Project Domotica

10-4-2023

2223 1.3 BSc TI01 Domotica

Thom van der Veen
Semih Can Karakoç

Inhoudsopgave

Problem 1a – Vision Board 3
1.1 Introductie 3
Problem 1b – A3 Poster 4
Problem 2 – Databases 5
Opdracht 2 (6 uur) – MariaDB / MySQL: 5
Opdracht 3 (10 uur) – MariaDB in C 7
Problem 3 – GPIO 12
Opdracht 2 (3 uur) – GPIO 12
Opdracht 3 (2 uur) – GPIO in C 15
Opdracht 4 (3 uur) - Tactile switch 20
Opdracht 5 (6 uur) – Daemon 28
Opdracht 6 (5 uur) - mariadb 29
Problem 4 – Web Interface 33
Opdracht 2 (2 uur) – Website 33
Opdracht 3 (6 uur) – Sockets 35
Opdracht 4 (3 uur) - Web interface 38
Opdracht 5 (4 uur) – Security 50
Opdracht 6 (2 uur) – API 59
Problem 5 – Arduino 68
Opdracht 2 (1 uur) - Arduino aansluiten aan de Raspberry Pi 68
Opdracht 3 (4 uur) – Protocol 74
Opdracht 4 (2 uur) - Ontwerp code 77
Opdracht 5 (2 uur) – Requirements 78
Opdracht 6 (9 uur) - Test en implementatie 80
Het eindproduct: de slimme deurbel 86

Problem 1a – Vision Board

1.1 Introductie

Voor dit project hebben we ervoor gekozen om een “slimme deurbel” te maken. Dit product bevat een knop, een camera, een bewegingssensor en een klein schermpje. Daarnaast sluiten we ook een echte lamp aan door middel van een relais. Deze lamp zal geactiveerd worden voor 60 seconden, wanneer de actuator (PIR) afgaat. De lamp zal na een bepaalde tijd deactiveren wanneer er geen beweging meer wordt gedetecteerd. Met dit slimme principe besparen we energie, hebben we een mooie deurbel, plus we besparen ook nog eens geld en gooien we geen elektriciteit weg!

Voor dit project willen we er ook voor zorgen dat er aangebeld kan worden. Zodra er aangebeld wordt, zal de camera een foto maken. Deze foto zal daarna verstuurd naar een lokale database op de Raspberry Pi. Via onze web interface kunnen we dan de foto terug zien met de time stamp. Via deze site kunnen we ook de lamp handmatig aan en uit zetten als we dat willen. Daarnaast is het ook mogelijk om de status van de lamp terug te zien door middel van een database!

Zie de figuur hieronder voor de representatie van onze vision board.

Problem 1b – A3 Poster

Project Domotics

A smart and eco-friendly doorbell

Semih Can Karakoç and Thom van der Veen
Inholland University of Applied Sciences

Abstract

My name is Semih Can Karakoç. I live at 2 Oregano, Heerhugowaard, Noord-Holland, 1705RM.

My name is Thom van der Veen. I live at 255 Kennemerstraatweg, Alkmaar, Noord-Holland, 1814GK.

We are students at Inholland University of Applied Sciences. We are in Technische Informatica year one.

Introduction

This doorbell can take pictures of the person in front of the door. The doorbell also has a built-in motion sensor. When a person is at the door, this motion sensor will be triggered, causing a light to come on for a certain amount of time. The doorbell includes an interactive button that allows the person to ring the doorbell. The important data is sent via serial connection to a database and stored there. Camera images are sent to the database by photo via WiFi. This important information is represented on our doorbell web interface On the web interface the user is also allowed to interact with the lamp and see data.

List of components

- ESP32 microcontroller with camera, PIR sensor, OLED screen and button;
- Raspberry Pi;
- 5V Relay
- Level Converter
- ESP32 case;
- Wires.

Methodology

For this product we used an ESP32 MCU as a slave module with a camera, OLED screen, PIR sensor and a button. We also used a Raspberry Pi as the master module. On the OLED screen the person will see the house number. When the PIR sensor detects motion, it sends an 'X' to the Raspberry Pi via serial connection. The Raspberry Pi has a C daemon running in the background and when it receives an 'X' via the serial port, the C program will turn on a 5V relay with a lamp wired to it. The lamp will keep on for 60 seconds and turn off when there is no motion detected anymore. When a person presses the button, the OLED screen will say "Ding Dong" for a couple of seconds and change back to the house number. The ESP32 will also take a picture of the person at the door. The picture is sent via WiFi to the web interface. The owner of the doorbell can see the date and time of the picture and is able to keep the picture or delete it. The owner can also see the status of the lamp and toggle it on or off.

Process and results

To start this project, we researched for components and solutions. We eventually decided to use an ESP32 based board with a MCU, camera, PIR sensor, OLED screen and a button as the doorbell and to use a 5V relay to control the light. To control the ESP32 we wrote C++ code via Arduino IDE.

On the Raspberry Pi we used:

- MariaDB for the database which stores the data of the light;
- PHP and HTML code running on Nginx to display the gallery with pictures and handle the connection for the image transfer;
- a python script to control the 5V relay;
- a C daemon that communicates with the ESP32 via serial connection with wires.

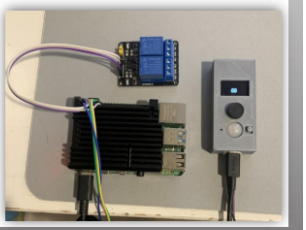
We first made the ESP32 camera capture and image transfer functionality with help from the ESP32CAM documentation examples. Then we made the serial connection between the Raspberry Pi and ESP32 to communicate PIR sensor state.
Lastly, we implemented the database and necessary connections in the PHP and C scripts.

Later during the project, we added the ability to see the state of the lamp and control it via the web interface.

At the end we have made a working doorbell connected to a Raspberry Pi

Final product

This is our final doorbell product, based on ESP32 and Raspberry Pi:



Tools and techniques

We used a lot of tools and software to create the final product.

We used a few physical tools;

- A 3D printer
- JST crimping tool
- Screwdriver

We used the following software: Windows, Linux, VNC Viewer, Arduino IDE, terminal, Geany, VSCode, Fritzing, draw.io, Powerpoint, Word, Google Chrome, nginx.

We did this all on our windows laptop.

Acknowledgements

We would like to thank the makers of the ESP32 CAM documentation and w3 schools for the datasheets and explanations.

Problem 2 – Databases

Opdracht 2 (6 uur) – MariaDB / MySQL:

***Voeg meer gebruikers toe aan de tabel:**

INSERT INTO gebruikers(gebruikersId, gebruikersNaam) VALUES(2, 'Semih'), (3, 'Thom'), (4, 'Elmer'), (5, 'Marina');

```
MariaDB [mijndb]> INSERT INTO gebruikers(gebruikersId, gebruikersNaam) VALUES(2, 'Semih'), (3, 'Thom'), (4, 'Elmer'), (5, 'Marina');
Query OK, 4 rows affected (0.004 sec)
Records: 4  Duplicates: 0  Warnings: 0

MariaDB [mijndb]> select * from gebruikers;
+--------------+----------------+
| gebruikersId | gebruikersNaam |
+--------------+----------------+
|            2 | Semih          |
|            3 | Thom           |
|            4 | Elmer          |
|            5 | Marina         |
+--------------+----------------+
4 rows in set (0.001 sec)
```

***Vraag de naam op van een gebruiker met een specifiek id:**

SELECT gebruikersNaam FROM gebruikers WHERE gebruikersId = 2;

```
MariaDB [mijndb]> SELECT gebruikersNaam FROM gebruikers WHERE gebruikersId = 2;
+----------------+
| gebruikersNaam |
+----------------+
| Semih          |
+----------------+
1 row in set (0.001 sec)
```

***Verwijder een gebruiker uit de tabel:**

DELETE FROM gebruikers WHERE gebruikersId = 4;

```
MariaDB [mijndb]> DELETE FROM gebruikers WHERE gebruikersId = 4;
Query OK, 1 row affected (0.018 sec)

MariaDB [mijndb]> select * from gebruikers;
+--------------+----------------+
| gebruikersId | gebruikersNaam |
+--------------+----------------+
|            2 | Semih          |
|            3 | Thom           |
|            5 | Marina         |
+--------------+----------------+
3 rows in set (0.001 sec)
```

***Pas de naam aan van een bestaande gebruiker:**
UPDATE gebruikers SET gebruikersNaam = 'Richard' WHERE gebruikersId = 3;

```
MariaDB [mijndb]> UPDATE gebruikers SET gebruikersNaam = 'Richard' WHERE gebruikersId = 3;
Query OK, 1 row affected (0.003 sec)
Rows matched: 1  Changed: 1  Warnings: 0

MariaDB [mijndb]> select * from gebruikers;
+--------------+----------------+
| gebruikersId | gebruikersNaam |
+--------------+----------------+
|            2 | Semih          |
|            3 | Richard        |
|            5 | Marina         |
+--------------+----------------+
3 rows in set (0.001 sec)
```

***Maak een nieuwe tabel aan en voeg data toe:**
CREATE TABLE inHolland (klasId int, klasNaam char(50));

INSERT INTO inHolland (klasId, klasNaam) VALUES (1, 'Technische Informatica'), (2, 'Informatica'), (3, 'Elektrotechniek');

```
MariaDB [mijndb]> CREATE TABLE inHolland (klasId int, klasNaam char(50));
LUES (1, 'Technische Informatica'), (2, 'Informatica'), (3, 'Elektrotechniek');
Query OK, 0 rows affected (0.028 sec)

MariaDB [mijndb]> INSERT INTO inHolland (klasId, klasNaam) VALUES (1, 'Technische Informatica'), (2, 'Informatica'), (3, 'Elektrotechniek');
Query OK, 3 rows affected (0.004 sec)
Records: 3  Duplicates: 0  Warnings: 0

MariaDB [mijndb]> select * from inHolland;
+--------+----------------------+
| klasId | klasNaam             |
+--------+----------------------+
|      1 | Technische Informatica |
|      2 | Informatica          |
|      3 | Elektrotechniek      |
+--------+----------------------+
3 rows in set (0.001 sec)
```

***Verwijder een hele tabel:**
DROP TABLE inHolland;

```
MariaDB [mijndb]> DROP TABLE inHolland;
Query OK, 0 rows affected (0.103 sec)

MariaDB [mijndb]> select * from inHolland;
ERROR 1146 (42S02): Table 'mijndb.inHolland' doesn't exist
```

***Bronnen:**
https://www.w3schools.com/sql/default.asp
https://www.tutorialspoint.com/sql/index.htm

Opdracht 3 (10 uur) – MariaDB in C

Code 1: Toon de gegevens uit een tabel op het scherm

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mariadb/mysql.h>

int rowCounter;

int main (int ac, char **ap) {
    MYSQL_RES *res;
    MYSQL_ROW row;
    MYSQL *connection = mysql_init(NULL);
    mysql_real_connect(connection, "localhost", "semih", "semih", "mijndb", 0,
NULL, 0);
    if (connection == NULL) {
        printf("Kan geen verbinding met de MariaDB server maken\n");
        exit(-1);
    }
    printf("\nZie hieronder een lijst van alle gegevens van de tabel
\"gebruikers\"\n");
    int error = mysql_query(connection, "SELECT * FROM gebruikers");
    if (error) printf("%s\n", mysql_error(connection));
        else {
            MYSQL_RES *result = mysql_use_result(connection);
            if (result) {
                MYSQL_ROW row;
                while ((row = mysql_fetch_row(result)) != NULL) {
                    rowCounter++;
                    printf("%s %s\n", row[0], row[1]);
                }
            mysql_free_result(result);
            }
        }
    mysql_close(connection);
    return 0;
}
```

Code 2: Voeg meer gebruikers toe aan de tabel

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mariadb/mysql.h>

int main (int ac, char **ap) {
    MYSQL_RES *res;
    MYSQL_ROW row;
    MYSQL *connection = mysql_init(NULL);
    mysql_real_connect(connection, "localhost", "semih", "semih", "mijndb", 0,
NULL, 0);
    if (connection == NULL) {
        printf("Kan geen verbinding met de MariaDB server maken\n");
        exit(-1);
    }
    int error = mysql_query(connection,
        "INSERT INTO gebruikers(gebruikersId, gebruikersNaam) values (123,
\"Elmer\")");
    if(error) printf("%s\n", mysql_error(connection));
    mysql_close(connection);
    return 0;
}
```

Code 3: Maak een nieuwe tabel aan en voeg data toe

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mariadb/mysql.h>

int main (int ac, char **ap) {
    MYSQL_RES *res;
    MYSQL_ROW row;
    MYSQL *connection = mysql_init(NULL);
    mysql_real_connect(connection, "localhost", "semih", "semih", "mijndb", 0,
NULL, 0);
    if (connection == NULL) {
        printf("Kan geen verbinding met de MariaDB server maken\n");
        exit(-1);
    }
    int error = mysql_query(connection,
        "CREATE TABLE newTable(id INT AUTO_INCREMENT PRIMARY KEY, tableName
VARCHAR(255))");
    if(error) printf("%s\n", mysql_error(connection));
    mysql_close(connection);
    return 0;
}
```

Code 4: Verwijder een gebruiker uit de tabel

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mariadb/mysql.h>

int main (int ac, char **ap) {
    MYSQL_RES *res;
    MYSQL_ROW row;
    MYSQL *connection = mysql_init(NULL);
    mysql_real_connect(connection, "localhost", "semih", "semih", "mijndb", 0,
NULL, 0);
    if (connection == NULL) {
        printf("Kan geen verbinding met de MariaDB server maken\n");
        exit(-1);
    }
    int error = mysql_query(connection,
        "DELETE FROM gebruikers WHERE gebruikersId = 123");
    if(error) printf("%s\n", mysql_error(connection));
    mysql_close(connection);
    return 0;
}
```

Code 5: Pas de naam aan van een bestaande gebruiker

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mariadb/mysql.h>

int main (int ac, char **ap) {
    MYSQL_RES *res;
    MYSQL_ROW row;
    MYSQL *connection = mysql_init(NULL);
    mysql_real_connect(connection, "localhost", "semih", "semih", "mijndb", 0,
NULL, 0);
    if (connection == NULL) {
        printf("Kan geen verbinding met de MariaDB server maken\n");
        exit(-1);
    }
    int error = mysql_query(connection,
        "UPDATE gebruikers SET gebruikersNaam = 'Richard' WHERE gebruikersId =
1");
    if(error) printf("%s\n", mysql_error(connection));
    mysql_close(connection);
    return 0;
}
```

Problem 3 – GPIO

Opdracht 2 (3 uur) – GPIO

Waarom wordt de negatieve kant (korte pootje) van de led aangesloten op GPIO 21 en niet de positieve kant?

Omdat de led stroom krijgt van de 3,3V voeding pin, de GPIO pin is dan de negatieve kant.

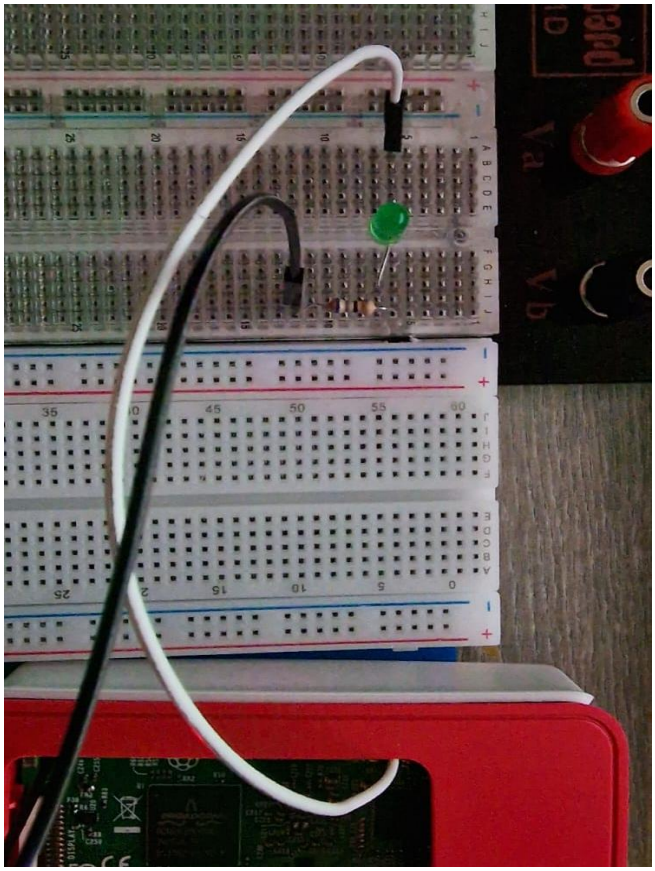
Hoe kan een tactile switch worden aangesloten?

Door die tussen de 3.3V en de LED te zetten, daarmee zorg je ervoor dat er alleen stroom door de led gaat als je de knop in drukt.

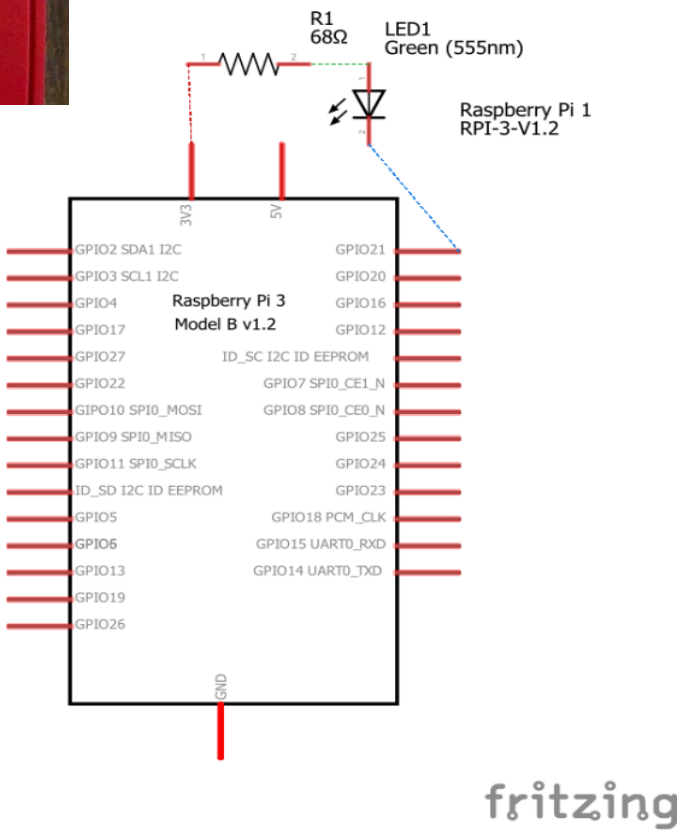
Code 1: bash script om de led te laten knipperen

```
#!/bin/bash

# Set GPIO pin 21 as output
echo "out" > /sys/class/gpio/gpio21/direction

# Loop forever
while true; do
  # Set GPIO pin 21 to 0
  echo "0" > /sys/class/gpio/gpio21/value
  # Wait for 0.5 seconds
  sleep 0.5
  # Set GPIO pin 21 to 1
  echo "1" > /sys/class/gpio/gpio21/value
  # Wait for 0.5 seconds
  sleep 0.5
done
```

LED schakeling:

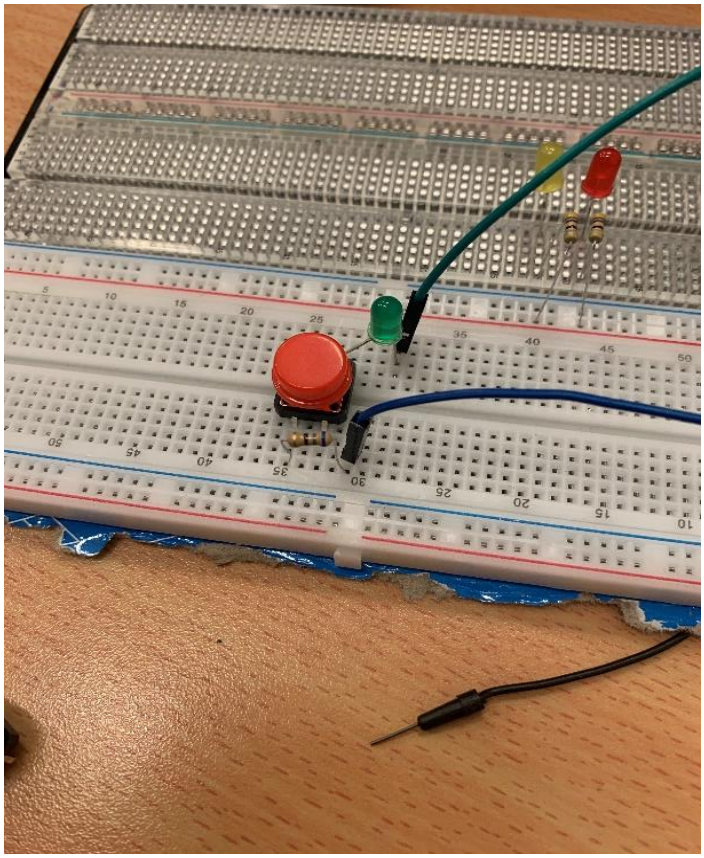


Figuur 1 - foto van de schakeling



Figuur 2 - schema van de schakeling

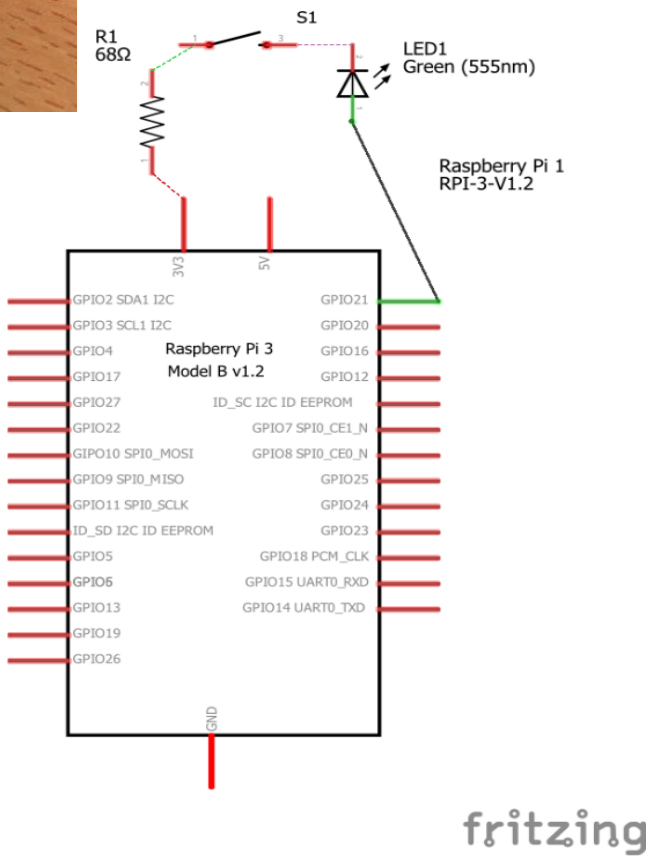
LED schakeling met knop:



Figuur 3 - foto van de schakeling

Bronnen:
https://roboticsbackend.com/arduino-led-complete-tutorial/

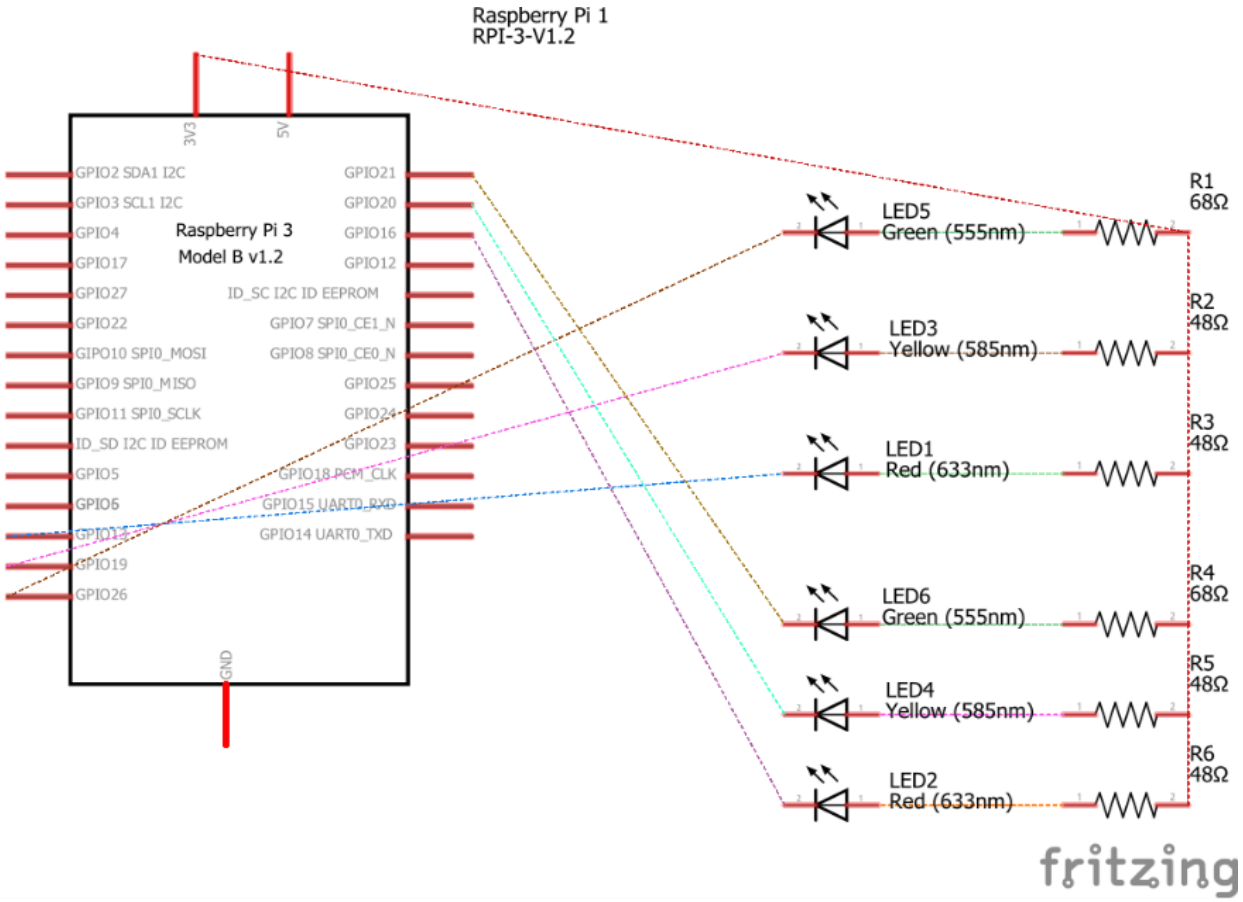
https://docs.arduino.cc/built-in-examples/digital/Button



Figuur 4 - schema van de schakeling

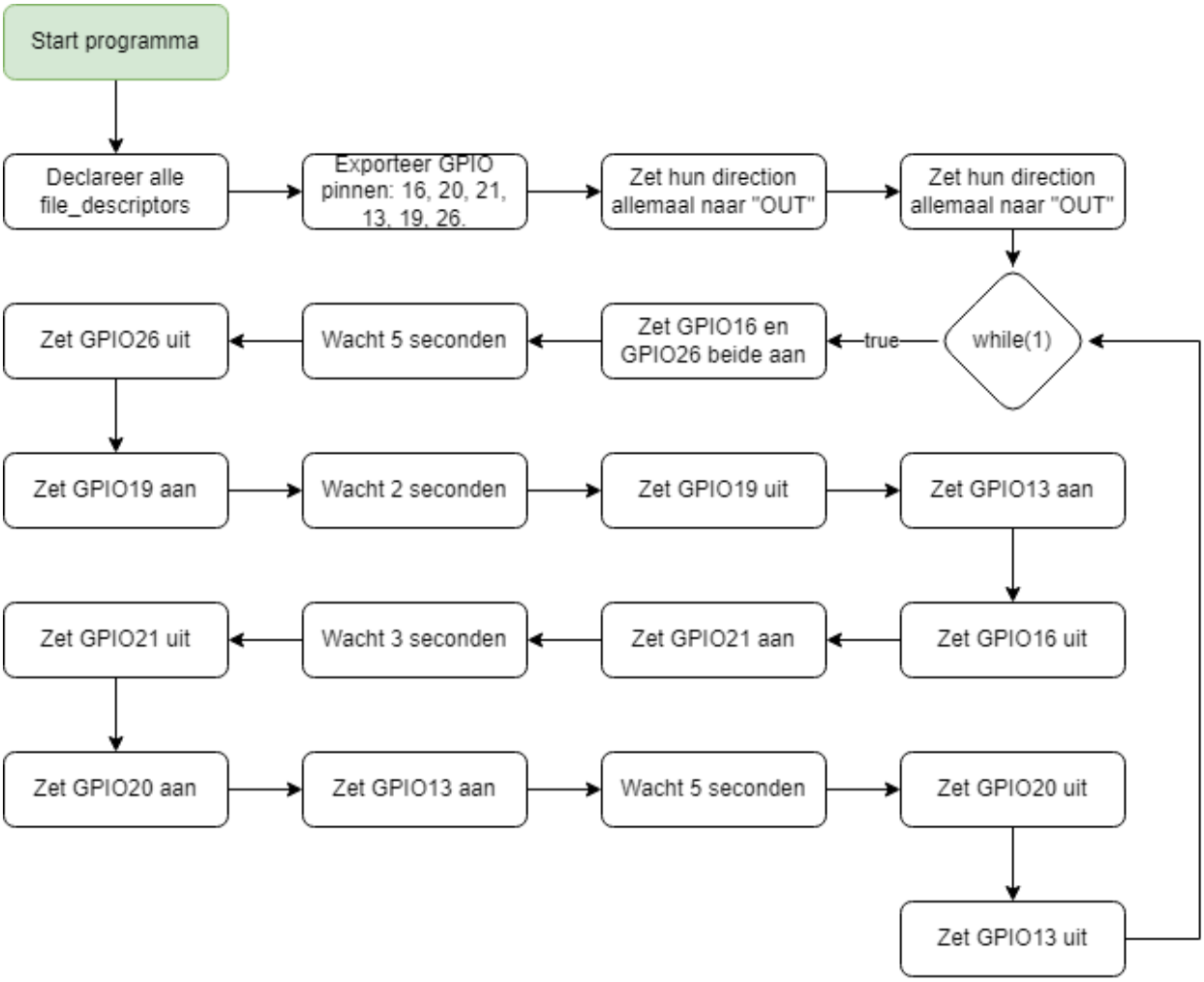
Opdracht 3 (2 uur) – GPIO in C

Stoplichten schakeling:



Figuur 5 - Schema van de schakeling

PSD van de code:



Code:

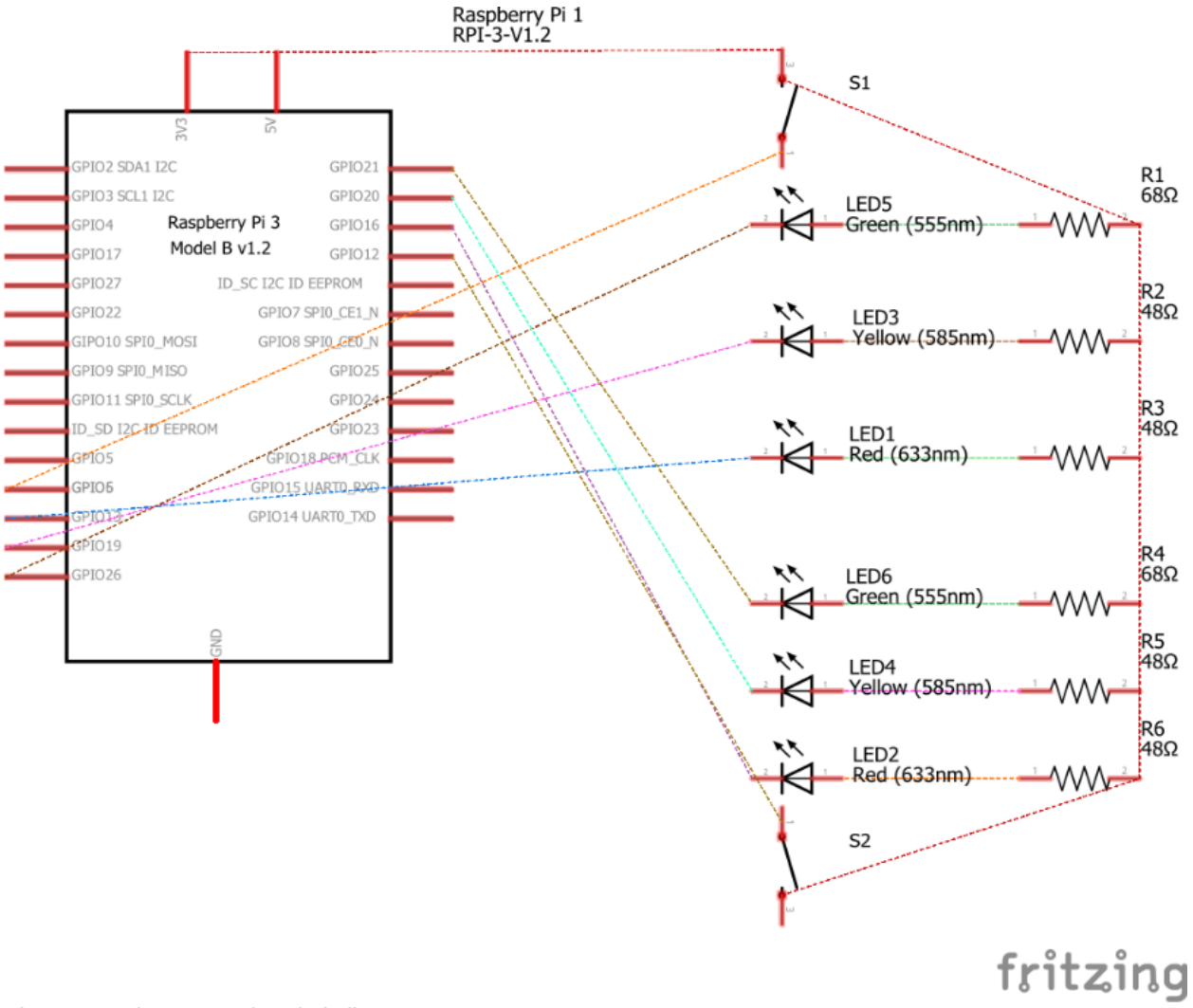
```c
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>

int main(int ac, char **ap) {
int file_descriptor1, file_descriptor2, file_descriptor3, file_descriptor4,
file_descriptor5, file_descriptor6;

file_descriptor1 = open("/sys/class/gpio/export", O_WRONLY);
write(file_descriptor1, "16", 2);
close(file_descriptor1);
file_descriptor1 = open("/sys/class/gpio/gpio16/direction", O_WRONLY);
write(file_descriptor1, "out", 3);
close(file_descriptor1);

file_descriptor2 = open("/sys/class/gpio/export", O_WRONLY);
write(file_descriptor2, "20", 2);
close(file_descriptor2);
file_descriptor2 = open("/sys/class/gpio/gpio20/direction", O_WRONLY);
write(file_descriptor2, "out", 3);
close(file_descriptor2);

file_descriptor3 = open("/sys/class/gpio/export", O_WRONLY);
write(file_descriptor3, "21", 2);
close(file_descriptor3);
file_descriptor3 = open("/sys/class/gpio/gpio21/direction", O_WRONLY);
write(file_descriptor3, "out", 3);
close(file_descriptor3);

file_descriptor4 = open("/sys/class/gpio/export", O_WRONLY);
write(file_descriptor4, "13", 2);
close(file_descriptor4);
file_descriptor4 = open("/sys/class/gpio/gpio13/direction", O_WRONLY);
write(file_descriptor4, "out", 3);
close(file_descriptor4);

file_descriptor5 = open("/sys/class/gpio/export", O_WRONLY);
write(file_descriptor5, "19", 2);
close(file_descriptor5);
file_descriptor5 = open("/sys/class/gpio/gpio19/direction", O_WRONLY);
write(file_descriptor5, "out", 3);
close(file_descriptor5);
```

```
file_descriptor6 = open("/sys/class/gpio/export", O_WRONLY);
write(file_descriptor6, "26", 2);
close(file_descriptor6);
file_descriptor6 = open("/sys/class/gpio/gpio26/direction", O_WRONLY);
write(file_descriptor6, "out", 3);
close(file_descriptor6);

while (1) {
    file_descriptor1 = open("/sys/class/gpio/gpio16/value", O_WRONLY); //16 rood2
    write(file_descriptor1, "0", 1);
    close(file_descriptor1);
    file_descriptor6 = open("/sys/class/gpio/gpio26/value", O_WRONLY); //26
green1
    write(file_descriptor6, "0", 1);
    close(file_descriptor6);
    sleep(5);

    file_descriptor6 = open("/sys/class/gpio/gpio26/value", O_WRONLY);
    write(file_descriptor6, "1", 1);
    close(file_descriptor6);



    file_descriptor5 = open("/sys/class/gpio/gpio19/value", O_WRONLY); //19 geel1
    write(file_descriptor5, "0", 1);
    close(file_descriptor5);
    sleep(2);
    file_descriptor5 = open("/sys/class/gpio/gpio19/value", O_WRONLY);
    write(file_descriptor5, "1", 1);
    close(file_descriptor5);
    file_descriptor4 = open("/sys/class/gpio/gpio13/value", O_WRONLY); //13 rood1
    write(file_descriptor4, "0", 1);
    close(file_descriptor4);
    file_descriptor1 = open("/sys/class/gpio/gpio16/value", O_WRONLY);
    write(file_descriptor1, "1", 1);
    close(file_descriptor1);
    file_descriptor3 = open("/sys/class/gpio/gpio21/value", O_WRONLY); //21
green2
    write(file_descriptor3, "0", 1);
    close(file_descriptor3);
    sleep(3);
    file_descriptor3 = open("/sys/class/gpio/gpio21/value", O_WRONLY);
    write(file_descriptor3, "1", 1);
    close(file_descriptor3);
```

```
    file_descriptor2 = open("/sys/class/gpio/gpio20/value", O_WRONLY);
    write(file_descriptor2, "0", 1);
    close(file_descriptor2);
    file_descriptor4 = open("/sys/class/gpio/gpio13/value", O_WRONLY); //13 rood1
    write(file_descriptor4, "0", 1);
    close(file_descriptor4);
    sleep(5);
    file_descriptor2 = open("/sys/class/gpio/gpio20/value", O_WRONLY);
    write(file_descriptor2, "1", 1);
    close(file_descriptor2);
    file_descriptor4 = open("/sys/class/gpio/gpio13/value", O_WRONLY);
    write(file_descriptor4, "1", 1);
    close(file_descriptor4);
    }
 return 0;
}
```

Opdracht 4 (3 uur) - Tactile switch

Stoplichten schakeling met knoppen:



Figuur 6 - Schema van de schakeling

PSD van de code:

C Code:

```c
#include <string.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>

int file_descriptor1, file_descriptor2, file_descriptor3, file_descriptor4,
file_descriptor5, file_descriptor6, file_descriptorB1, file_descriptorB2;
char value1[1024];
char value2[1024];
int n, n1;

void set_gpio_state(const char *gpio_pin_path, const char *state) {
    int file_descriptor = open(gpio_pin_path, O_WRONLY);
    write(file_descriptor, state, strlen(state));
    close(file_descriptor);
}

void rood2Aan() {
    set_gpio_state("/sys/class/gpio/gpio16/value", "0");
}
void rood2Uit() {
    set_gpio_state("/sys/class/gpio/gpio16/value", "1");
}
void geel2Aan() {
    set_gpio_state("/sys/class/gpio/gpio20/value", "0");
}
void geel2Uit() {
    set_gpio_state("/sys/class/gpio/gpio20/value", "1");
}
void groen2Aan() {
    set_gpio_state("/sys/class/gpio/gpio21/value", "0");
}
void groen2Uit() {
    set_gpio_state("/sys/class/gpio/gpio21/value", "1");
}
void rood1Aan() {
    set_gpio_state("/sys/class/gpio/gpio13/value", "0");
}
void rood1Uit() {
    set_gpio_state("/sys/class/gpio/gpio13/value", "1");
}
void geel1Aan() {
```

```
    set_gpio_state("/sys/class/gpio/gpio19/value", "0");
}
void geel1Uit() {
    set_gpio_state("/sys/class/gpio/gpio19/value", "1");
}
void groen1Aan() {
    set_gpio_state("/sys/class/gpio/gpio26/value", "0");
}
void groen1Uit() {
    set_gpio_state("/sys/class/gpio/gpio26/value", "1");
}

int knopTest2() {
        file_descriptorB2 = open("/sys/class/gpio/gpio3/value", O_RDONLY);
        read(file_descriptorB2, value2, sizeof(value2));
        printf("\nknop 2: %s\n", value2);
        n = strcmp(value2, "1");
        close(file_descriptorB2);
}

int knopTest1() {
        file_descriptorB1 = open("/sys/class/gpio/gpio2/value", O_RDONLY);
        read(file_descriptorB1, value1, sizeof(value1));
        printf("\nknop 1: %s\n", value1);
        n1 = strcmp(value1, "1");
        close(file_descriptorB1);
}

void allesUit() {
    geel1Uit();
    geel2Uit();
    rood1Uit();
    rood2Uit();
    groen1Uit();
    groen2Uit();
}
int main(int ac, char **ap) {
    file_descriptor1 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor1, "16", 2);
    close(file_descriptor1);
    file_descriptor1 = open("/sys/class/gpio/gpio16/direction", O_WRONLY);
    write(file_descriptor1, "out", 3);
    close(file_descriptor1);

    file_descriptor2 = open("/sys/class/gpio/export", O_WRONLY);
```

```
    write(file_descriptor2, "20", 2);
    close(file_descriptor2);
    file_descriptor2 = open("/sys/class/gpio/gpio20/direction", O_WRONLY);
    write(file_descriptor2, "out", 3);
    close(file_descriptor2);

    file_descriptor3 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor3, "21", 2);
    close(file_descriptor3);
    file_descriptor3 = open("/sys/class/gpio/gpio21/direction", O_WRONLY);
    write(file_descriptor3, "out", 3);
    close(file_descriptor3);
    //1
    file_descriptor4 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor4, "13", 2);
    close(file_descriptor4);
    file_descriptor4 = open("/sys/class/gpio/gpio13/direction", O_WRONLY);
    write(file_descriptor4, "out", 3);
    close(file_descriptor4);

    file_descriptor5 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor5, "19", 2);
    close(file_descriptor5);
    file_descriptor5 = open("/sys/class/gpio/gpio19/direction", O_WRONLY);
    write(file_descriptor5, "out", 3);
    close(file_descriptor5);

    file_descriptor6 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor6, "26", 2);
    close(file_descriptor6);
    file_descriptor6 = open("/sys/class/gpio/gpio26/direction", O_WRONLY);
    write(file_descriptor6, "out", 3);
    close(file_descriptor6);

    file_descriptorB1 = open("/sys/class/gpio/export", O_WRONLY); //knop
stoplicht 1
    write(file_descriptorB1, "2", 2);
    close(file_descriptorB1);
    file_descriptorB1 = open("/sys/class/gpio/gpio2/direction", O_WRONLY);
    write(file_descriptorB1, "out", 3);
    close(file_descriptorB1);

    file_descriptorB2 = open("/sys/class/gpio/export", O_WRONLY); //knop
stoplicht 2
    write(file_descriptorB2, "3", 2);
```

```
    close(file_descriptorB2);
    file_descriptorB2 = open("/sys/class/gpio/gpio3/direction", O_WRONLY);
    write(file_descriptorB2, "out", 3);
    close(file_descriptorB2);
    // Zet alles eerst uit
    allesUit();
    while (1) {
        knopTest2();
        if (10 == n) {
            allesUit();
            printf("knop 2 is ingedrukt");
            goto knop2;
        }

        knopTest1();
        if (10 == n1) {
            allesUit();
            printf("knop 1 is ingedrukt");
            goto knop1;
        }
        knop2:
        rood1Aan();
        groen2Aan();
        for (int i = 0; i < 5; i++) {

            knopTest2();
            if (10 == n) {
                allesUit();
                printf("knop 2 is ingedrukt");
                goto knop2;
            }

            knopTest1();
            if (10 == n1) {
                allesUit();
                printf("knop 1 is ingedrukt");
                goto knop1;
            }
            sleep(1);
        }
        groen2Uit();
        geel2Aan();
        for (int i = 0; i < 2; i++) {

            knopTest2();
```

```
            if (10 == n) {
                allesUit();
                printf("knop 2 is ingedrukt");
                goto knop2;
            }
            knopTest1();
            if (10 == n1) {
                allesUit();
                printf("knop 1 is ingedrukt");
                goto knop1;
            }
            sleep(1);
        }
        rood1Uit();
        geel2Uit();
        knop1:
        rood2Aan();
        groen1Aan();
        for (int i = 0; i < 5; i++) {
            knopTest2();
            if (10 == n) {
                allesUit();
                printf("knop 2 is ingedrukt");
                goto knop2;
            }
            knopTest1();
            if (10 == n1) {
                allesUit();
                printf("knop 1 is ingedrukt");
                goto knop1;
            }
            sleep(1);
        }
        groen1Uit();
        geel1Aan();
        for (int i = 0; i < 2; i++) {
            knopTest2();
            if (10 == n) {
                allesUit();
                printf("knop 2 is ingedrukt");
                goto knop2;
            }
            knopTest1();
            if (10 == n1) {
                allesUit();
```

```
                printf("knop 1 is ingedrukt");
                goto knop1;
            }
            sleep(1);
        }
        rood2Uit();
        geel1Uit();
    }
    return 0;
}
```

Bronvermelding:
https://docs.arduino.cc/built-in-examples/digital/Button
https://www.allaboutcircuits.com/technical-articles/using-interrupts-on-arduino/

Opdracht 5 (6 uur) – Daemon

De daemontest.service file:

```
[Unit]
Description=Stoplicht-programmatje

[Service]
Type=simple
WorkingDirectory=/home/semihpi/Codon
ExecStart=/home/semihpi/Codon/run
Restart=always

[Install]
WantedBy=multi-user.target
```

De stoplichtInstall.sh file:

```
#!/bin/bash

# Start de service
sudo systemctl start daemontest

# Schakel de service in
sudo systemctl enable daemontest
```

De stoplichtUninstall.sh file:

```
#!/bin/bash

# Stop de service
sudo systemctl stop daemontest

# Schakel de service uit
sudo systemctl disable daemontest
```

Opdracht 6 (5 uur) - MariaDB

SQL query om de tabel aan te maken:

```
CREATE TABLE lichten(stoplichtID INT, stoplichtKleur VARCHAR(255),
stoplichtStatus BOOLEAN);
```

SQL C Code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <mariadb/mysql.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <fcntl.h>

int file_descriptor1, file_descriptor2, file_descriptor3, file_descriptor4,
file_descriptor5, file_descriptor6;
char value1[1024], value2[1024], value3[1024], value4[1024], value5[1024],
value6[1024];
int n1, n2, n3, n4, n5, n6;
int main (int ac, char **ap) {
    MYSQL_RES *result;
    MYSQL_ROW row;
    MYSQL *connection = mysql_init(NULL);

    mysql_real_connect(connection, "localhost", "semih", "semih", "stoplicht", 0,
NULL, 0);
    if (connection == NULL) {
        printf("Kan geen verbinding met de MariaDB server maken\n");
        exit(-1);
    } else {
        printf("\n~Connected to SQL server!\n");
    }

    while (1) {
        file_descriptor1 = open("/sys/class/gpio/gpio16/value", O_RDONLY);
        read(file_descriptor1, value1, sizeof(value1));
        n1 = strcmp(value1, "1");
        if (n1 == 10) {
            printf("\nRood2 = FALSE");
            int injectQuery = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '0' WHERE stoplichtID = 2
AND stoplichtKleur = 'rood'");
                if(injectQuery) printf("%s\n", mysql_error(connection));
        } else {
```

```
            printf("\nRood2 = TRUE");
            int injectQuery = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '1' WHERE stoplichtID = 2
AND stoplichtKleur = 'rood'");
                if(injectQuery) printf("%s\n", mysql_error(connection));
        }
        close(file_descriptor1);

        file_descriptor2 = open("/sys/class/gpio/gpio20/value", O_RDONLY);
        read(file_descriptor2, value2, sizeof(value2));
        n2 = strcmp(value2, "1");
        if (n2 == 10) {
            printf("\nGeel2 = FALSE");
            int injectQuery2 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '0' WHERE stoplichtID = 2
AND stoplichtKleur = 'geel'");
        } else {
            printf("\nGeel2 = TRUE");
            int injectQuery2 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '1' WHERE stoplichtID = 2
AND stoplichtKleur = 'geel'");
        }
        close(file_descriptor2);

        file_descriptor3 = open("/sys/class/gpio/gpio21/value", O_RDONLY);
        read(file_descriptor3, value3, sizeof(value3));
        n3 = strcmp(value3, "1");
        if (n3 == 10) {
            printf("\nGroen2 = FALSE");
            int injectQuery3 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '0' WHERE stoplichtID = 2
AND stoplichtKleur = 'groen'");
        } else {
            printf("\nGroen2 = TRUE");
            int injectQuery3 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '1' WHERE stoplichtID = 2
AND stoplichtKleur = 'groen'");
        }
        close(file_descriptor3);

        file_descriptor4 = open("/sys/class/gpio/gpio13/value", O_RDONLY);
        read(file_descriptor4, value4, sizeof(value4));
        n4 = strcmp(value4, "1");
        if (n4 == 10) {
            printf("\nRood1 = FALSE");
```

```
            int injectQuery4 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '0' WHERE stoplichtID = 1
AND stoplichtKleur = 'rood'");
        } else {
            printf("\nRood1 = TRUE");
            int injectQuery4 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '1' WHERE stoplichtID = 1
AND stoplichtKleur = 'rood'");
        }
        close(file_descriptor4);

        file_descriptor5 = open("/sys/class/gpio/gpio19/value", O_RDONLY);
        read(file_descriptor5, value5, sizeof(value5));
        n5 = strcmp(value5, "1");
        if (n5 == 10) {
            printf("\nGeel1 = FALSE");
            int injectQuery5 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '0' WHERE stoplichtID = 1
AND stoplichtKleur = 'geel'");
        } else {
            printf("\nGeel1 = TRUE");
            int injectQuery5 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '1' WHERE stoplichtID = 1
AND stoplichtKleur = 'geel'");
        }
        close(file_descriptor5);

        file_descriptor6 = open("/sys/class/gpio/gpio26/value", O_RDONLY);
        read(file_descriptor6, value6, sizeof(value6));
        n6 = strcmp(value6, "1");
        if (n6 == 10) {
            printf("\nGroen1 = FALSE");
            int injectQuery6 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '0' WHERE stoplichtID = 1
AND stoplichtKleur = 'groen'");
        } else {
            printf("\nGroen1 = TRUE");
            int injectQuery6 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '1' WHERE stoplichtID = 1
AND stoplichtKleur = 'groen'");
        }
        close(file_descriptor6);
        printf("\n");
        sleep(30); // Wait 30 seconden (moet van de opdracht)
    }
```

```
    mysql_close(connection);
    return 0;
}
```

DB manipulatiescript (voorbeeld: 1/6):

```
file_descriptor6 = open("/sys/class/gpio/gpio26/value", O_RDONLY);
        read(file_descriptor6, value6, sizeof(value6));
        n6 = strcmp(value6, "1");
        if (n6 == 10) {
            printf("\nGroen1 = FALSE");
            int injectQuery6 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '0' WHERE stoplichtID = 1
AND stoplichtKleur = 'groen'");
        } else {
            printf("\nGroen1 = TRUE");
            int injectQuery6 = mysql_query(connection,
                "UPDATE lichten SET stoplichtStatus = '1' WHERE stoplichtID = 1
AND stoplichtKleur = 'groen'");
        }
        close(file_descriptor6);
```

De Daemonscript:

```
[Unit]
Description=StoplichtDB programmatje

[Service]
Type=simple
WorkingDirectory=/home/semihpi/Codon
ExecStart=/home/semihpi/Codon/sql
Restart=always

[Install]
WantedBy=multi-user.target
```

Problem 4 – Web Interface

Opdracht 2 (2 uur) – Website

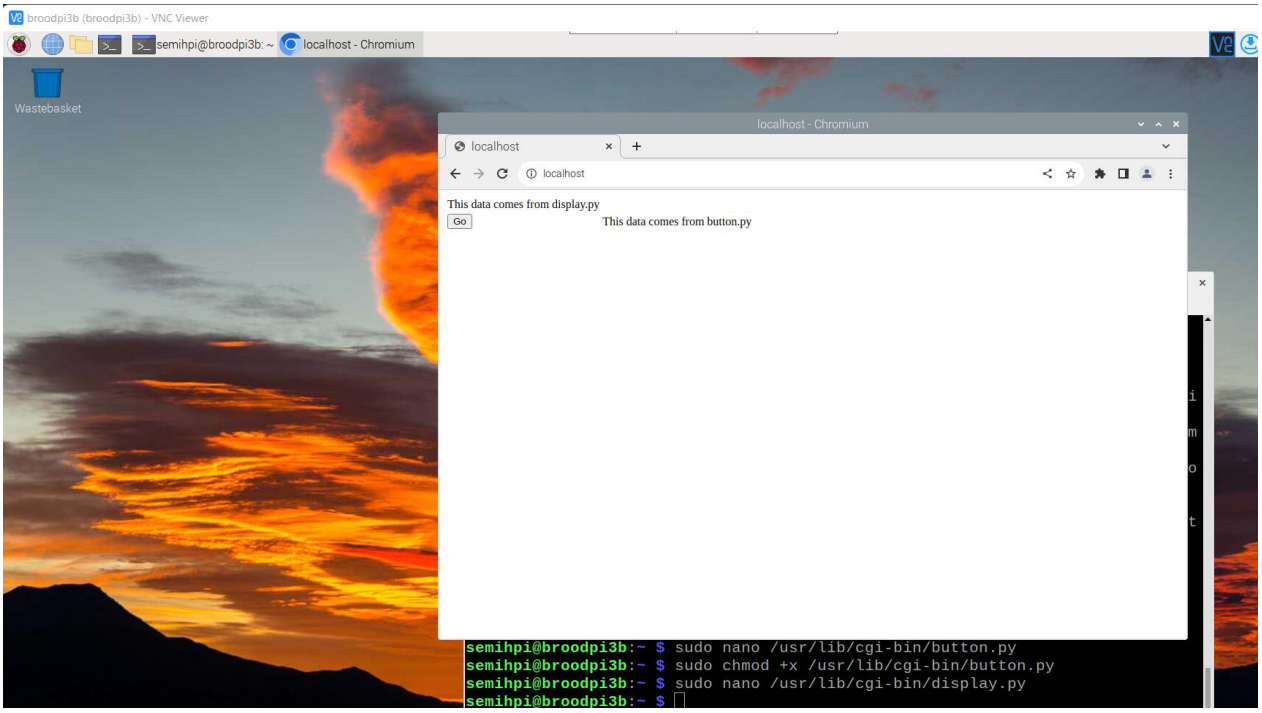


Figure 1: Proof that it works!

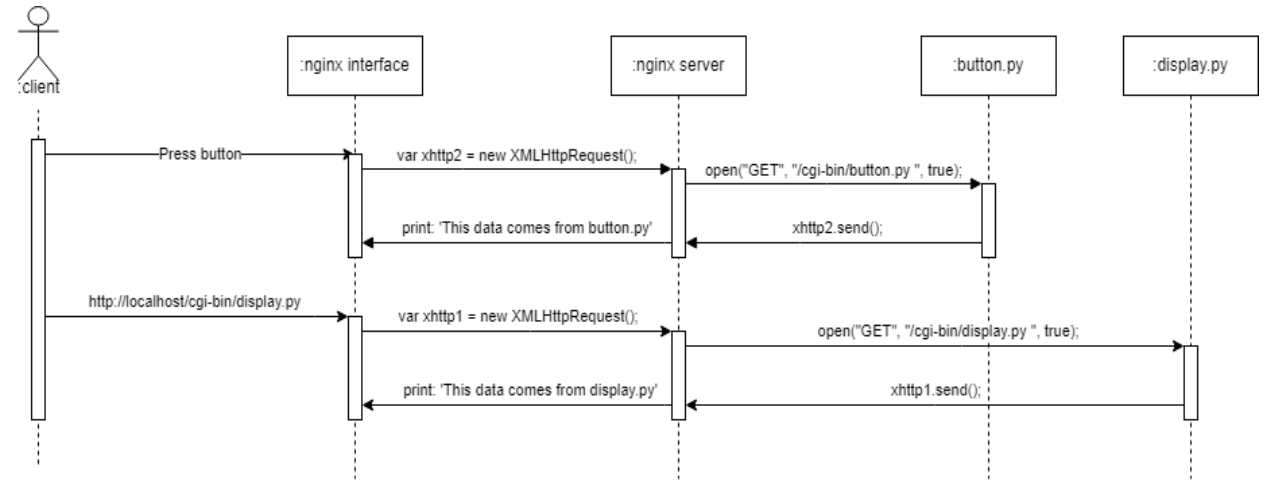


Figure 2:Dynamic Interaction Diagram

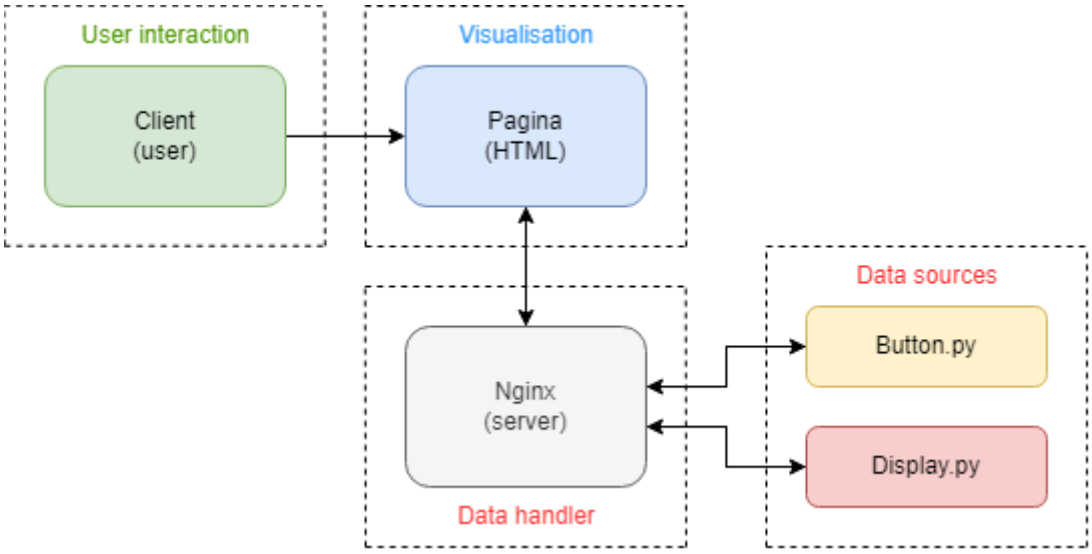


Figure 3: Static Architecture Diagram

Bronvermeldingen:

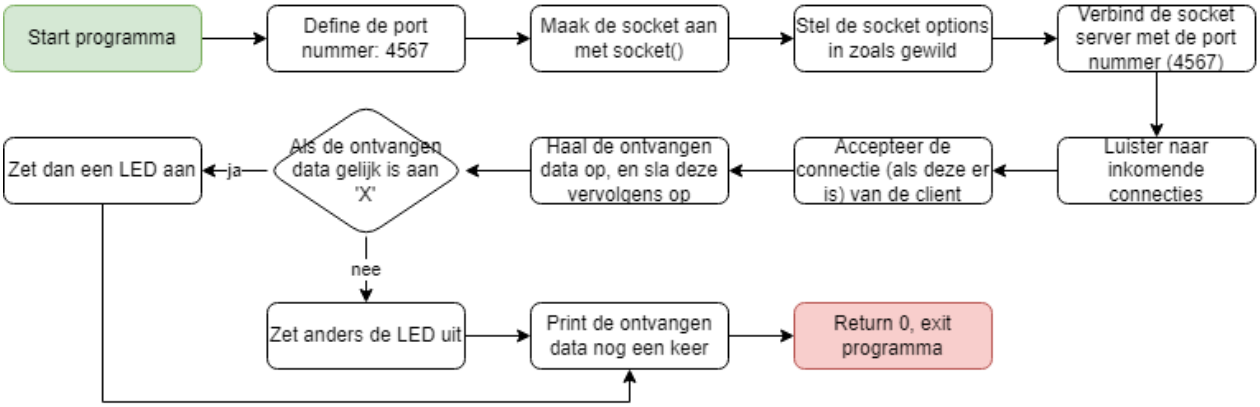
1. https://learn.microsoft.com/nl-nl/azure/architecture/reference-architectures/data/enterprise-bi-adf
2. https://www.cs.fsu.edu/~myers/cop3331/notes/dynamicmodel.html

Opdracht 3 (6 uur) – Sockets

Python code:

```python
import socket

HOST = 'broodpi3b'  # or the IP address of the server
PORT = 4567 # Port number

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT)) # Make connection with C-server
s.send('X')  # Send char 'X' to server
s.close()   # Close connection
```

PSD van de C code:



C code:

```c
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>

#define PORT 4567   // Port number

int main()
{
    int server_fd, new_socket, file_descriptor, n;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    const char *hello = "Hello from server";
```

```
    // Create the socket
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Some socket options (not needed)
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                   &opt, sizeof(opt)))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }

    // Bind the created socket with port number
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    if (bind(server_fd, (struct sockaddr *)&address,
             sizeof(address)) < 0)
    {
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    // Listen for connections
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }

    // Accept the connection
    if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                             (socklen_t *)&addrlen)) < 0)
    {
        perror("accept");
        exit(EXIT_FAILURE);
    }

    // Receive the data from pythonscript
    int valread = read(new_socket, buffer, 1024);
```

```
    n = strcmp(buffer, "1");
    if (n == 39)
    {
        file_descriptor = open("/sys/class/gpio/export", O_WRONLY);
        write(file_descriptor, "2", 2);
        close(file_descriptor);
        file_descriptor = open("/sys/class/gpio/gpio2/direction", O_WRONLY);
        write(file_descriptor, "out", 3);
        close(file_descriptor);

        file_descriptor = open("/sys/class/gpio/gpio2/value", O_WRONLY); // 26
green1
        write(file_descriptor, "0", 1);
        close(file_descriptor);
        printf("LED staat aan!");
    }
    else
    {
        file_descriptor = open("/sys/class/gpio/export", O_WRONLY);
        write(file_descriptor, "2", 2);
        close(file_descriptor);
        file_descriptor = open("/sys/class/gpio/gpio2/direction", O_WRONLY);
        write(file_descriptor, "out", 3);
        close(file_descriptor);

        file_descriptor = open("/sys/class/gpio/gpio2/value", O_WRONLY); // 26 g>
        write(file_descriptor, "1", 1);
        close(file_descriptor);
        printf("LED staat uit!");

        printf("\nwaarde is n = %d", n);
    }
    printf("Received %d bytes: %s\n", valread, buffer);
    return 0;
}
```

Bronvermeldingen:

1. http://www.linuxhowtos.org/C_C++/socket.htm
2. https://realpython.com/python-sockets/

Opdracht 4 (3 uur) - Web interface

HTML code:

```
<html>

<head>
    <style>
        .button1-green {
            background-color: green;
        }

        .button2-green {
            background-color: green;
        }

        .button1-red {
            background-color: red;
        }

        .button2-red {
            background-color: red;
        }

        .button1-orange {
            background-color: orange;
        }

        .button2-orange {
            background-color: orange;
        }
    </style>
    <script>
        var buttonOneActivated = false
        var buttonTwoActivated = false

        function dinges() {
            fetch('value')
                .then(response => response.text())
                .then(data => {
                    console.log(data); // Print the data read from the file
                })
                .catch(error => {
                    console.error(error);
                });
        }
        setInterval(function () {
```

```
            fetch('value')
                .then(response => response.text())
                .then(data => {
                    if (data.trim() === '1') {
                        var button1 = document.getElementById("button1");
                        button1.classList.remove("button1-red");
                        button1.classList.add("button1-green");
                        var button2 = document.getElementById("button2");
                        button2.classList.remove("button2-green");
                        button2.classList.add("button2-red");

                        setTimeout(function () {
                            button1.classList.remove("button1-green");
                            button1.classList.add("button1-orange");
                            setTimeout(function () {
                                button1.classList.remove("button1-orange");
                                button2.classList.remove("button2-red");
                            }, 2000);
                        }, 5000);
                    } else {
                        fetch('value2')
                            .then(response => response.text())
                            .then(data => {
                                if (data.trim() === '1') {
                                    var button1 =
document.getElementById("button1");
                                    button1.classList.remove("button1-green");
                                    button1.classList.add("button1-red");
                                    var button2 =
document.getElementById("button2");
                                    button2.classList.remove("button2-red");
                                    button2.classList.add("button2-green");

                                    setTimeout(function () {
                                        button2.classList.remove("button2-
green");

                                        button2.classList.add("button2-orange");
                                        setTimeout(function () {
                                            button2.classList.remove("button2-
orange");

                                            button1.classList.remove("button1-
red");

                                        }, 2000);
                                    }, 5000);
                                }
```

```
                        })
                        .catch(error => {
                            console.error(error);
                        });
                }
            })
            .catch(error => {
                console.error(error);
            });
    }, 1);

    function doButton1() {
        if (buttonOneActivated) {
            return
        }
        buttonOneActivated = true
        var xhttp = new XMLHttpRequest();
        xhttp.onreadystatechange = function () {
            if (this.readyState == 4 && this.status == 200) {
                console.log(this.responseText);
            }
        };
        xhttp.open("GET", "/cgi-bin/pyknop1.py", true);
        xhttp.send();

        var button1 = document.getElementById("button1");
        button1.classList.remove("button1-red");
        button1.classList.add("button1-green");
        var button2 = document.getElementById("button2");
        button2.classList.remove("button2-green");
        button2.classList.add("button2-red");

        setTimeout(function () {
            button1.classList.remove("button1-green");
            button1.classList.add("button1-orange");
            setTimeout(function () {
                button1.classList.remove("button1-orange");
                button2.classList.remove("button2-red");
            }, 2000);
        }, 5000);
    }

    function doButton2() {
        if (buttonTwoActivated) {
            return
```

```
            }
            buttonTwoActivated = true
            var xhttp1 = new XMLHttpRequest();
            xhttp1.onreadystatechange = function () {
                if (this.readyState == 4 && this.status == 200) {
                    console.log(this.responseText);
                }
            };
            xhttp1.open("GET", "/cgi-bin/pyknop2.py", true);
            xhttp1.send();

            var button1 = document.getElementById("button1");
            button1.classList.remove("button1-green");
            button1.classList.add("button1-red");
            var button2 = document.getElementById("button2");
            button2.classList.remove("button2-red");
            button2.classList.add("button2-green");

            setTimeout(function () {
                button2.classList.remove("button2-green");
                button2.classList.add("button2-orange");
                setTimeout(function () {
                    button2.classList.remove("button2-orange");
                    button1.classList.remove("button1-red");
                }, 2000);
            }, 5000);
        }
    </script>


</head>

<body>
    <table>
        <tr>
            <td><input type="button" id="button1" value="Button 1"
onclick="doButton1()"></td>
            <td>
                <div id="status1"></div>
            </td>
        </tr>
        <tr>
            <td><input type="button" id="button2" value="Button 2"
onclick="doButton2()"></td>
            <td>
```

```
                <div id="status2"></div>
            </td>
        </tr>
        <tr>
            <td><input type="button" id="button3" value="Button 3"
onclick="dinges()"></td>
            <td>
                <div id="status2"></div>
            </td>
        </tr>
    </table>
</body>

</html>
```

Python codes:

Code 1:

```
#!/usr/bin/env python3
import socket
print("Content-Type: text/plain")
print("")
print("This data comes from button.py")

HOST = 'broodpi3b'  # or the IP address of the server
PORT = 4567         # Port number

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
s.send(b'k')    # Send 'k' to C-server
s.close()
```

Code 2:

```
#!/usr/bin/env python3
import socket
print("Content-Type: text/plain")
print("")
print("This data comes from button.py")

HOST = 'broodpi3b'  # or the IP address of the server
PORT = 4567         # Port number

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))
s.send(b'K')    # Send 'K' to C-server
s.close()
```

C code:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <pthread.h>

#define PORT 4567 // mn poort

void set_gpio_state(const char *gpio_pin_path, const char *state)
{
    int file_descriptor = open(gpio_pin_path, O_WRONLY);
    write(file_descriptor, state, strlen(state));
    close(file_descriptor);
}
// functies voor aan/uit voor elke led
void rood2Aan()
{
    set_gpio_state("/sys/class/gpio/gpio16/value", "0");
}
void rood2Uit()
{
    set_gpio_state("/sys/class/gpio/gpio16/value", "1");
}
void geel2Aan()
{
    set_gpio_state("/sys/class/gpio/gpio20/value", "0");
}
void geel2Uit()
{
    set_gpio_state("/sys/class/gpio/gpio20/value", "1");
}
void groen2Aan()
{
    set_gpio_state("/sys/class/gpio/gpio21/value", "0");
}
void groen2Uit()
{
    set_gpio_state("/sys/class/gpio/gpio21/value", "1");
```

```
}
void rood1Aan()
{
    set_gpio_state("/sys/class/gpio/gpio13/value", "0");
}
void rood1Uit()
{
    set_gpio_state("/sys/class/gpio/gpio13/value", "1");
}
void geel1Aan()
{
    set_gpio_state("/sys/class/gpio/gpio19/value", "0");
}
void geel1Uit()
{
    set_gpio_state("/sys/class/gpio/gpio19/value", "1");
}
void groen1Aan()
{
    set_gpio_state("/sys/class/gpio/gpio26/value", "0");
}
void groen1Uit()
{
    set_gpio_state("/sys/class/gpio/gpio26/value", "1");
}

void allesUit()
{
    geel1Uit();
    geel2Uit();
    rood1Uit();
    rood2Uit();
    groen1Uit();
    groen2Uit();
}

void lightOne()
{
    rood1Aan();
    groen2Aan();
    for (int i = 0; i < 5; i++)
    {
        sleep(1);
    }
    groen2Uit();
```

```
    geel2Aan();
    for (int i = 0; i < 2; i++)
    {
        sleep(1);
    }
    rood1Uit();
    geel2Uit();
}

void lightTwo()
{
    rood2Aan();
    groen1Aan();
    for (int i = 0; i < 5; i++)
    {
        sleep(1);
    }
    groen1Uit();
    geel1Aan();
    for (int i = 0; i < 2; i++)
    {
        sleep(1);
    }
    rood2Uit();
    geel1Uit();
}

void *buttonOne(void *arg)
{
    int n1, n2;
    int file_descriptorB1, file_descriptorB2;
    char value1[1024];
    char value2[1024];
    while (1)
    {
        file_descriptorB1 = open("/sys/class/gpio/gpio2/value", O_RDONLY);
        read(file_descriptorB1, value1, sizeof(value1));
        close(file_descriptorB1);
        n1 = strcmp(value1, "1");
        if (n1 == 10)
        {
            printf("\nButton 1 pressed!\n");
            lightOne();
        }
        usleep(100000); // wacht 100ms voordat je opnieuw controleert
```

```
        file_descriptorB2 = open("/sys/class/gpio/gpio3/value", O_RDONLY);
        read(file_descriptorB2, value2, sizeof(value2));
        close(file_descriptorB2);
        n2 = strcmp(value2, "1");
        if (n2 == 10)
        {
            printf("\nButton 2 pressed!\n");
            lightTwo();
        }
        usleep(100000); // wacht 100ms voordat je opnieuw controleert
    }
}

int main()
{
    int file_descriptor1, file_descriptor2, file_descriptor3, file_descriptor4,
file_descriptor5, file_descriptor6;
    int kD1 = 0;
    int kD2 = 0;
    int n, n3, n4;
    int valread, valread1;
    int server_fd, new_socket, file_descriptor;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char buffer1[1024] = {0};
    const char *hello = "Hello from server";
    printf("\nStart!\n");
    file_descriptor1 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor1, "16", 2);
    close(file_descriptor1);
    file_descriptor1 = open("/sys/class/gpio/gpio16/direction", O_WRONLY);
    write(file_descriptor1, "out", 3);
    close(file_descriptor1);

    file_descriptor2 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor2, "20", 2);
    close(file_descriptor2);
    file_descriptor2 = open("/sys/class/gpio/gpio20/direction", O_WRONLY);
    write(file_descriptor2, "out", 3);
    close(file_descriptor2);

    file_descriptor3 = open("/sys/class/gpio/export", O_WRONLY);
```

```
    write(file_descriptor3, "21", 2);
    close(file_descriptor3);
    file_descriptor3 = open("/sys/class/gpio/gpio21/direction", O_WRONLY);
    write(file_descriptor3, "out", 3);
    close(file_descriptor3);
    // 1
    file_descriptor4 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor4, "13", 2);
    close(file_descriptor4);
    file_descriptor4 = open("/sys/class/gpio/gpio13/direction", O_WRONLY);
    write(file_descriptor4, "out", 3);
    close(file_descriptor4);

    file_descriptor5 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor5, "19", 2);
    close(file_descriptor5);
    file_descriptor5 = open("/sys/class/gpio/gpio19/direction", O_WRONLY);
    write(file_descriptor5, "out", 3);
    close(file_descriptor5);

    file_descriptor6 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor6, "26", 2);
    close(file_descriptor6);
    file_descriptor6 = open("/sys/class/gpio/gpio26/direction", O_WRONLY);
    write(file_descriptor6, "out", 3);
    close(file_descriptor6);

    allesUit();
    // Zet alles uit

    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Socket opties
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                   &opt, sizeof(opt)))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }

    // Adress geven
```

```
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    // Binden
    if (bind(server_fd, (struct sockaddr *)&address,
             sizeof(address)) < 0)
    {
        if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                                 (socklen_t *)&addrlen)) < 0)
        {
            perror("accept");
            exit(EXIT_FAILURE);
        }
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    // Luisteren
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }

    // Knop monitoren in aparte thread
    pthread_t buttonOne_T;
    pthread_create(&buttonOne_T, NULL, buttonOne, NULL);

    // De while loop (met stoplicht)
    while (1)
    {
        if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                                 (socklen_t *)&addrlen)) < 0)
        {
            perror("accept");
            exit(EXIT_FAILURE);
        }
        valread = recv(new_socket, buffer, 1024, 0);
        n3 = strcmp(buffer, "1");
        printf("\nReceived %d bytes: %s, waarde: %d\n", valread, buffer, n3);
        if (n3 == 58)
        {
            lightOne();
        }
```

```
        if (n3 == 26)
        {
            lightTwo();
        }
        close(new_socket);
    }
    return 0;
}
```

Bronvermeldingen:

1. https://www.tutorialspoint.com/system-function-in-c-cplusplus
2. https://stackoverflow.com/questions/2180079/how-can-i-copy-a-file-on-unix-using-c
3. http://www.linuxhowtos.org/C_C++/socket.htm
4. https://www.geeksforgeeks.org/multithreading-in-c/

Opdracht 5 (4 uur) – Security

Beschrijving van beveiligingsrisico's en mogelijke oplossingen:

Er zijn veel verschillende beveiligingsrisico's, zoals toegang van onbekende users, het afluisteren/onderscheppen van data en specifiek gerichte aanvallen. Om een protocol te beveiligen kunnen we gebruik maken van public en private key's (AES, RES, DES, ECC), maar we kunnen ook rolling key's security gebruiken.

In deze opdracht is het gebruik van rolling key's wat makkelijker. De ontvanger en verzender hebben beide een lijst met allerlei data, zoals 12, 43, etc. De data die verzonden wordt, moet eerst door een XOR-functie. Deze XOR'd de data die we willen verzenden met een van de data van de sleutellijst, daarna wordt de index van zowel de verzender als ontvanger de sleutellijst met +1 verhoogd. Hieruit komt data dat zonder risico's verzonden kan worden, aangezien de data niks essentieels bevat. Echter kan het toch fout gaan, als bijvoorbeeld de hacker de lijst met sleutels heeft gevonden. In dat geval moet de lijst zo snel mogelijk opnieuw worden gemaakt/gegenereerd.

Python code:

```python
import socket

HOST = 'broodpi3b'  # or the IP address of the server
PORT = 4567         # Port number

rolling_keys = ['00001100', '00101011', '01001101', '00001010']

# Read the next key index from the 'keys' file
with open('keys', 'r') as f:
    key_index = int(f.read().strip())

# XOR function
def xor(a, b):
    return ''.join(str(int(x) ^ int(y)) for x, y in zip(a, b))

# Connect to the server
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect((HOST, PORT))

# Char to binary
k_bin = bin(ord('k'))[2:].zfill(8)

# Encrypt 'k' with the rolling keys
encrypted_k = xor(k_bin, rolling_keys[key_index])
key_index = (key_index + 1) % len(rolling_keys)

# Send the encrypted binary data
```

```
s.sendall(encrypted_k.encode())

# Write the next rolling key index to the keys file
with open('keys', 'w') as f:
    f.write(str(key_index))

s.close() # Close connection
```

C code:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <errno.h>
#include <pthread.h>

#define PORT 4567 // mn poort

long bin_to_dec(char *bin)
{
    return strtol(bin, NULL, 2);
}

void xor (char *a, char *b, char *result) {
    int i;
    for (i = 0; i < strlen(a); i++)
    {
        result[i] = ((a[i] - '0') ^ (b[i] - '0')) + '0';
    }
    result[i] = '\0';
}

    void set_gpio_state(const char *gpio_pin_path, const char *state)
{
    int file_descriptor = open(gpio_pin_path, O_WRONLY);
    write(file_descriptor, state, strlen(state));
    close(file_descriptor);
}
// functies voor aan/uit voor elke led
void rood2Aan()
```

```
{
    set_gpio_state("/sys/class/gpio/gpio16/value", "0");
}
void rood2Uit()
{
    set_gpio_state("/sys/class/gpio/gpio16/value", "1");
}
void geel2Aan()
{
    set_gpio_state("/sys/class/gpio/gpio20/value", "0");
}
void geel2Uit()
{
    set_gpio_state("/sys/class/gpio/gpio20/value", "1");
}
void groen2Aan()
{
    set_gpio_state("/sys/class/gpio/gpio21/value", "0");
}
void groen2Uit()
{
    set_gpio_state("/sys/class/gpio/gpio21/value", "1");
}
void rood1Aan()
{
    set_gpio_state("/sys/class/gpio/gpio13/value", "0");
}
void rood1Uit()
{
    set_gpio_state("/sys/class/gpio/gpio13/value", "1");
}
void geel1Aan()
{
    set_gpio_state("/sys/class/gpio/gpio19/value", "0");
}
void geel1Uit()
{
    set_gpio_state("/sys/class/gpio/gpio19/value", "1");
}
void groen1Aan()
{
    set_gpio_state("/sys/class/gpio/gpio26/value", "0");
}
void groen1Uit()
{
```

```
    set_gpio_state("/sys/class/gpio/gpio26/value", "1");
}

void allesUit()
{
    geel1Uit();
    geel2Uit();
    rood1Uit();
    rood2Uit();
    groen1Uit();
    groen2Uit();
}

void lightOne()
{
    rood1Aan();
    groen2Aan();
    for (int i = 0; i < 5; i++)
    {
        sleep(1);
    }
    groen2Uit();
    geel2Aan();
    for (int i = 0; i < 2; i++)
    {
        sleep(1);
    }
    rood1Uit();
    geel2Uit();
}

void lightTwo()
{
    rood2Aan();
    groen1Aan();
    for (int i = 0; i < 5; i++)
    {
        sleep(1);
    }
    groen1Uit();
    geel1Aan();
    for (int i = 0; i < 2; i++)
    {
        sleep(1);
    }
```

```
    rood2Uit();
    geel1Uit();
}

void *buttonOneTwo()
{
    int n1, n2;
    int file_descriptorB1, file_descriptorB2;
    char value1[1024];
    char value2[1024];
    while (1)
    {
        file_descriptorB1 = open("/sys/class/gpio/gpio2/value", O_RDONLY);
        read(file_descriptorB1, value1, sizeof(value1));
        close(file_descriptorB1);
        n1 = strcmp(value1, "1");
        if (n1 == 10)
        {
            printf("\nButton 1 pressed!\n");
            system("python /home/semihpi/Codon/kopi1.py");
            system("python /home/semihpi/Codon/kopi2.py");
            lightOne();
        }
        else
        {
            system("python /home/semihpi/Codon/kopi1.py");
            system("python /home/semihpi/Codon/kopi2.py");
        }
        usleep(100000); // wacht 100ms voordat je opnieuw controleert

        file_descriptorB2 = open("/sys/class/gpio/gpio3/value", O_RDONLY);
        read(file_descriptorB2, value2, sizeof(value2));
        close(file_descriptorB2);
        n2 = strcmp(value2, "1");
        if (n2 == 10)
        {
            printf("\nButton 2 pressed!\n");
            system("python /home/semihpi/Codon/kopi1.py");
            system("python /home/semihpi/Codon/kopi2.py");
            lightTwo();
        }
        else
        {
            system("python /home/semihpi/Codon/kopi2.py");
            system("python /home/semihpi/Codon/kopi1.py");
```

```
        }
        usleep(100000);
    }
}

int main()
{
    char decryptedBin[1024] = {0};
    int key_index = 0;
    int file_descriptor1, file_descriptor2, file_descriptor3, file_descriptor4,
file_descriptor5, file_descriptor6;
    int kD1 = 0;
    int kD2 = 0;
    int n, n3, n4;
    int valread, valread1;
    int server_fd, new_socket, file_descriptor;
    struct sockaddr_in address;
    int opt = 1;
    int addrlen = sizeof(address);
    char buffer[1024] = {0};
    char buffer1[1024] = {0};
    const char *hello = "Hello from server";

    char *rolling_keys[] = {"00001100", "00101011", "01001101", "00001010"};
    printf("\n1\n");
    file_descriptor1 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor1, "16", 2);
    close(file_descriptor1);
    file_descriptor1 = open("/sys/class/gpio/gpio16/direction", O_WRONLY);
    write(file_descriptor1, "out", 3);
    close(file_descriptor1);

    file_descriptor2 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor2, "20", 2);
    close(file_descriptor2);
    file_descriptor2 = open("/sys/class/gpio/gpio20/direction", O_WRONLY);
    write(file_descriptor2, "out", 3);
    close(file_descriptor2);

    file_descriptor3 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor3, "21", 2);
    close(file_descriptor3);
    file_descriptor3 = open("/sys/class/gpio/gpio21/direction", O_WRONLY);
    write(file_descriptor3, "out", 3);
    close(file_descriptor3);
```

```
    // 1
    file_descriptor4 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor4, "13", 2);
    close(file_descriptor4);
    file_descriptor4 = open("/sys/class/gpio/gpio13/direction", O_WRONLY);
    write(file_descriptor4, "out", 3);
    close(file_descriptor4);

    file_descriptor5 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor5, "19", 2);
    close(file_descriptor5);
    file_descriptor5 = open("/sys/class/gpio/gpio19/direction", O_WRONLY);
    write(file_descriptor4, "out", 3);
    close(file_descriptor5);

    file_descriptor6 = open("/sys/class/gpio/export", O_WRONLY);
    write(file_descriptor6, "26", 2);
    close(file_descriptor6);
    file_descriptor6 = open("/sys/class/gpio/gpio26/direction", O_WRONLY);
    write(file_descriptor6, "out", 3);
    close(file_descriptor6);

    allesUit(); // Zet alles uit
    // Maak de socket aan
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
    {
        perror("socket failed");
        exit(EXIT_FAILURE);
    }

    // Socket opties
    if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
                   &opt, sizeof(opt)))
    {
        perror("setsockopt");
        exit(EXIT_FAILURE);
    }

    // Adress geven (port)
    address.sin_family = AF_INET;
    address.sin_addr.s_addr = INADDR_ANY;
    address.sin_port = htons(PORT);

    // Binden aan socket
    if (bind(server_fd, (struct sockaddr *)&address,
```

```
             sizeof(address)) < 0)
    {
        if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                                 (socklen_t *)&addrlen)) < 0)
        {
            perror("accept");
            exit(EXIT_FAILURE);
        }
        perror("bind failed");
        exit(EXIT_FAILURE);
    }

    // Luisteren...
    if (listen(server_fd, 3) < 0)
    {
        perror("listen");
        exit(EXIT_FAILURE);
    }

    // Knop monitoren in aparte thread (multithreading)
    pthread_t buttonOneTwo_T;
    pthread_create(&buttonOneTwo_T, NULL, buttonOneTwo, NULL);

    // De while loop
    while (1)
    {
        if ((new_socket = accept(server_fd, (struct sockaddr *)&address,
                                 (socklen_t *)&addrlen)) < 0)
        {
            perror("accept");
            exit(EXIT_FAILURE);
        }
        valread = recv(new_socket, buffer, 1024,
0);                                                 // Lees waarde uit
        xor(buffer, rolling_keys[key_index],
decryptedBin);                                          // XOR'd de message
(decryption)
        char decryptedMessage =
(char)bin_to_dec(decryptedBin);                                     // Zet
bin > dec
        printf("decrypted message: %c\n",
decryptedMessage);                                    // Print decrypted
message (for debugging)
```

```
        printf("decrypted binary message: %s\n",
decryptedBin);                                      // Print decrypted binary
message (for debugging)
        n3 = strcmp(buffer,
"1");                                                                   //
Zet n3 (afgelezen waarden (1)) om in een datatype int met strcmp()
        printf("\nServer received message: %d, aantal bytes: %s en waarde: %d\n",
valread, buffer, n3); // Print dat de server iets ontvangen heeft
        if (key_index ==
3)                                                                      //
Check of de index van de rolling keys het maximale index heeft bereikt
        {
            key_index = 0;
        }
        else
        {
            key_index++;
        }
        if (n3 == 58 || decryptedMessage == 'k') // Check of de gelezen waarden
== 58 OF decryptedMessage == 'k'
        {
            lightOne();
        }
        if (n3 == 26 || decryptedMessage == 'K') // Check of de gelezen waarden
== 26 OF decryptedMessage == 'K'
        {
            lightTwo();
        }
        close(new_socket); // Sluit de socketVerbinding
    }
    return 0;
}
```

Bronvermeldingen:

1. https://www.ictportal.nl/ict-lexicon/it-security
2. https://iopscience.iop.org/article/10.1088/1742-6596/1566/1/012074
3. https://www.youtube.com/watch?v=pvII6_O6KAc
4. https://www.programmingalgorithms.com/algorithm/xor-encryption/c/
5. https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_xor_process.htm

Opdracht 6 (2 uur) – API

EISEN:

We willen dat de API moet voldoen aan securityvereisten, zoals het gebruik van de rolling key encryptie. Hiervoor gebruiken we een lijst met rollende sleutels die alleen de daemon en de pythonscripts mogen hebben. De API moet niet moeilijk te begrijpen zijn, want anders zal niemand deze API gebruiken. Daarnaast moet de API ook de vereisten functies kunnen uitvoeren die hieronder allemaal staan uitgelegd (API beschrijving).

API beschrijving

Functie 1: *bin_to_dec:*

Deze functie zet de binaire string om in een decimale int datatype.

Endpoint: /bin_to_dec

Method: POST

Input parameter:

- bin: char en pointer datatype (binair).

Returnwaarde:

- 'long' datatype dat gelijk is aan de uitkomst van het decimale getal.

De functie:

```
long bin_to_dec(char *bin)
{
    return strtol(bin, NULL, 2);
}
```

De 'strtol' bibliotheekfunctie is een onderdeel van de stdio.h library en wordt gebruikt om 'bin' om te zetten in een decimaal getal (long).

Functie 2: *xor*

Deze functie voert een bitwise XOR-operatie uit op de twee gegeven binaire strings en slaat het resultaat op in de derde string.

Endpoint: /xor

Method: POST

Input parameters:

- a: char pointer naar de eerste binaire string
- b: char pointer naar de tweede binaire string
- result: char pointer naar de string waarin het resultaat wordt opgeslagen

Returnwaarde:

- Deze functie heeft geen returnwaarde, want het resultaat wordt in de result variabele opgeslagen.

De functie:

```
void xor (char *a, char *b, char *result) {
    int i;
    for (i = 0; i < strlen(a); i++)
    {
        result[i] = ((a[i] - '0') ^ (b[i] - '0')) + '0';
    }
    result[i] = '\0';
}
```

Functie 3: *set_gpio_state*
Deze functie stelt de GPIO pin op de opgegeven locatie in op de opgegeven staat.
Endpoint: /set_gpio_state
Method: POST
Input parameters:
- gpio_pin_path: een constante pointer naar de pad naam van de GPIO pin.
- state: een constante pointer naar een string die de staat van de GPIO pin voorstelt. Het kan de waarde "0" of "1" bevatten.

Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void set_gpio_state(const char *gpio_pin_path, const char *state)
{
    int file_descriptor = open(gpio_pin_path, O_WRONLY);
    write(file_descriptor, state, strlen(state));
    close(file_descriptor);
}
```

Functie 4: *rood2aan*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 2, rode LED wordt veranderd naar: ‘aan’.
Endpoint: /rood2aan
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void rood2Aan()
{
    set_gpio_state("/sys/class/gpio/gpio16/value", "0");
}
```

Functie 5: *rood2uit*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 2, rode LED wordt veranderd naar: ‘uit’.
Endpoint: /rood2uit
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void rood2Uit()
{
    set_gpio_state("/sys/class/gpio/gpio16/value", "1");
}
```

Functie 6: *geel2aan*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 2, gele LED wordt veranderd naar: ‘aan’.
Endpoint: /geel2aan
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void geel2Aan()
{
    set_gpio_state("/sys/class/gpio/gpio20/value", "0");
}
```

Functie 7: *geel2uit*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 2, gele LED wordt veranderd naar: ‘uit’.
Endpoint: /geel2uit
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void geel2Uit()
{
    set_gpio_state("/sys/class/gpio/gpio20/value", "1");
}
```

Functie 8: *groen2aan*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 2, groene LED wordt veranderd naar: ‘aan’.
Endpoint: /groen2aan
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void groen2Aan()
{
    set_gpio_state("/sys/class/gpio/gpio21/value", "0");
}
```

Functie 9: *groen2uit*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 2, groene LED wordt veranderd naar: ‘uit’.
Endpoint: /groen2uit
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void groen2Uit()
{
    set_gpio_state("/sys/class/gpio/gpio21/value", "1");
}
```

Functie 10: *rood1Aan*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 1, rode LED wordt veranderd naar: ‘aan’.
Endpoint: /rood1Aan
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void rood1Aan()
{
    set_gpio_state("/sys/class/gpio/gpio13/value", "0");
}
```

Functie 11: *rood1Uit*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 1, rode LED wordt veranderd naar: ‘uit’.
Endpoint: /rood1Uit
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void rood1Uit()
{
    set_gpio_state("/sys/class/gpio/gpio13/value", "1");
}
```

Functie 12: *geel1Aan*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 1, gele LED wordt veranderd naar: ‘aan’.
Endpoint: /geel1Aan
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void geel1Aan()
{
    set_gpio_state("/sys/class/gpio/gpio19/value", "0");
}
```

Functie 13: *geel1Uit*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 1, gele LED wordt veranderd naar: ‘uit’.
Endpoint: /geel1Uit
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void geel1Uit()
{
    set_gpio_state("/sys/class/gpio/gpio19/value", "1");
}
```

Functie 14: *groen1Aan*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 1, groene LED wordt veranderd naar: ‘aan’.
Endpoint: /groen1Aan
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void groen1Aan()
{
    set_gpio_state("/sys/class/gpio/gpio26/value", "0");
}
```

Functie 15: *groen1Uit*
Deze functie zorgt ervoor dat de GPIO state van stoplicht 1, groene LED wordt veranderd naar: ‘uit’.
Endpoint: /groen1Uit
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void groen1Uit()
{
    set_gpio_state("/sys/class/gpio/gpio26/value", "1");
}
```

Functie 16: *allesUit*
Deze functie zorgt ervoor dat alle GPIO states van beide stoplichten, alle LEDs worden veranderd naar: ‘uit’.
Endpoint: /allesUit
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void allesUit()
{
    geel1Uit();
    geel2Uit();
    rood1Uit();
    rood2Uit();
    groen1Uit();
    groen2Uit();
}
```

Functie 17: *lightOne*
Deze functie zorgt ervoor dat er een stoplicht-cyclus gaat draaien, waarbij aanvankelijk stoplicht 1 op rood staat en stoplicht 2 op groen staat.
Endpoint: /lightOne
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void lightOne()
{
    rood1Aan();
    groen2Aan();
    for (int i = 0; i < 5; i++)
```

```
    {
        sleep(1);
    }
    groen2Uit();
    geel2Aan();
    for (int i = 0; i < 2; i++)
    {
        sleep(1);
    }
    rood1Uit();
    geel2Uit();
}
```

Functie 18: *lightTwo*
Deze functie zorgt ervoor dat er een stoplicht-cyclus gaat draaien, waarbij aanvankelijk stoplicht 2 op rood staat en stoplicht 1 op groen staat.
Endpoint: /lightTwo
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void lightTwo()
{
    rood2Aan();
    groen1Aan();
    for (int i = 0; i < 5; i++)
    {
        sleep(1);
    }
    groen1Uit();
    geel1Aan();
    for (int i = 0; i < 2; i++)
    {
        sleep(1);
    }
    rood2Uit();
    geel1Uit();
}
```

Functie 19: **buttonOneTwo*
Deze functie zorgt ervoor dat de input van de knopjes continue worden gecheckt of ze zijn ingedrukt of niet. Bij deze functie wordt er gebruik gemaakt van multi-threading. Hier voor wordt de <pthread.h> C library gebruikt.
Endpoint: /buttonOneTwo
Method: POST
Input parameters: De functie heeft geen input parameters.
Returnwaarde: De functie heeft geen returnwaarde.
De functie:

```
void *buttonOneTwo()
{
    int n1, n2;
    int file_descriptorB1, file_descriptorB2;
    char value1[1024];
    char value2[1024];
    while (1)
    {
        file_descriptorB1 = open("/sys/class/gpio/gpio2/value", O_RDONLY);
        read(file_descriptorB1, value1, sizeof(value1));
        close(file_descriptorB1);
        n1 = strcmp(value1, "1");
        if (n1 == 10)
        {
            printf("\nButton 1 pressed!\n");
            system("python /home/semihpi/Codon/kopi1.py");
            system("python /home/semihpi/Codon/kopi2.py");
            lightOne();
        }
        else
        {
            system("python /home/semihpi/Codon/kopi1.py");
            system("python /home/semihpi/Codon/kopi2.py");
        }
        usleep(100000); // wacht 100ms voordat je opnieuw controleert

        file_descriptorB2 = open("/sys/class/gpio/gpio3/value", O_RDONLY);
        read(file_descriptorB2, value2, sizeof(value2));
        close(file_descriptorB2);
        n2 = strcmp(value2, "1");
        if (n2 == 10)
        {
            printf("\nButton 2 pressed!\n");
            system("python /home/semihpi/Codon/kopi1.py");
            system("python /home/semihpi/Codon/kopi2.py");
            lightTwo();
```

```
        }
        else
        {
            system("python /home/semihpi/Codon/kopi2.py");
            system("python /home/semihpi/Codon/kopi1.py");
        }
        usleep(100000);
    }
}
```

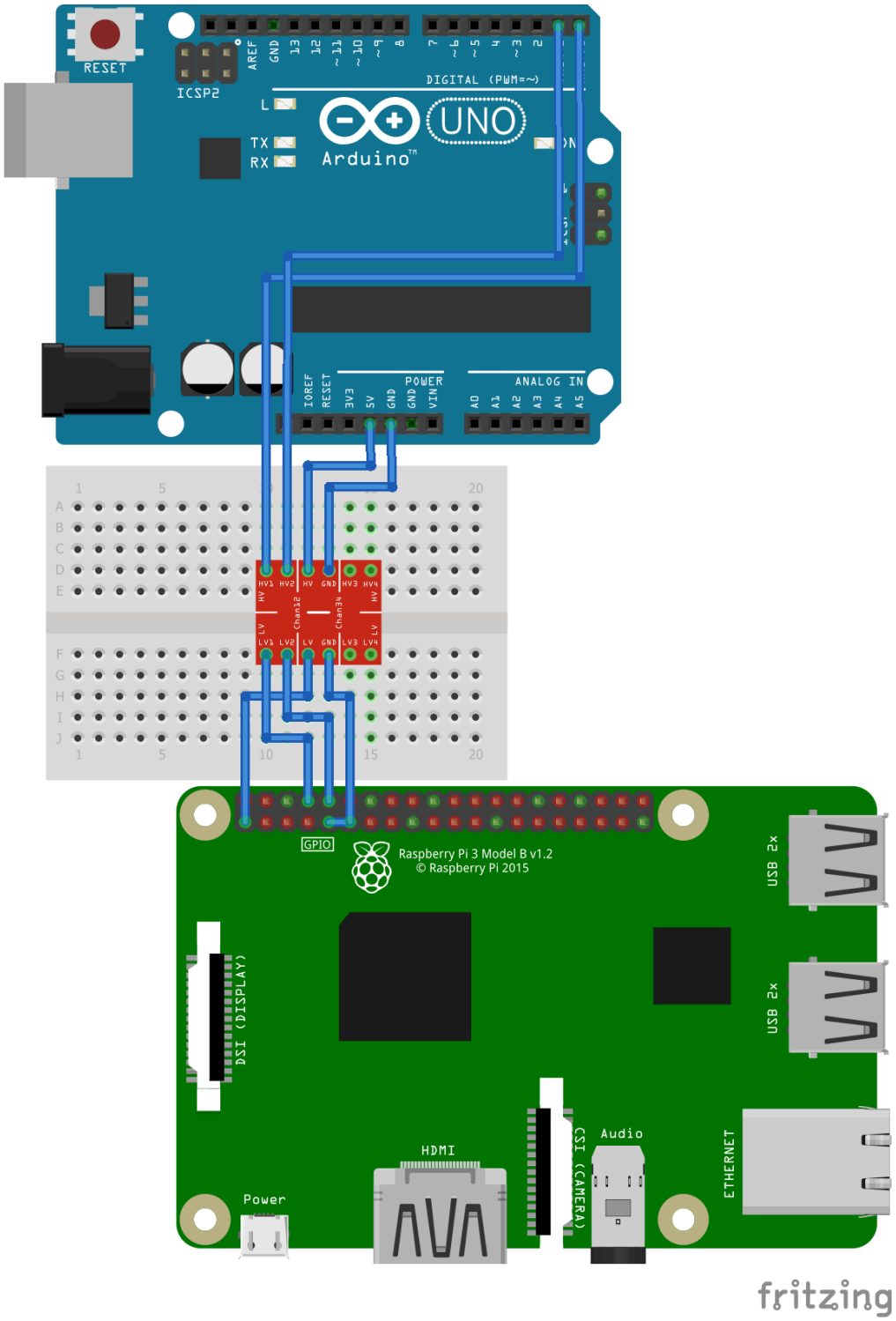
Verschillende opties voor de API
Uit onderzoek hebben we gevonden dat als we een API willen bouwen in python, dat Flask en FastAPI goede opties zijn. Voor C is een eigen socket server eigenlijk al genoeg, aangezien je via scripts allerlei commando's kunt posten naar de socket server. Daarnaast kunnen we ook RESTful API in combinatie met FastCGI ook een krachtige web gebaseerde API maken, maar is dat een stuk minder handig, omdat het gebruikt maakt van HTTP protocollen. Als we deze API zouden implementeren, zouden we de POST commando's ook graag willen gebruiken.

Onderbouwing van onze keuze
Voor ons geval is een socket server in principe genoeg voor een lokale API. Andere API's zijn voor in dit geval een beetje overkill. We hebben hierboven de documentatie van de API gegeven. Het implementeren van de POST commando's is wel leuk, maar is voor ons geen vereiste.

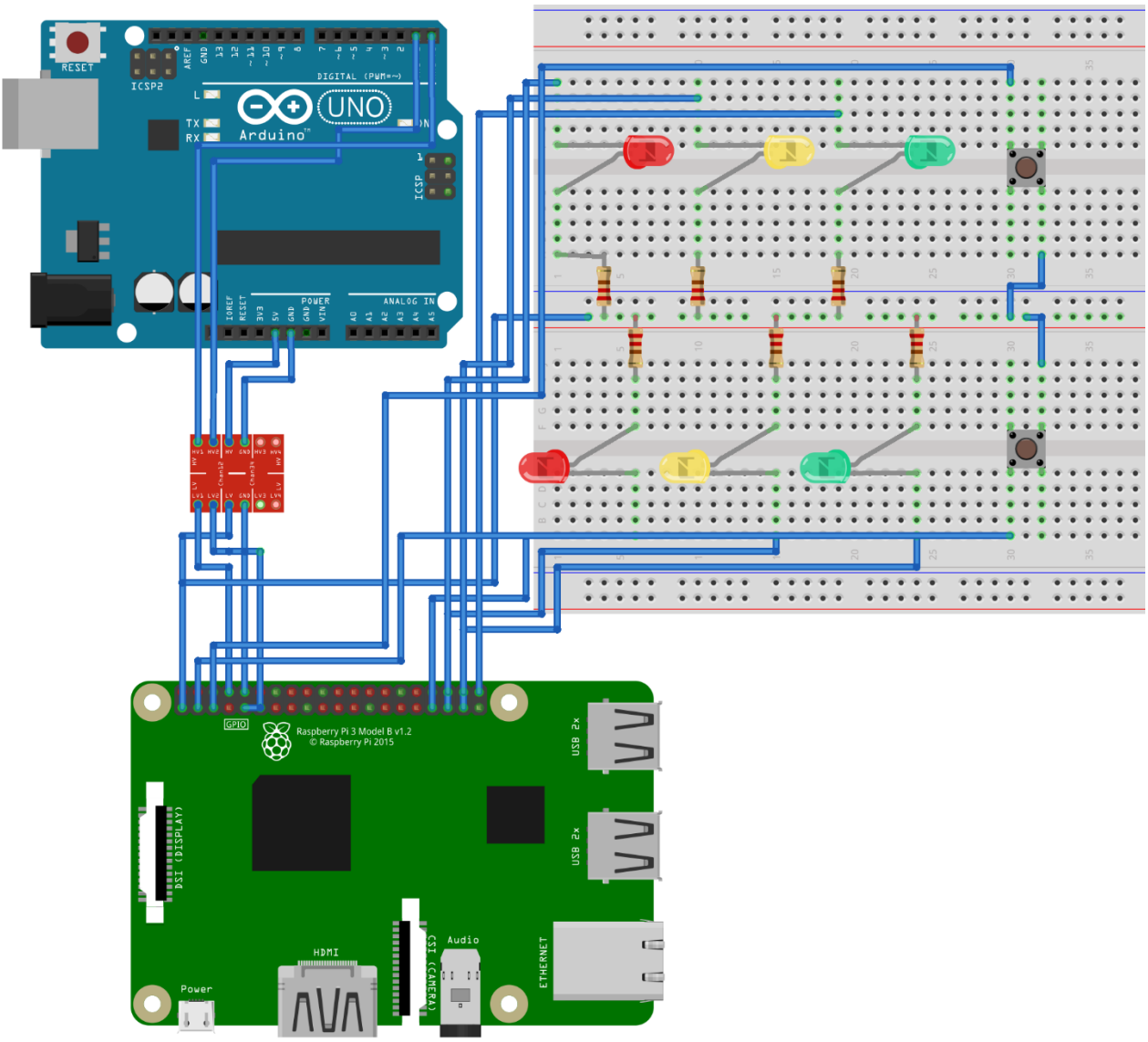
Problem 5 – Arduino

Opdracht 2 (1 uur) - Arduino aansluiten aan de Raspberry Pi

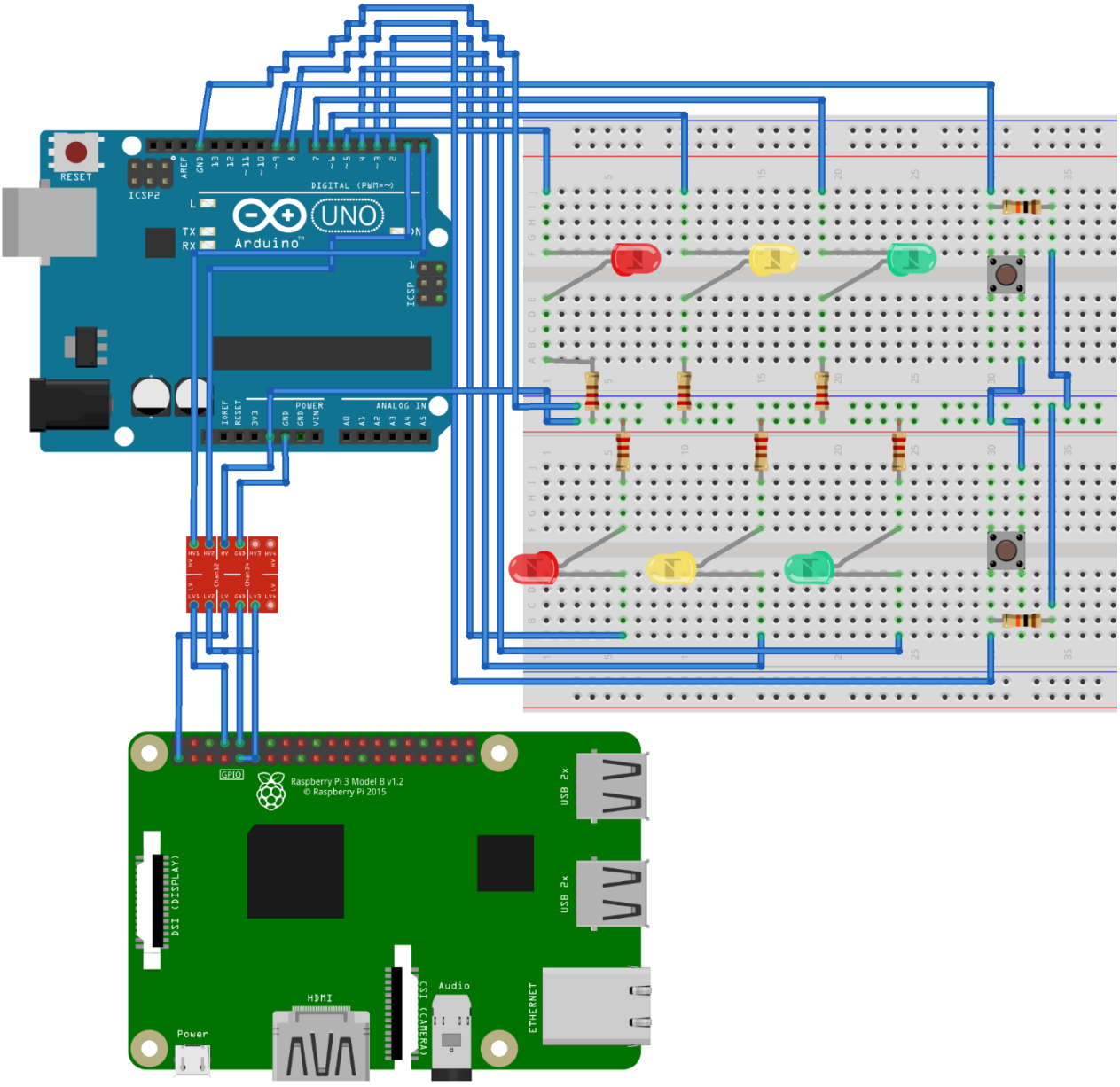
Schema van de verbinding tussen de Raspberry Pi en de Arduino (full-duplex – Rx/Tx):



Schema van de verbindingen tussen de stoplichten, knoppen, en de Arduino (vanaf Pi kant aangesloten):



Schema van de verbindingen tussen de stoplichten, knoppen, en de Arduino (vanaf Arduino kant aangesloten) (1/2 - Schema):



Schema van de verbindingen tussen de stoplichten, knoppen, en de Arduino (vanaf Arduino kant aangesloten) (2/2 – Bijbehorende code):

```
#define OFF 0
#define ON 1

/* Define and initialise LED pins for both traffic lights */
int roodLichtPin1 = 2;
int geelLichtPin1 = 3;
int groenLichtPin1 = 4;
int roodLichtPin2 = 5;
int geelLichtPin2 = 6;
int groenLichtPin2 = 7;

/* Define and initialise buttons for traffic lights one and two */
int knop1Pin = 8;
int knop2Pin = 9;

/* Set knop1 and knop2 to be initially OFF */
int knop1Status = OFF;
int knop2Status = OFF;

void setup() {
  // Set all LEDs te be output pins
  pinMode(roodLichtPin1, OUTPUT);
  pinMode(geelLichtPin1, OUTPUT);
  pinMode(groenLichtPin1, OUTPUT);
  pinMode(roodLichtPin2, OUTPUT);
  pinMode(geelLichtPin2, OUTPUT);
  pinMode(groenLichtPin2, OUTPUT);
  /* Set knop1 and knop2 to be INPUT pins */
  pinMode(knop1Pin, INPUT);
  pinMode(knop2Pin, INPUT);
}
/* Traffic light buttons reader */
void knop_status_reader() {
  knop1Status = digitalRead(knop1Pin);
  knop2Status = digitalRead(knop2Pin);
}
void alles_uit() {
  digitalWrite(groenLichtPin1, LOW);
  digitalWrite(groenLichtPin2, LOW);
  digitalWrite(geelLichtPin1, LOW);
  digitalWrite(geelLichtPin2, LOW);
  digitalWrite(roodLichtPin1, LOW);
  digitalWrite(roodLichtPin2, LOW);
```

```
}
/* Traffic light one cycle */
void knop1_cycle() {
  digitalWrite(groenLichtPin1, HIGH);
  digitalWrite(roodLichtPin2, HIGH);
  for (int tijd = 0; tijd < 500; tijd++) {
    delay(10);
    knop_status_reader();
  }
  digitalWrite(groenLichtPin1, LOW);
  digitalWrite(geelLichtPin1, HIGH);
  for (int tijd = 0; tijd < 300; tijd++) {
    delay(10);
    knop_status_reader();
  }
  digitalWrite(geelLichtPin1, LOW);
  digitalWrite(roodLichtPin2, LOW);
}
/* Traffic Light Two Cycle */
void knop2_cycle() {
  digitalWrite(groenLichtPin2, HIGH);
  digitalWrite(roodLichtPin1, HIGH);
  for (int tijd = 0; tijd < 500; tijd++) {
    delay(10);
    knop_status_reader();
  }
  digitalWrite(groenLichtPin2, LOW);
  digitalWrite(geelLichtPin2, HIGH);
  for (int tijd = 0; tijd < 300; tijd++) {
    delay(10);
    knop_status_reader();
  }
  digitalWrite(geelLichtPin2, LOW);
  digitalWrite(roodLichtPin1, LOW);
}
void loop() {
  knop_status_reader();              // Read status of knopjes 1 & 2
  if (knop1Status == HIGH) {         // If knop 1 is pressed, then set TL1 on
green and TL2 on red, etc.
    alles_uit();                     // Set everything off.
    knop1_cycle();                   // Call cycle one
  } else if (knop2Status == HIGH) {  // If knop 2 is pressed, then set TL2 on
green and TL1 on red, etc.
    alles_uit();                     // Set everything off.
    knop2_cycle();                   // Call cycle two
```

```
  } else {
    /* Keep cycling between two trafficlights */
    knop1_cycle();
    knop2_cycle();
  }
}
```

Opdracht 3 (4 uur) – Protocol

UART protocol:
UART (Universal Asynchronous Receiver-Transmitter) is een seriële communicatieprotocol dat wordt gebruikt voor het verzenden (TX) en ontvangen (RX) van gegevens tussen twee apparaten. Het gebruikt de RX en TX pinnen voor zowel de verzender en ontvanger. De voordelen van het UART-protocol zijn de eenvoud en de lage overhead, omdat het protocol geen gecompliceerde communicatie vereist. Een nadeel is dat het protocol niet in staat is om meerdere apparaten te ondersteunen zonder extra hardware of software om de communicatie te regelen.

I^2C (A.K.A. IIC, I2C) protocol:
I2C (Inter-Integrated Circuit) is een seriële communicatieprotocol dat wordt gebruikt voor het verzenden en ontvangen van gegevens tussen apparaten op een printplaat of binnen een systeem. Het protocol gebruikt twee datakabels, SDA (Serial Data Line) en SCL (Serial Clock Line), om de gegevensstroom te synchroniseren en te verzenden. Een voordeel van het I2C-protocol is dat het in staat is om meerdere apparaten te ondersteunen en dat het relatief eenvoudig is om de communicatie tussen de apparaten te regelen. Een nadeel is dat het protocol minder geschikt is voor lange afstanden of hoge snelheden vanwege de beperkte bandbreedte en het risico op signaalstoringen.

Voordelen en nadelen van ieder protocol:
Het UART-protocol is eenvoudig en heeft weinig overhead, maar ondersteunt geen meerdere apparaten zonder extra hardware of software. Het I2C-protocol is in staat om meerdere apparaten te ondersteunen en is relatief eenvoudig te regelen, maar heeft beperkte bandbreedte en is minder geschikt voor lange afstanden of hoge snelheden.

Verschil tussen BAUD en bitrate:
BAUD is een maatstaf voor de symbolen of signalen die in een gegevensstroom worden verzonden, terwijl bitrate verwijst naar de hoeveelheid gegevens die in een seconde worden verzonden. BAUD en bitrate worden vaak door elkaar gebruikt. In één BAUD kunnen er namelijk meerdere lagen bits zitten, waardoor de bitrate en BAUD verschillend worden.

Manchester encoding:
Manchester encoding is een coderingstechniek die wordt gebruikt om gegevens te verzenden via seriële communicatieprotocollen zoals UART en I2C. De techniek gebruikt zowel de stijgende als de dalende flank van een signaal om informatie over te dragen, wat zorgt voor een hogere betrouwbaarheid van de gegevensoverdracht. Manchester encoding wordt vaak gebruikt in toepassingen waarbij een hoge mate van betrouwbaarheid en nauwkeurigheid van de gegevensoverdracht vereist is.

Security aspecten:
Om de beveiliging van het protocol te waarborgen is een rollende sleutel handig voor gebruik, waarbij elk bericht wordt geleverd met een willekeurig controlenummer (d.m.v. XOR’en).

Beschrijving van security risico's, beschrijving van je eigen protocol, en hoe je protocol bescherming biedt:
Onze protocol blijft continue de ttyS0 bestand aflezen (kan ook via terminal met "cat /dev/ttyS0"). De Arduino verzendt namelijk een 'X' als er bijvoorbeeld op een knop is ingedrukt. Dit gaat via de seriële pinnen van zowel de Arduino als die van de Raspberry Pi. Hiervoor worden de RX en TX pinnen gebruikt en we hebben ingesteld dat die 'X' naar de ttyS0 worden geschreven. Bij deze protocol hebben we dan ook security toegevoegd door middel van rollende sleutels. De Raspberry Pi (C-daemon) en de Arduino hebben beide een lijst van rollende sleutels. Voordat de 'X' wordt verzonden, wordt deze nog eerst ge-XOR'd met een van de rollende sleutels en wordt deze array van sleutels met +1 index verhoogd. Het is belangrijk dat deze index synchroon blijft tussen de Arduino en Raspberry Pi, want anders zal de Raspberry Pi de data niet accepteren. De Raspberry Pi ontvangt namelijk geencrypte data en XOR'd deze met dezelfde sleutel en zal dan de-crypten tot 'X' en zal daarna de index met +1 verhogen. Dit principe biedt bescherming, omdat data-sniffers alleen geencrypte data kunnen opslaan. Hackers hebben er niks aan, zolang ze de lijst van de rollende sleutels niet in handen hebben!

Verschillende opties specifieke onderdelen van je eigen protocol:
De protocol opent de file descriptor met:
Int fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY);: open de file "dev/ttyS0", rechten: ReaD/WRite en NOControleTTY
fcntl(fd, F_SETFL, 0);: hiermee wordt de besturing van de poort ingesteld op standaardinstellingen.
tcgetattr(fd, &options);: haalt de huidige instellingen van de seriële poort op en slaat deze op in de variabele 'options'.
cfsetispeed(&options, B9600);: stelt de invoersnelheid (baudrate) van de seriële poort in op 9600 bits per seconde.
cfsetospeed(&options, B9600);: stelt de uitvoersnelheid (baudrate) van de seriële poort in op 9600 bits per seconde.
options.c_cflag |= (CLOCAL | CREAD);: stelt de poort in op het lezen van gegevens en het negeren van de modemstatuslijnen.
options.c_cflag |= CS8;: stelt het aantal data-bits in op 8.
options.c_cflag &= ~(tcflag_t)PARENB;: stelt de pariteitscontrole uit
options.c_cflag &= ~(tcflag_t)CSTOPB;: stelt het aantal stop-bits in op 1.
options.c_cflag &= ~(tcflag_t)CSIZE;: stelt het aantal data-bits in op 8
options.c_lflag &= ~(tcflag_t)(ICANON | ECHO | ECHOE | ISIG);: zet de canonieke modus, echo en signalen uit. De daemon hoeft niet te wachten op een invoer regel.
options.c_iflag &= ~(tcflag_t)(IXON | IXOFF | IXANY);: schakelt softwarematige handshaking uit
options.c_oflag &= ~(tcflag_t)OPOST;: zet de uitvoermodus naar "RAW".
tcsetattr(fd, TCSANOW, &options);: past de nieuwe seriële poortinstellingen toe.

*__EXTRA__: we hebben later **(tcflag_t)** geïmplementeerd: de gegevenstype is een unsigned integer dat wordt gebruikt om een combinatie van bit-flags op te slaan die

verschillende terminalinstellingen specificeren. We kregen namelijk waarschuwingen tijdens het compileren met *-Wall -Wextra -Wconversion*, toen tcflag_t **niet** er in stond.

Onderbouwing van de keuze voor de specifieke onderdelen:

We hebben die specifieke onderdelen nodig, want anders wil de protocol niet meewerken, zo is het onder andere belangrijk om de BAUD-rate van beide pins op 9600 te zetten, want de Arduino werkt ook op die BAUD-rate. Ook wordt de pariteitscontrole uitgeschakeld: dat houdt in dat de code niet controleert op fouten van overdracht van data, want dat kost teveel tijd en is in ons geval niet nodig. Daarnaast worden er ook controles uitgevoerd om ervoor te zorgen dat het proces netjes zal verlopen. In dit geval hebben we ook de softwarematige handshaking uitgezet, omdat het ten eerste het proces zal vertragen, omdat handshaking ook data weer terug zal koppelen aan de Arduino. Het probleem is ook dat de Arduino het handshaken niet echt ondersteunt, tenzij ik het implementeer. Als alternatief om alles secure te krijgen, hebben we gekozen om een rollende sleutel encryptie toe te voegen, wat een stuk makkelijker is. We hebben ook gekozen om “raw” data naar de Arduino te sturen als optie. Dit, omdat anders de Raspberry Pi extra bewerkingen zal toepassen op de data die verzonden wordt. We zullen ook weer de read() functie moeten toepassen in een oneindige loep om de gegevens te lezen van de /dev/ttyS0 bestand. De open(), read() en close() hebben we gelukkig al eerder moeten gebruiken in voorgaande opdrachten, dus dat proces moet lukken!

Voorbereiding van protocol onderzoek:

Om de seriële (RX/TX) poort (/dev/ttyS0) straks aan de gang te krijgen moeten we die configureren:

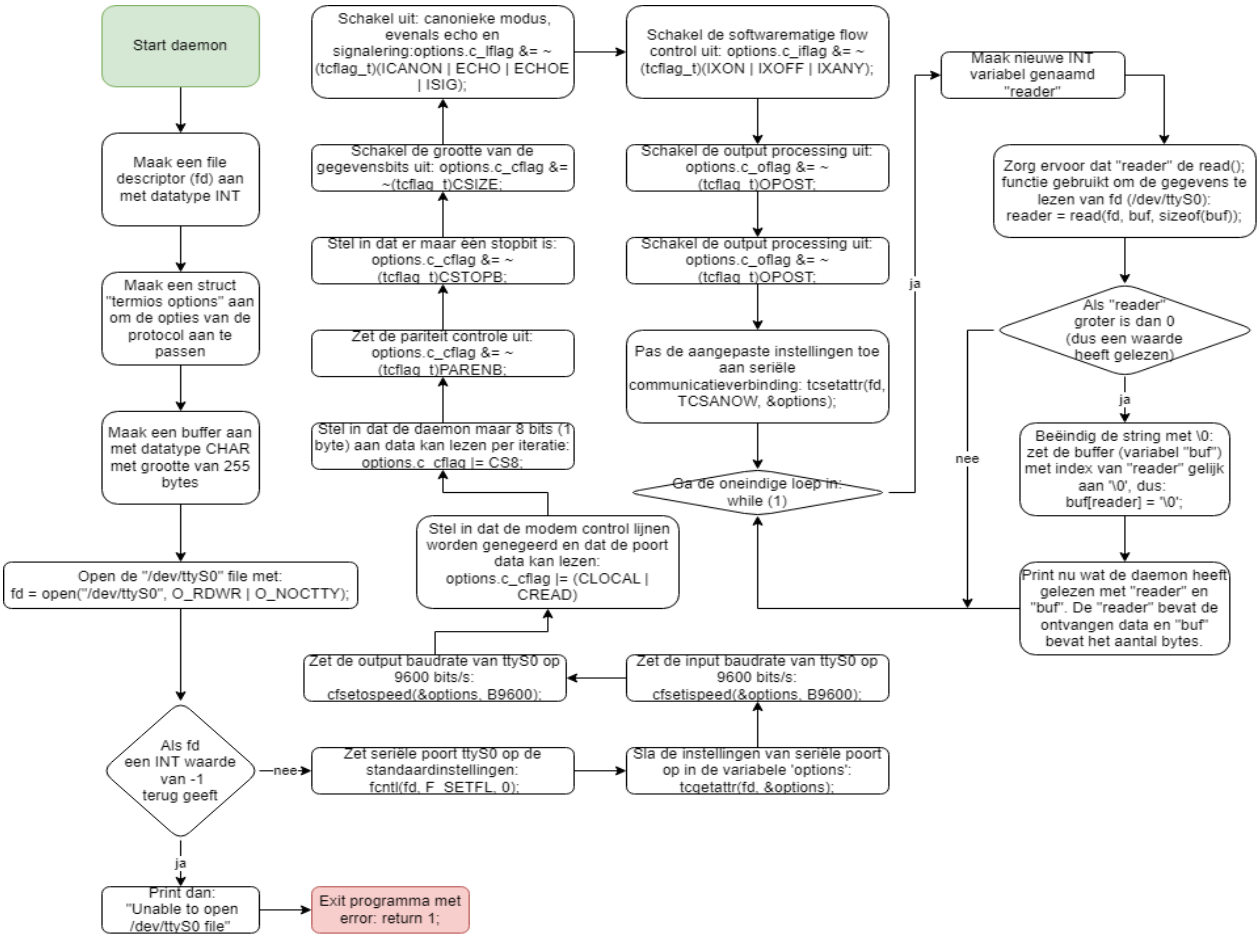
1. We moeten naar de Raspi-Config menu, en daar moet je de “seriële poort hardware” aan zetten, want dan wordt /dev/ttyS0 ingeschakeld.
2. Daarna moesten we /dev/ttyS0 configureren met commando: stty -F /dev/ttyS0 9600 cs8 -cstopb -parenb

In de C daemon configureren we dit later waarschijnlijk ook met behulp van de <termios.h> library. De afkorting van de <termios.h> library komt van: TERMinal Input/Output Structure, en is nodig om die asynchrone communicatie poorten in C te beheren.

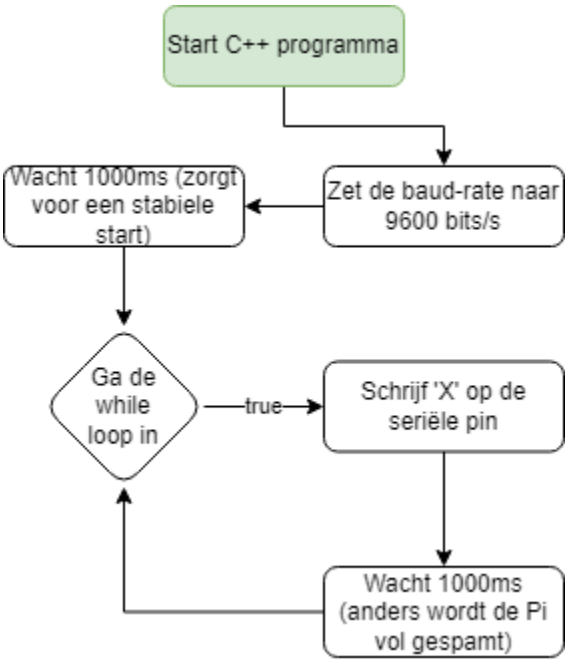
Erg essentiële bron voor later: https://man7.org/linux/man-pages/man3/termios.3.html

Opdracht 4 (2 uur) - Ontwerp code

Flowchart (of PSD) van de C daemon:



Flowchart (of PSD) van de C++ code op de Arduino:



Opdracht 5 (2 uur) – Requirements

Beschrijving functionele requirements m.b.v. de template:

Item naam	**Rol van deze item in het project**
Knop 1 (button 1)	Zorgt ervoor dat als de knop ingedrukt is, dat stoplicht 1 op groen gaat en dat stoplicht 2 op rood gaat.
Knop 2 (button 2)	Zorgt ervoor dat als de knop ingedrukt is, dat stoplicht 1 op rood gaat en dat stoplicht 2 op groen gaat.
Stoplicht 1 (traffic light 1)	Bevat een rode, gele, groene LEDs (TL1). Groen heeft een duur van 5 seconden, geel een duur van 3 seconden en rood (TL2) moet dan wel een totale duur hebben van 8 seconden. Als knop 1 wordt ingedrukt, zal de cyclus van stoplicht 1 zoals hierboven beschreven gewoon opnieuw uitgevoerd worden. Als knop 2 wordt ingedrukt, zal de cyclus van stoplicht 2 afgespeeld worden.
Stoplicht 2 (traffic light 2)	Bevat een rode, gele, groene LEDs (TL2). Groen heeft een duur van 5 seconden, geel een duur van 3 seconden en rood (TL1) moet dan wel een totale duur hebben van 8 seconden. Als knop 2 wordt ingedrukt, zal de cyclus van stoplicht 2 zoals hierboven beschreven gewoon opnieuw uitgevoerd worden. Als knop 1 wordt ingedrukt, zal de cyclus van stoplicht 1 afgespeeld worden.

Beschrijving van security eisen m.b.v. de template:

Beveiligingsrisico (security risk)	Verzachtende aanpak (mitigation approach)
Eavesdropping (Afluisteren): De aanvaller kan belangrijke (sensitieve) data aflezen via de seriële communicatie tussen de Raspberry Pi en de Arduino.	Om dit te voorkomen, kunnen we gebruik maken van een rollende sleutel. Hierdoor wordt data versleuteld en heeft de aanvaller er niks aan.
Replay attack (Her verzend-aanval): De aanvaller kan data aflezen via de seriële communicatie tussen de Raspberry Pi en de Arduino en vervolgens dezelfde data opnieuw verzenden, zonder dat het de bedoeling is van de Raspberry Pi of Arduino.	Om dit te voorkomen kunnen we gebruik maken van “timestamps”. Terwijl we de versleutelde data sturen, sturen we ook meteen de time stamp van dat moment (ook versleuteld uiteraard). Als die dan overeenkomt met de timestamp van het andere apparaat, dan gaat alles goed. Zo niet, dan wordt er een replay attack uitgevoerd (dat geen nut meer heeft).
Man-in-the-middle (MitM-aanvallen): De aanvaller kan tussen de communicatie van de Raspberry Pi en de Arduino onderscheppen en daarvandaan bepaalde data wijzigen en vervolgens doorsturen.	Om dit probleem op te lossen kunnen we ook gebruik maken van (rollende) sleutels, maar ook door gebruik te maken van digitale certificaten (SSL/TLS-encryptie).
Denial of Service (DoS): De aanvaller kan de seriële communicatielijn overspoelen (spam) met hoge snelheden met grote hoeveelheden aan data. Hierdoor zal de communicatielijn overbelast raken en niet meer kunnen normaal kunnen functioneren.	Om een DoS aanval te voorkomen, kunnen we gebruik maken van bepaalde tijd controles. We kunnen dan instellen wat de maximale aantal toegestane berichten per tijdseenheid is.

Opdracht 6 (9 uur) - Test en implementatie

Test item	Test descriptie (methodologie)	Resultaat
Stoplichten aangesloten op de Arduino	Eerst sluiten we de eerste stoplicht aan. De pinnen daarvoor zijn: groen1 = pin-4, geel1 = pin-3, rood1 = pin-2. Nu sluiten we de tweede stoplicht aan: groen2 = pin-7, geel2 = pin-6, rood2 = pin-5 Uiteraard moet tussen Vcc en alle LED's nog weerstanden van ~220 Ohm. In het C++ programma moet er een logische cyclus oneindig afspelen. Als stoplicht 2 op rood staat voor 8 seconden, dan moet stoplicht 1 op groen voor 5 seconden en daarna 3 seconden op geel. Vervolgens zal het C++ programma van stoplicht cyclus omdraaien, waardoor stoplicht 1 op rood gaat voor 8 seconden, stoplicht 2 op groen voor 5 seconden en ten slotte op geel voor 3 seconden. Dit zal zo oneindig doorgaan.	Alles werkt! Zie afbeelding 1 op pagina 83.
Knopjes aangesloten op de Arduino	Sluit knop-1 aan pin 8 van de Arduino, en sluit knop-2 aan pin 9 van de Arduino. Sluit ook Vcc aan. Je hebt dus Vcc naar knop en van knop naar pin.	Dit werkt correct! Zie referentie afbeelding (1) hierboven.

	Vanaf beide knoppen moet er ook een draad gaan naar een pull-up weerstand en vervolgens naar GND. Als knop 1 wordt ingedrukt moet stoplicht 1 op groen en stoplicht 2 op rood en vervolgens de cyclus uitvoeren. Als knop 2 wordt ingedrukt moet stoplicht 2 op groen springen en stoplicht 1 op rood en vervolgens cyclus uitvoeren.	
De verbinding tussen de Raspberry Pi en de Arduino	De Raspberry Pi moet met de 3.3V pin naar LV pin van de levelshifter en die gaat vanaf HV naar de 5V ingang van de Arduino. Ook gaat er vanaf de Raspberry Pi een GND draad naar de GND pin van de levelshifter die vervolgens weer naar de GND ingang gaat van de Arduino. Nu sluiten we de TX (GPIO14) en RX (GPIO15) pinnen aan voor de communicatie protocol. Sluit GPIO14 aan LV1 van de levelshifter en dan van HV1 naar RX pin van de Arduino. Sluit nu ook de RX (GPIO15) van de Raspberry Pi aan LV2 van de levelshifter en dan van HV2 naar de TX pin van de Arduino. Als het goed is ga je van TX naar RX en van RX naar TX.	Zie afbeelding 2 op pagina 83.

	Nu stellen we de zowel de Raspberry Pi als de Arduino in op Serial 9600 BOUD. Nu kun je vanaf de Arduino een ‘X’ sturen. De Raspberry Pi ontvangt dit via ttyS0.	
C daemon (socket server)	In dit C programma wordt er een socket server aangemaakt. Deze werkt op port 4567 en als host localhost. De socket server zit in een oneindige while loop, dus de socket server blijft continue checken op clients. Als er dan een client wil verbinden, dan stuurt de client eerst een request naar de C server en die bepaald of hij de verbinding aangaat of niet. Zodra er dan een verbinding is, dan kan de server bijvoorbeeld ontvangen data van de client weer printen in de terminal.	Ja het is meerde malen gelukt, zowel met de stoplicht met web interface als met een simpele client-server connectie. Zie de simpele client-server interactie op afbeelding 5 op pagina 84.
Fysieke knopjes (Raspberry Pi)	Eerst prikken we de twee knopjes ergens in een breadboard. Vervolgens sluiten we Vcc aan een van de pinnen. Voor knop 1 sluiten we dan GPIO pin-2, en knop 2 sluiten we aan GPIO pin-3. Als het goed is hebben we nu Vcc -> KNOP1 -> GPIO2 en Vcc -> KNOP2 -> GPIO3. Nu moeten we de GPIO pinnen 2 en 3 exporten en configureren voor ‘IN’.	Zie afbeelding 3 op pagina 83.

	Als een van de knopjes nu wordt ingedrukt, zal de GPIO state veranderen en in een C programma kunnen we dan die file openen en data aflezen/opslaan/verwerken.	
Website knopjes (Nginx)	Eerst maken we twee knopjes aan: “knop1” en “knop2” in HTML. Als een van de knopjes wordt ingedrukt dan zullen ze van kleur veranderen, afhankelijk van welke je indrukt. Dat zullen dan de stoplicht kleur-cyclussen worden. Dus als ik op knop1 druk, zal die op groen gaan, en knop2 op rood, enzovoorts. Beide knoppen hebben een pythonscript ready staan in de FastCGI folder van de Pi. Deze pythonscripts communiceren met de C socket server, vervolgens zal dan de stoplicht zijn werk doen.	Zie afbeelding 4 op pagina 84.

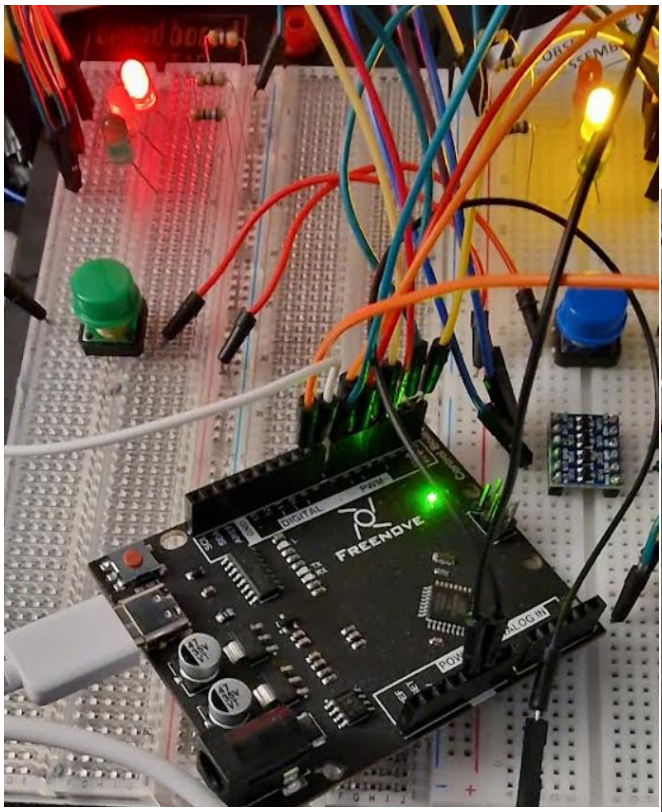


Figure 1: Arduino met stoplicht en knopjes aangesloten.

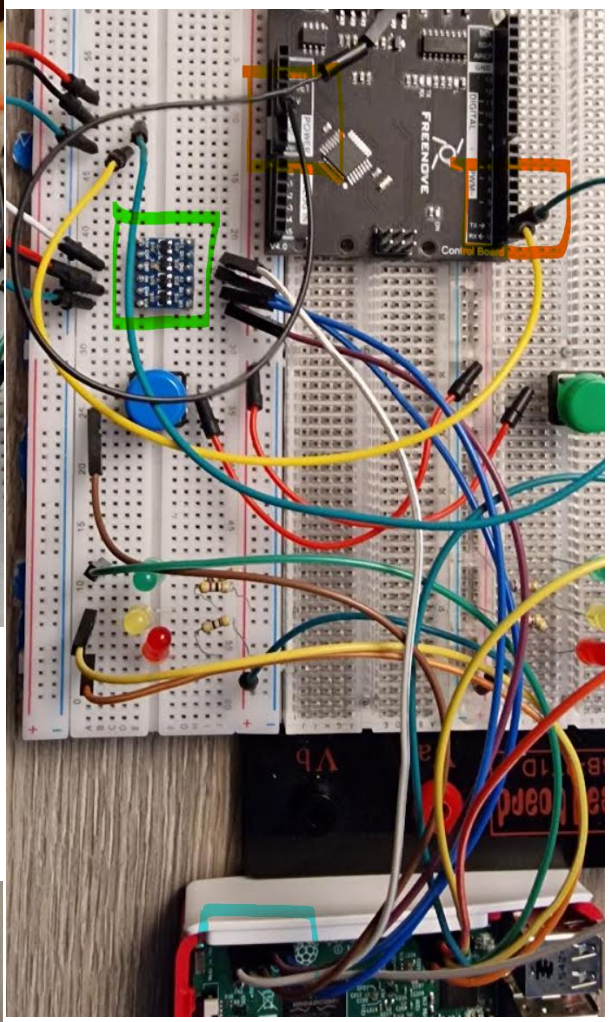


Figure 2: Arduino aangesloten aan de Raspberry Pi d.m.v. een levelshifter. Groen vakje is de levelshifter, Cyan vakje is de GPIO en Oranje vakjes zijn de digital-I/O en 5V, GND pinnen van de Arduino.

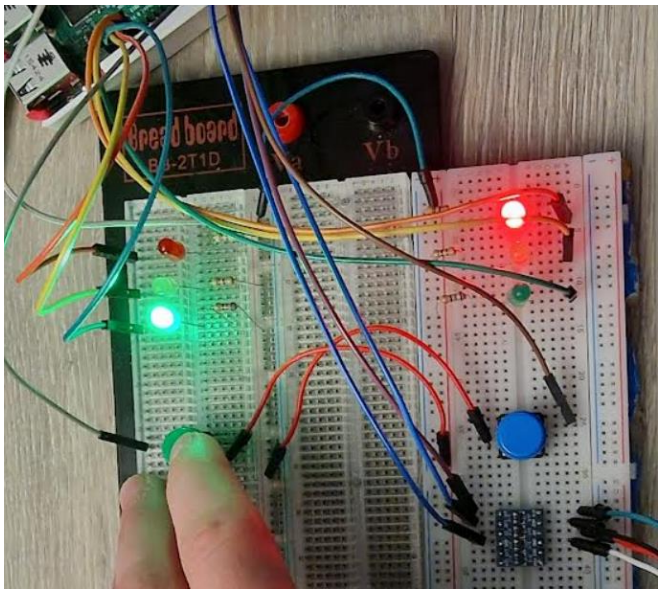


Figure 3: De fysieke knop_1 wordt ingedrukt en stoplicht_1 springt op groen!

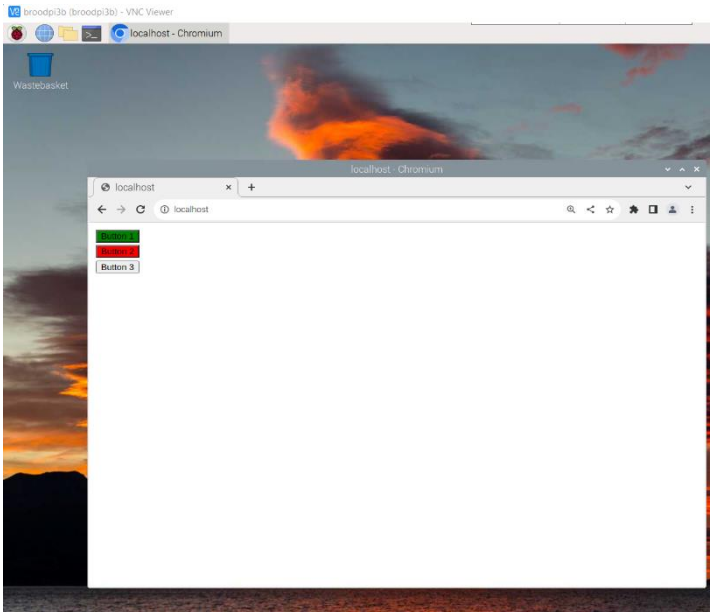


Figure 4: Nginx HTML web interface met twee web knopjes. P.S. de button_3 zat er in voor debugging, dus dat knopje is niet relevant!

```
semihpi@broodpi3b:~/Codon $ ./server
LED staat aan!Received 2 bytes: X
```

Figure 5a: Deze socket server ontvangt een "X" van de client, waardoor er een LED aan zal gaan!

```
semihpi@broodpi3b:~/Codon $ ./client localhost 4567
Please enter the message: X

semihpi@broodpi3b:~/Codon $
```

Figure 5b: Dit is de client die een connectie maakt met localhost als host en 4567 als poort. De client stuurt in dit geval een "X" naar de socket server.

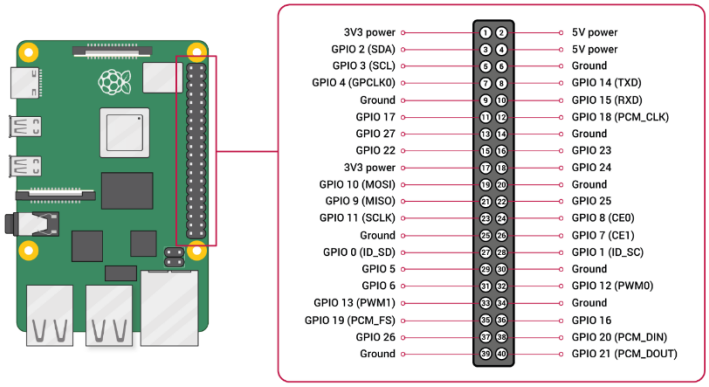
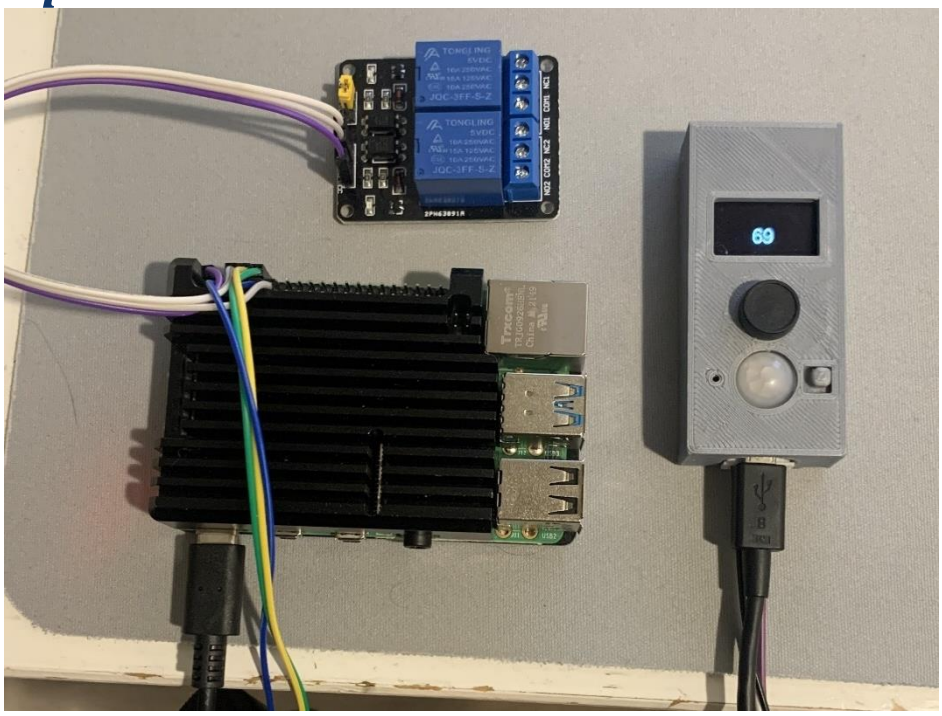
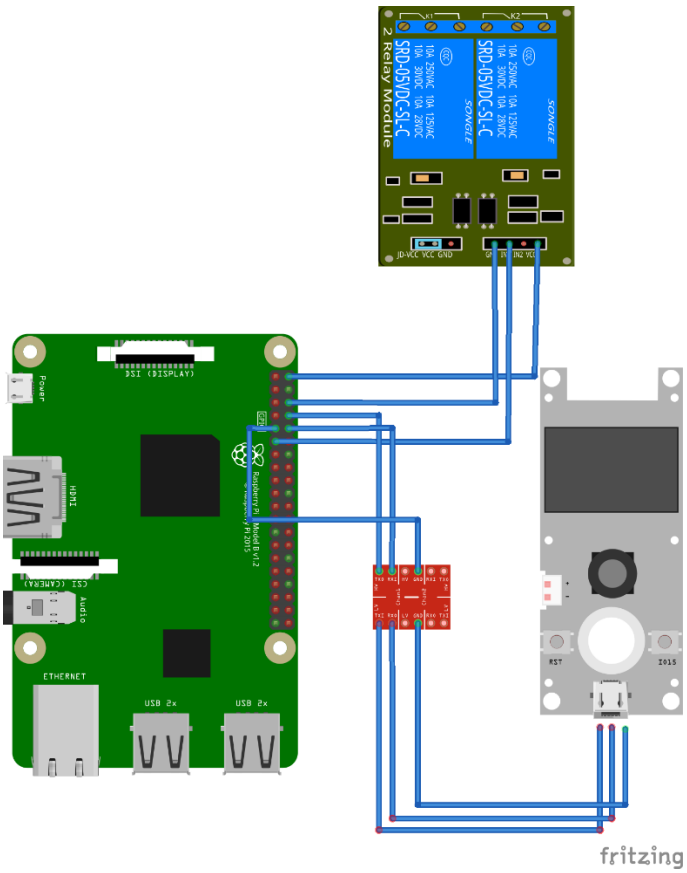


Figure 6: Handige afbeelding met alle General Purpose Input/Output (GPIO) aansluitingen.

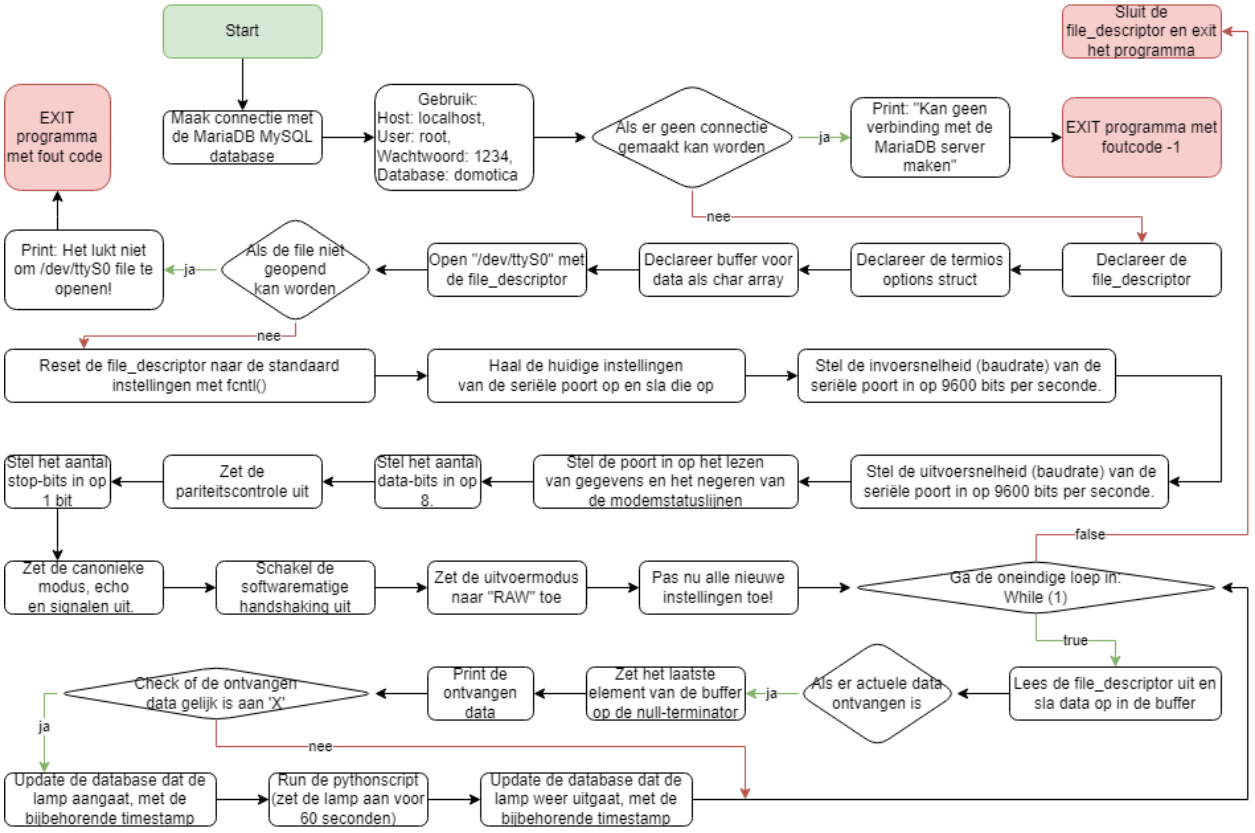
Het eindproduct: de slimme deurbel



Schema van de Raspberry Pi, ESP32 en 5V Relay:



PSD (flowchart) van de C daemon:



Arduino ESP32 Code:

```
#include <Arduino.h>
#include <WiFi.h>
#include <Wire.h>
#include "SSD1306.h"
#include "soc/soc.h"
#include "soc/rtc_cntl_reg.h"
#include "esp_camera.h"
#include "pinout.h"

SSD1306  display(0x3c, 21, 22);

// WiFi gegevens
const char* ssid = "thom";
const char* password = "12345678";

//raspi server gegevens
String serverName = "192.168.68.102";
String serverPath = "/upload.php";
const int serverPort = 80;
```

```
bool motionDetected = false; // Flag voor motion sensor.

WiFiClient client;

void setup() {

  //Display setup
  display.init();
  display.setTextAlignment(TEXT_ALIGN_CENTER);
  display.setFont(ArialMT_Plain_24);
  display.drawString(64, 22, "69");
  display.display();

  //pin setup
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  pinMode(AS312_PIN, INPUT);

  WRITE_PERI_REG(RTC_CNTL_BROWN_OUT_REG, 0);  //zet uit dat hij dingen doet bij
te weinig wattage
  Serial.begin(9600); //start seriele verbinging op 9600 baud

  //connect met WiFi
  WiFi.mode(WIFI_STA);
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(500);
  }
  Serial.println();
  Serial.print("ESP32-CAM IP Address: ");
  Serial.println(WiFi.localIP()); //print ip adress van esp32

  //camera instellingen
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
```

```
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sscb_sda = SIOD_GPIO_NUM;
  config.pin_sscb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 20000000;
  config.pixel_format = PIXFORMAT_JPEG;

  config.frame_size = FRAMESIZE_UXGA;
  config.jpeg_quality = 10;  //0-63 lower number means higher quality
  config.fb_count = 2;

  // camera initializer met de instellingen
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
  }

  //overige instelling voor na het maken van de foto.
  sensor_t *s = esp_camera_sensor_get();
  s->set_vflip(s, 1); //foto flippen
  s->set_hmirror(s, 1); //foto flippen (is ondersteboven?)
  s->set_brightness(s, 1);
  s->set_saturation(s, -2);


}
void loop() {
  //Button functie
  if (digitalRead(BUTTON_PIN) == LOW) {
    display.clear();
    display.drawString(64, 10, "ding dong!"); //ding dong op scherm
    display.display();
    sendPhoto(); //start sendPhoto functie
    display.clear();
    display.setTextAlignment(TEXT_ALIGN_CENTER);
    display.drawString(64, 10, "69"); //laat huisnummer weer zien op scherm
    display.display();
  }
```

```
  int pirState = digitalRead(AS312_PIN); // Lees PIR status
  if (pirState == HIGH && !motionDetected) { // Als PIR motion ziet en niet al
motion heeft gezien net
    motionDetected = true; // Maak motion flag = true

    Serial2.begin(9600, SERIAL_8N1, 21, 22); //start 2e seriele connectie op de
pins met baud 9600 op pin 21/22
    Serial2.write('X'); //stuur letter X over de nieuwe connectie

    Serial.println("Motion detected!");

  } else if (pirState == LOW && motionDetected) { // Als PIR geen motion meer
ziet maar net wel
    motionDetected = false; // Zet motion flag weer als false
    Serial.println("Motion stopped!");
  }
}


//functie voor versturen foto
String sendPhoto() {
  String getAll;
  String getBody;

  // Initialize de camera
  camera_fb_t * fb = NULL;
  esp_err_t res = ESP_OK;
  for(int i = 0; i < 3; i++){
    // maak een foto
    fb = esp_camera_fb_get();
    if(!fb) { // als foto niet gemaakt kan worden dan zeggen en restarten
      Serial.println("Camera capture failed");
      ESP.restart();
    }
    //maak 3 fotos (om te flushen) en bewaar de laatste
    if (i == 2) {
      break;
    }
    //Buffer legen voor volgende foto
    esp_camera_fb_return(fb);
  }

  Serial.println("Connecting to server: " + serverName);

  //Met server (raspi) verbinden
```

```
  if (client.connect(serverName.c_str(), serverPort)) {
    Serial.println("Connection successful!");
    String head = "--boundry\r\nContent-Disposition: form-data;
name=\"imageFile\"; filename=\"esp32-cam.jpg\"\r\nContent-Type:
image/jpeg\r\n\r\n";
    String tail = "\r\n--boundry--\r\n";

    uint32_t imageLen = fb->len;
    uint32_t extraLen = head.length() + tail.length();
    uint32_t totalLen = imageLen + extraLen;

    client.println("POST " + serverPath + " HTTP/1.1");
    client.println("Host: " + serverName);
    client.println("Content-Length: " + String(totalLen));
    client.println("Content-Type: multipart/form-data; boundary=boundry");
    client.println();
    client.print(head);

    uint8_t *fbBuf = fb->buf;
    size_t fbLen = fb->len;
    for (size_t n=0; n<fbLen; n=n+1024) {
      if (n+1024 < fbLen) {
        client.write(fbBuf, 1024);
        fbBuf += 1024;
      }
      else if (fbLen%1024>0) {
        size_t remainder = fbLen%1024;
        client.write(fbBuf, remainder);
      }
    }
    client.print(tail);

    esp_camera_fb_return(fb);

    int timoutTimer = 10000;
    long startTimer = millis();
    boolean state = false;

    while ((startTimer + timoutTimer) > millis()) {
      Serial.print(".");
      delay(100);
      while (client.available()) {
        char c = client.read();
        if (c == '\n') {
          if (getAll.length()==0) { state=true; }
```

```
          getAll = "";
        }
        else if (c != '\r') { getAll += String(c); }
        if (state==true) { getBody += String(c); }
        startTimer = millis();
      }
      if (getBody.length()>0) { break; }
    }
    Serial.println();
    client.stop();
    Serial.println(getBody);
  }
  else {
    getBody = "Connection to " + serverName +  " failed.";
    Serial.println(getBody);
  }
  return getBody;
}
```

Raspberry Pi codes:
C deamon:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <unistd.h>
#include <fcntl.h>
#include <termios.h>
#include <time.h>
#include <mariadb/mysql.h>

int main()
{
    MYSQL *connection = mysql_init(NULL); // Maak connectie met de MariaDB MySQL
database
    mysql_real_connect(connection, "localhost", "root", "1234", "domotica", 0,
NULL, 0);
    if (connection == NULL) // Als er geen connectie gemaakt kan worden
    {
        printf("Kan geen verbinding met de MariaDB server maken\n");
        exit(-1);
    }

    int fd;                 // De file_descriptor
    struct termios options; // Termios opties
    char buf[255];          // Char buffer array voor data
```

```
    fd = open("/dev/ttyS0", O_RDWR | O_NOCTTY); // Open de /dev/ttyS0 file met
read/write en no control tty rechten
    if (fd == -1)                               // Als de file_descriptor niet
geopend kan worden, dus een fout heeft
    {
        perror("open_port: Unable to open /dev/ttyS0 - "); // Print dan een error
message
        return 1;                                          // Exit de C daemon
met foutcode 1
    }

    fcntl(fd, F_SETFL, 0); // Reset de file_descriptor naar standaard
instellingen

    tcgetattr(fd, &options);             // Haal de huidige instellingen van de
seriële poort op en sla die op
    cfsetispeed(&options, B9600);        // Stel de invoersnelheid (baudrate) van
de seriële poort in op 9600 bits per seconde.
    cfsetospeed(&options, B9600);        // Stel de uitvoersnelheid (baudrate)
van de seriële poort in op 9600 bits per seconde.
    options.c_cflag |= (CLOCAL | CREAD); // Stel de poort in op het lezen van
gegevens en het negeren van de modemstatuslijnen
    options.c_cflag |= CS8;              // Stel het aantal data-bits in op 8.

    options.c_cflag &= ~(tcflag_t)PARENB;                          // Zet de
pariteitscontrole uit
    options.c_cflag &= ~(tcflag_t)CSTOPB;                          // Stel het
aantal stop-bits in op 1 bit
    options.c_cflag &= ~(tcflag_t)CSIZE;                           // Reset de bit
size en stel het aantal data-bits in op 8.
    options.c_lflag &= ~(tcflag_t)(ICANON | ECHO | ECHOE | ISIG); // Zet de
canonieke modus, echo en signalen uit.
    options.c_iflag &= ~(tcflag_t)(IXON | IXOFF | IXANY);         // Schakel de
softwarematige handshaking uit
    options.c_oflag &= ~(tcflag_t)OPOST;                           // Zet de
uitvoermodus naar "RAW" toe

    tcsetattr(fd, TCSANOW, &options); // Pas nu alle nieuwe seriele poort
instellingen toe

    while (1) // Ga de oneindige loep in:
    {
        int n = read(fd, buf, sizeof(buf)); // Lees de file_descriptor uit en sla
data op in de buffer
```

```
        if (n > 0)                          // Als er actuele data ontvangen is
        {
            buf[n] = '\0';                              // Zet het laatste element
van de buffer op de null-terminator
            printf("Received %d bytes: %s\n", n, buf); // Print de ontvangen data
            if (strcmp(buf, "X") == 0)                  // Check of de ontvangen
data gelijk is aan 'X'
            {
                printf("Received X!\n"); // Print nogmaals dat 'X' ontvangen is
                /* Update de database dat de lamp aangaat, met de bijbehorende
timestamp */
                int error = mysql_query(connection,
                                        "UPDATE Light SET timestamp =
CURRENT_TIMESTAMP(), state = '1'");
                if (error)
                    printf("%s\n", mysql_error(connection)); // Als de query niet
klopt, print dan error

                system("/home/thomvdv/relay-test.py"); // Run de pythonscript
(die doet de lamp aan voor 60 seconden)

                /* Update de database dat de lamp weer uitgaat, met de
bijbehorende timestamp */
                char query[110];
                sprintf(query, "UPDATE Light SET state = 0, timestamp = (SELECT
timestamp FROM Light WHERE lightid = 1) WHERE lightid = 1");
                int error1 = mysql_query(connection, query);
                if (error1)
                    printf("%s\n", mysql_error(connection)); // Als de query niet
klopt, print dan error

                mysql_query(connection, "COMMIT"); // Verzend de SQL query
            }
        }
    }
    close(fd); // Sluit de file_descriptor
    return 0;  // Exit de daemon
}
```

Python relay script:

```
#!/usr/bin/env python

import time #Importeer time voor de time.sleep functie
import RPi.GPIO as GPIO #Importeer library voor de GPIO pins
```

```
GPIO.setmode(GPIO.BCM) #setmode als BCm om nummers te geven

GPIO.setup(17, GPIO.OUT) #set pin 17 als output
GPIO.output(17, GPIO.LOW) #set pin 17 uit/laag

time.sleep(10) #wacht 10 seconden (60 minuten in final product)

GPIO.output(17, GPIO.HIGH) #set pin 17 weer aan
GPIO.cleanup() #Zet alle pins weer terug naar standaard stand, einde script
```

PHP file upload/saver:

```
<?php
// Mapje is uploads, datum in bepaalt formaat.
$target_dir = "uploads/";
$datum = mktime(date('H')+0, date('i'), date('s'), date('m'), date('d'),
date('y'));
// Datum en tijd word de naam zo krijg je geen duplicates
$target_file = $target_dir . date('Y.m.d_H:i:s_', $datum) . "esp32-cam.jpg";
$uploadOk = 1;
$imageFileType = strtolower(pathinfo($target_file,PATHINFO_EXTENSION));

// Kijk of file er al is
if (file_exists($target_file)) {
  echo "Sorry, file already exists.";
  $uploadOk = 0;
}

// kijkt of er ergens een foutmelding is gekomen zoja dan gaat het niet verder
if ($uploadOk == 0) {
  echo "Sorry, your file was not uploaded.";

// Geen foutmelding dus bestand vanuit tijdelijke opslaglocatie naar de map
}
else {
  if (move_uploaded_file($_FILES["imageFile"]["tmp_name"], $target_file)) {
    echo "The file ". basename( $_FILES["imageFile"]["name"]). " has been
uploaded.";
  }
  else {
    echo "Sorry, there was an error uploading your file.";
  }
}
?>
```

PHP Gallery and light:

```
<!DOCTYPE html>
<html>
<head>
  <title>ESP32-CAM Photo Gallery</title>
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <style>
    .flex-container {
      display: flex;
      flex-wrap: wrap;
    }
    .flex-container > div {
      text-align: center;
      margin: 10px;
    }


  </style>
</head><body>
<h2>ESP32-CAM Photo Gallery</h2>

<?php
// DB gegevens
$servername = "localhost";
$username = "root";
$password = "1234";
$dbname = "domotica";

// maak connectie DB
$conn = new mysqli($servername, $username, $password, $dbname);

// Check connectie
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}

// select uit Light tabel om status te kunnen laten zien van de relay
$sql = "SELECT timestamp, state FROM Light";
$result = $conn->query($sql);

// uit het resultaat van de query haal je de timestamp en state, als state 1 is
dan laat ie on zien.
$row = $result->fetch_assoc();
echo "<br> Last power on: " . $row["timestamp"] . " - Status: " . ($row["state"]
== 1 ? "on" : "off") . "<br>";
```

```php
$conn->close();
?>


<?php
  // Haal alle files uit mapje uploads en echo ze met naam+afbeelding
  // Kijkt of het een bestaand mapje is, sorteert bestanden nieuw naar oud, maakt voor elke file een paragraph.
  $dir = 'uploads/';
  if (is_dir($dir)){
    echo '<div class="flex-container">';
    $files = scandir($dir);
    rsort($files);
    foreach ($files as $file) {
      if ($file != '.' && $file != '..') {?>
        <div>
          <p><?php echo $file; ?></p>
          <a href="<?php echo $dir . $file; ?>">
            <img src="<?php echo $dir . $file; ?>" style="width: 350px;" alt="" title=""/>
          </a>
       </div>
<?php
      }
    }
  }
?>
  </div>
</body>
</html>
```