```
    if (gpioInitialise() < 0) {
        printf("Gpio initialisation failed\n");
        return 1;
    }

    if (gpioRead(15) == 0)
        gpioWrite(15,1);

    gpioSetMode(12, PI_INPUT);
    gpioSetPullUpDown(12, PI_PUD_UP);
    gpioSetMode(14, PI_INPUT);
    gpioSetPullUpDown(14, PI_PUD_UP);
    gpioSetMode(15, PI_OUTPUT);
    gpioSetMode(20, PI_OUTPUT);
    gpioSetMode(21, PI_OUTPUT);

    printf("\tTraffic light is operational.\n\n");

    while(1) {
        o = gpioRead(14);
        q = gpioRead(12);

        if (o == 0)
            start_traffic_light();

        if (q == 0)
            shutdown_traffic_light();
    }
}
```

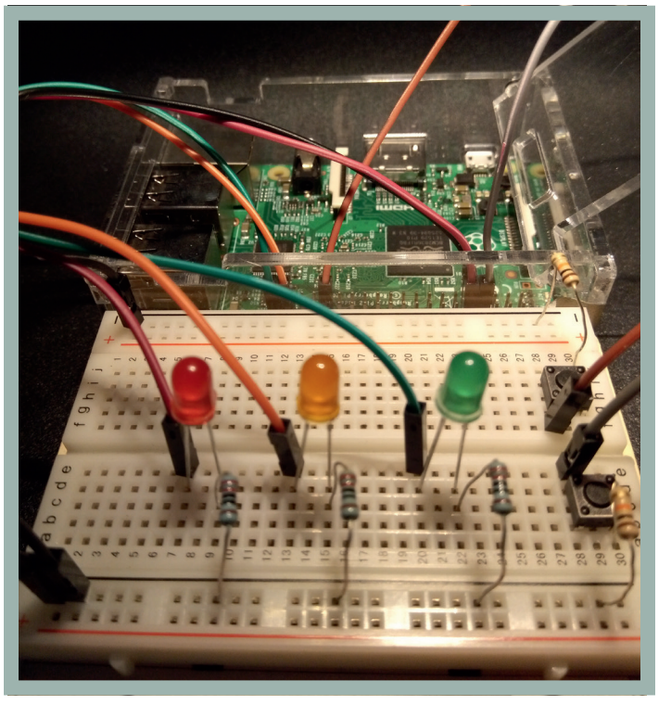
De eerste drie regels van het programma zijn de benodigde bibliotheken om het programma te laten werken.

In `Main` worden een aantal zaken geïnitialiseerd / gedefinieerd, zoals `o`, `q` en `int`. Daarna wordt er met een `if` statement gecontroleerd of dat de bibliotheek PiGPIO wel geïnitialiseerd kan worden. Zo niet, dan zal het programma stoppen. Nu zeker is dat de bibliotheek te gebruiken is, gaan de GPIO pinnen geconfigureerd worden.

Het `if` statement op regel 37 zorgt ervoor dat bij het starten van het programma het stoplicht op rood staat. `GpioRead` kijkt of dat pin 15 uit staat. Indien dat het geval is, dan wordt deze aangezet met `gpioWrite(pinnummer,1)`. In deze functie betekent 1 aan en 0 uit.

Op de pinnen 12 en 14 zitten knoppen aangesloten. Deze worden geïnitialiseerd door middel van de regels:

```
    gpioSetMode(12, PI_INPUT);
    gpioSetPullUpDown(12, PI_PUD_UP);
    gpioSetMode(14, PI_INPUT);
    gpioSetPullUpDown(14, PI_PUD_UP);
```



`gpioSetMode` zet de specifieke mode voor pin 12 en pin 14 naar input, zodat de knoppen gebruikt kunnen worden. `gpioSetPullUpDown` zet voor beide knoppen “hoog” (1) door `PI_PUD_UP`.

Op pin 15, 20 en 21 zijn de led’s gekoppeld. Deze geven alleen maar licht en `gpioSetMode PI_OUTPUT` is voldoende om ze op de correcte manier te laten werken. Je zet de led aan of uit door `gpioWrite`.

Na de initialisaties is er een `printf` statement, zodat je ook op het scherm feedback hebt, dat er daadwerkelijk op een knop gedrukt kan worden.

In de `while` wordt er continue gekeken of er op een van de knoppen gedrukt wordt, die zijn gekoppeld aan pin 12 of 14.

Bij een druk op de knop, die aangesloten is op pin 14, zal de functie `start_traffic_light` worden aangesproken. Deze functie is in het begin voorzien van een `printf` statement, zodat je weet dat de druk op de knop is geregistreerd.

`time_sleep` is een functie, die is beschreven in de pigpio.c.

`time_sleep` zorgt voor een korte pauze, voordat de rode led (pin 15) zal worden uitgeschakeld en groen (pin 21) zal worden aangezet. Na weer een korte pauze te hebben ingelast, zal groen (pin 21) worden uitgeschakeld en oranje (pin 20) worden aangezet. Dit wordt nogmaals herhaald van oranje naar rood.

De code van `time_sleep` is te achterhalen door pigpio.c te openen en te zoeken naar `void time_sleep`, zodat je weet wat de achterliggende techniek is achter deze functie. Dit geldt voor alle functies en is een mooi voorbeeld van open source.

Zoals je duidelijk ziet in `start_traffic_light` is dat 0 ervoor zorgt dat iets uit gaat en dat 1 zorgt voor een inschakeling.

De functie `shutdown_traffic_light` is er voor om het programma op een nette manier af te sluiten. Bij het afsluiten van het programma door CTRL+C zal dit niet gebeuren. Als je programma niet goed werkt, is CTRL+C zeker nog te gebruiken. De functie `shutdown_traffic_light` zorgt ervoor dat de rode led uit wordt geschakeld en `gpioTerminate` regelt dat de gebruikte DMA-kanalen worden gereset, het geheugen weer vrij wordt gegeven en dat alle actieve threads beëindigd worden.

COMPILEREN

Het compileren van het programma vergt wat meer dan alleen het commando `gcc` aanroepen en je C programma op te geven. Lukt het niet om gcc te gebruiken, dan is deze zeer waarschijnlijk niet geïnstalleerd. Controleer als eerste of dat dit daadwerkelijk het geval is door het volgende commando uit te voeren op de CLI:

```
gcc --version
```

Krijg je geen versie terug? Dan is gcc daadwerkelijk niet geïnstalleerd. Installeer gcc met het commando:

```
sudo apt-get install gcc
```

Compileer het programma als volgt:

```
gcc -Wall -pthread -o prog.com prog.c -lpigpio -lrt
```

`-Wall` toont alle waarschuwingen bij het compileren en is niet per se noodzakelijk. `-pthread` voegt ondersteuning toe voor multithreading. De `-o` met de naam daarachter is de naam van de executable, die je uiteindelijk gaat uitvoeren.

`-lpigpio` en `-lrt` starten beide met `-l` en deze optie zorgt voor het linken van de libraries `pigpio` en `rt`. De rt -ibliotheek (librt) handelt verschillende POSIX.1b interfaces af.

`Gcc` heeft een zeer uitgebreide manpage. Voor meer achtergrondinformatie over de gebruikte opties of andere opties, die je wilt onderzoeken, raad ik aan om de manpage te openen (`man gcc`).

Als het compileren zonder problemen is gelukt, voer je het gecompileerde bestand uit:

```
sudo ./prog.com
```

Het programma uitvoeren met `sudo` is noodzakelijk vanwege het gebruik van de GPIO pinnen.

TOT SLOT

Python is voor prototyping waarschijnlijk nog wel steeds de beste optie, maar hopelijk heeft het artikel je interesse gewekt om met C aan de slag te gaan op de Raspberry Pi en het aansturen van General Purpose Input/Output (GPIO). De bibliotheek PiGPIO heeft ook zeker veel meer in zich, zoals bijvoorbeeld een daemon optie. Leuk dus om thuis verder mee te spelen!

Op de website van de Raspberry Pi Foundation is. Er is alleen een veel makkelijkere manier om het overzicht te krijgen van je eigen specifieke Raspberry Pi zonder gebruik te maken van het internet.

32-34 PIGPI.indd 34 21-08-19 11:26