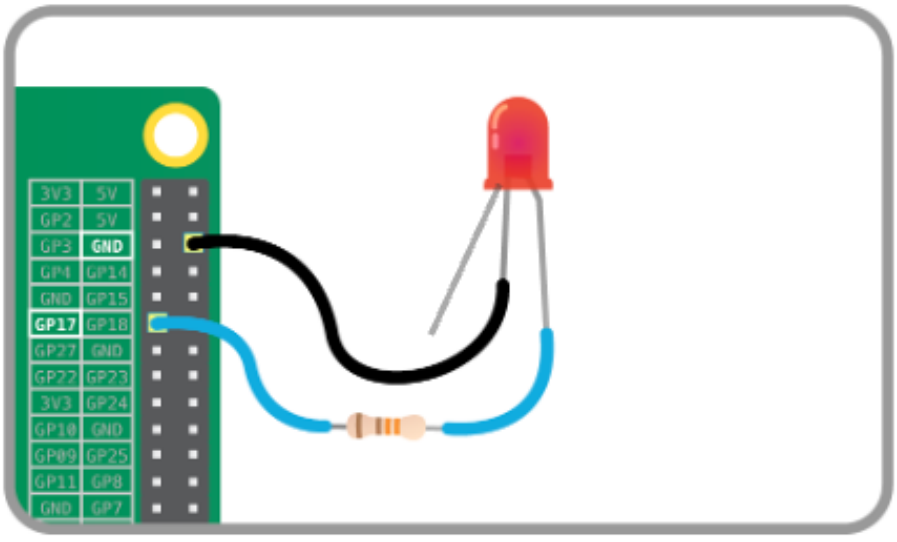

Figure 6: Controlling an LED through a GPIO pin.[4]

[4]https://projects.raspberrypi.org/en/projects/physical-computing/2

Exercise 6: Read/Write Operations on an RFID Chip using Python and Raspberry Pi

The next GPIO component that we are going to learn how to interact with is an RFID chip. We are going to use this chip for reading an RFID token and updating its content. The model of the chip that we are going to utilize is RFID-RC522.

1. Please copy the exact wiring shown in Figure 7 to connect the RFID-RC522 to your Raspberry Pi. Once connected correctly, you will see a red light appearing on the chip.

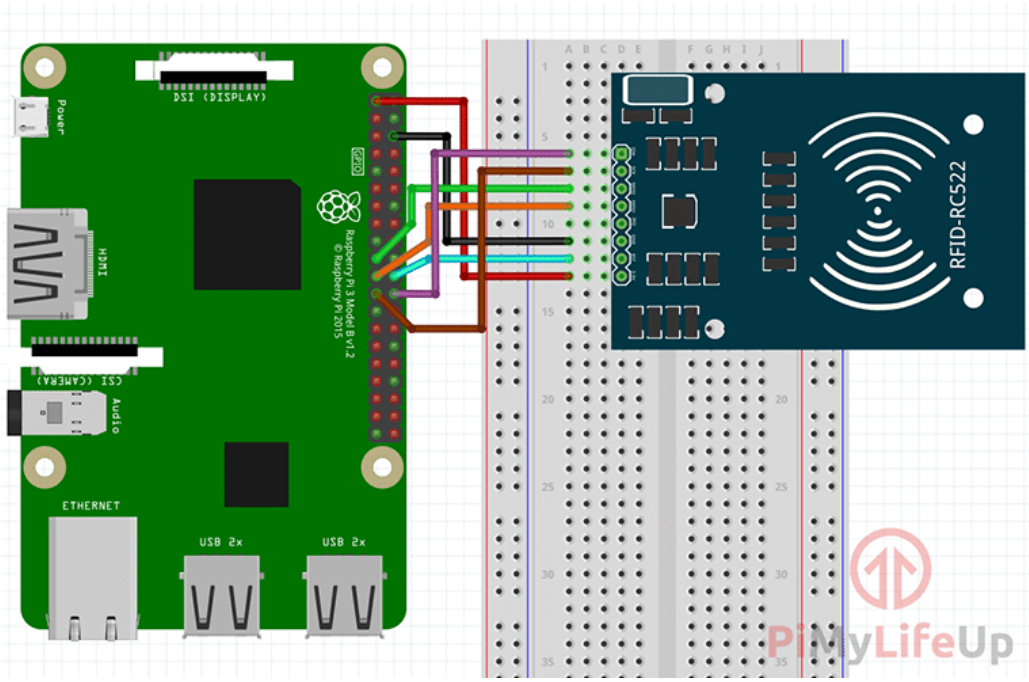

Figure 7: Wiring the RFID-RC522 to a Raspberry Pi.[5]

2. To interact with the RFID chip, you have to first enable the Serial Peripheral Interface (SPI) in your Raspberry Pi. Open a command line and run `sudo raspi-config`. On the screen that comes up, go to *5 Interfacing Options* by using the arrow keys. Once you enter into this option, choose **P4 SPI** on the new screen that appears. Now, it will ask you whether you would like to enable the SPI interface. Choose `yes` and press *Enter*. Wait until you see **The SPI interface is enabled** message on the screen. Now, go back to the command line and restart your Raspberry Pi with `sudo reboot` command.

3. Now, let's proceed to installing the MFRC522 library.[6]
   `sudo pip3 install mfrc522`

4. Once the library is installed, you can import it with:

```
1  from mfrc522 import SimpleMFRC522
2  reader = SimpleMFRC522()
```

5. To read an RFID token, you can use the `read()` method of the `reader` object that we have initialized on Line 2 of the code above. Calling this method will return the *id* and the *text* content of the scanned token.

```
1  from mfrc522 import SimpleMFRC522
2  reader = SimpleMFRC522()
3  while True:
4      id, text = reader.read()
5  except KeyboardInterrupt:
6      # GPIO must be cleaned up once you exit the script
7      # Otherwise, other scripts may not work as you expect
8      GPIO.cleanup()
9      raise
```

6. You can also overwrite the *text* field of a token by using the `write()` method of the `reader` object. **During this operation, the token must be held on the scanner.**

[5] https://pimylifeup.com/raspberry-pi-rfid-rc522/
[6] https://github.com/pimylifeup/MFRC522-python

7. Now, open the *rfid_exercise.py* file and work within your group to write a Python script that waits for a token to be scanned, prints its *id* and *text*, and then asks the user if they want to overwrite the content. Once overwritten, sleep the program for 5 seconds and then make it wait for a token to be scanned again.

8. Test whether the overwrite is successful by re-scanning the token.

Exercise 7: A Simple Authentication System with Raspberry Pi

You have now learned how to interact with different components (LED, RFID Scanner) using a Raspberry Pi. In this exercise we want you to combine them and come up with a simple authentication system. This exercise will also help you with your *ASEDelivery* project.

For the setup of this program, overwrite one of the RFID tokens we have provided you with a JSON object in the following format `{"id":"0000"}`. Then, create the following files under *ASE_Exercises/Week_1*:

- *user_ids.json*$\longrightarrow$`[{"id":"0000"}, {"id":"0001"},...]`
- *authenticator.py*$\longrightarrow$the authentication program

Also, instead of lighting a single color with your three-legged LED, you are now required to light two different colors (red, green). To do so, connect both positive legs to GPIO pins and control them as shown in Exercise 5. **Don't forget to use the resistors. Doesn't matter which resistor is connected to which leg/color.**

The expected behavior of the program:

- Expects an RFID token to be scanned.
- Reads the scanned token and prints its *text* field. (The text field contains a user id.)
- Calls a function called *authenticate(user_id)* to check whether the user id exists in the *credentials.json* file.
- If the user id exists, lights the green LED for 3 seconds, turns it off, and waits for a new RFID token to be scanned.
- If the user id does not exist, then lights the red LED for 3 seconds, turns it off, and waits for a new RFID token to be scanned.
- **If you are having trouble with reading JSON from a file, just store the user id that you have written into the token, inside a variable. Later, check whether the values you read from scanned tokens match the value of this variable.**