DON BOSCO HAACHT
BRON VAN TALENT

# Domotica Systeem

Mentor: Caroline Wille

Gerealiseerd door Ian Blockmans

Derde graad TSO – Elektriciteit-elektronica

Schooljaar 2017-2018

DON BOSCO HAACHT
BRON VAN TALENT

# Domotica Systeem

Mentor: Caroline Wille

Gerealiseerd door Ian Blockmans

Derde graad TSO – Elektriciteit-elektronica

Schooljaar 2017-2018

# Voorwoord

In het 6de middelbaar moeten alle leerlingen een eindwerk maken. Voor mij was het moeilijk om te kiezen wat ik wauw maken. Uiteindelijk heb ik gekozen om een domotica systeem te maken omdat ik vind dat domotica nog veel te weinig wordt toegepast. Ik had als doel dit zo goedkoop mogelijk te doen omdat ik ook vind dat domotica systemen veel te duur zijn.

Ik heb dit natuurlijk niet alleen kunnen realiseren dus ik zou de volgende mensen willen bedanken:

- Mijn mentor mevrouw C. Wille voor de hulp en de tijd die ze heeft besteed aan dit eindwerk.
- Mijn vader (Gerrit Blockmans) voor het helpen met mijn maquette.
- Mijn moeder (Carine Cloetens) voor financiële steun.

# Inhoud

# 1 Omschrijving

## 1.1 Concept

Mijn GIP is een domoticasysteem met de RaspberryPi. Domotica is heel uitgebreid. Ik kan niet alles maken dat wordt beschouwd als domotica maar ik ga deze onderdelen uitwerken:

- Verlichting
- Verwarming
- Beveiliging
- Rolluiken

## 1.2 Concreet

### a) Verlichting

De verlichting gaat in één kamer van kleur veranderd worden (RGB) en gedimd worden door het touchscreen en de site*. In de andere ruimtes zal de verlichting kunnen geschakeld worden door het touchscreen en de site en een fysieke knop.

### b) Verwarming

Kan geregeld worden door het touchscreen en de site. Eén kamer zal geïsoleerd worden en de temperatuur kan ingesteld worden op het touchscreen en de site.

### c)Alarmsysteem

Een alarm kan in of uit geschakeld worden door een persoon door middel van het touchscreen en de site. Als het afgaat zal er een visuele (verlichting knippert) en auditieve (geluid door speaker) melding komen. Er zal ook op de site kunnen gezien worden als er een alarm is afgegaan. Het alarm kan afgaan als het deurcontact wordt verbroken, de bewegingssensor beweging opvangt, de camerabeweging opvangt. Op de site zal ook via een camera gezien kunnen worden wat er in het huis gebeurt. Als het alarm af gaat, gaat de camera een bepaalt aantal seconde opslaan aan video.

### d)Rolluiken

De rolluiken kunnen bestuurd worden door middel van een knop en het touchscreen en de site.

### e) Bewegingsdetectie

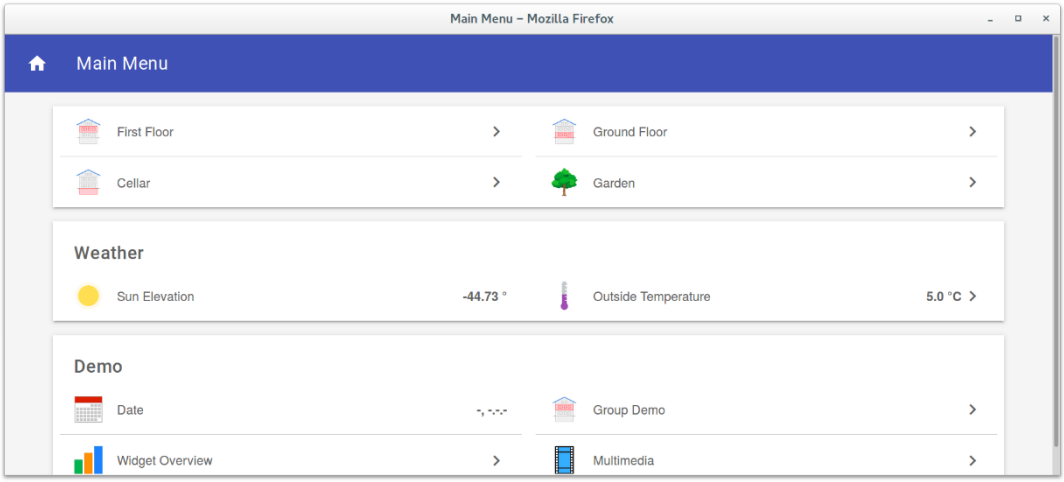
Een PIR-sensor en een camera gaan beweging detecteren.

## 1.3 Camera

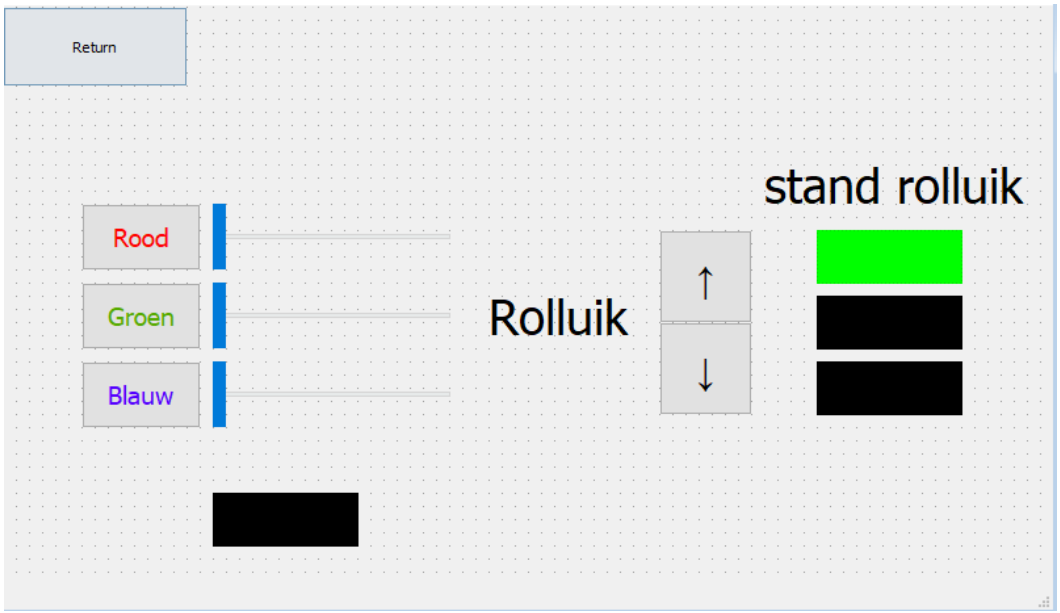
De Raspberry Pi gaat video van een webcam op de site tonen.

## 1.4 Site/Touchscreen

Voor de site gebruik ik software (OpenHAB).




Voor het touchscreen ga ik een GUI maken in Python met de PyQt library.



## 1.5 Maquette

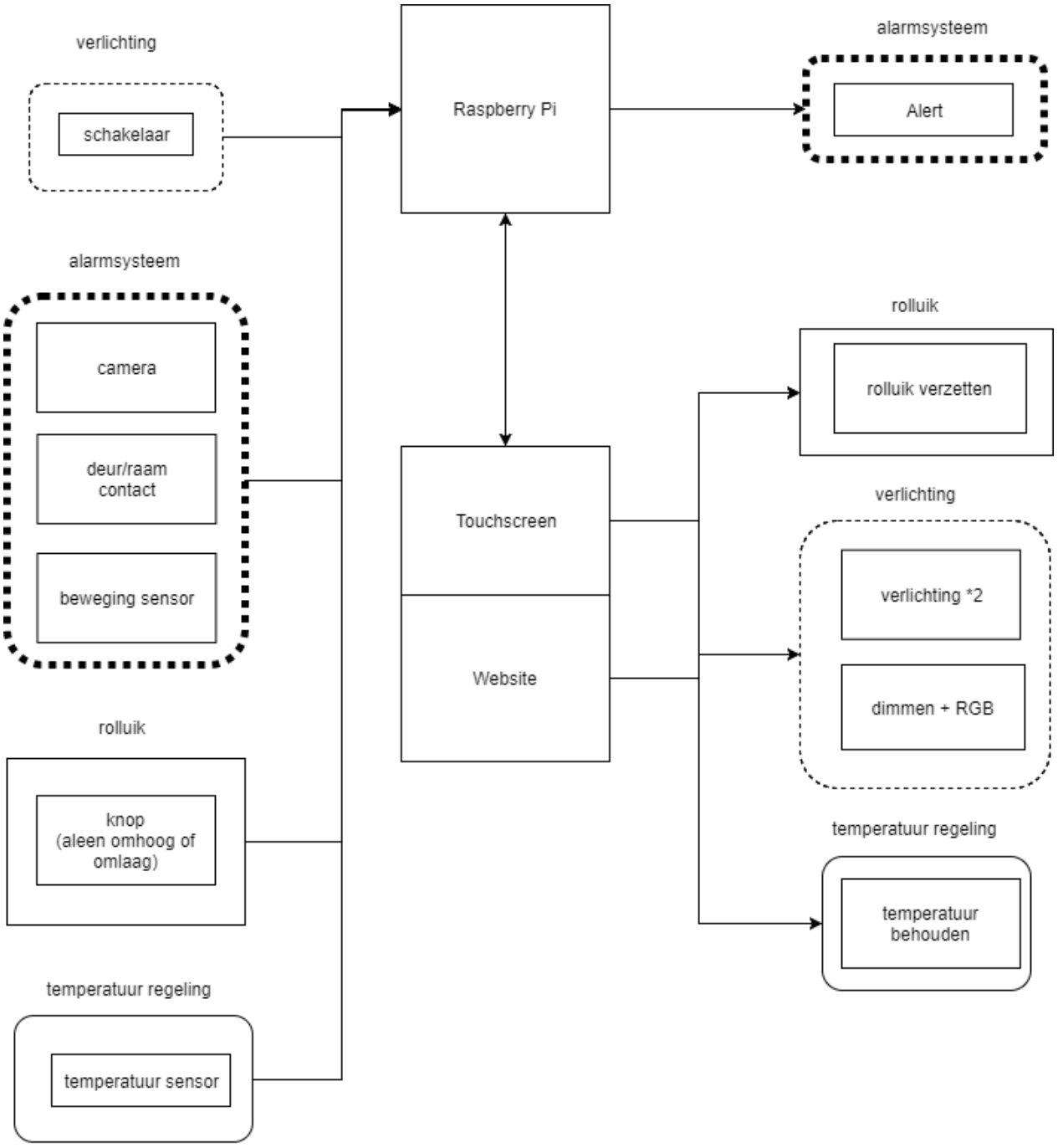
Ik ga een maquette bouwen (zie onderstaande tekening) die de opgesomde functionaliteiten zal demonstreren.

# 2 Planning

| BEGIN-DATUM | Opdracht | EIND-DATUM | AAN-GEPASTE EINDDATUM |
|---|---|---|---|
| Zomervakantie | | | |
| | • **Mechanisch opstelling gemaakt**<br>• **Opzoek werk (OpenHAB)**<br>• **Leren werken met RaspberryPi** | 31-08 | |
| September | | | |
| | **Opdrachtomschrijving eerste versie afwerken** | 14-09 | |
| | • **Verbetering blokschema.**<br>• **Beginnen aan schema's.**<br>• **Bestellen van meeste onderdelen.** | 27-09 | |
| | • **Verbeterde versie omschrijving afwerken.**<br>• **Planning afwerken.** | 1-10 | |
| Oktober | | | |
| | • **RGB dimmen werkt principieel.**<br>• **Schrijf software voor verlichting.** | 8-10 | |
| | • **Scriptie inhoudstafel maken**<br>• **Scriptie stuk verlichting schrijven** | 15-10 | |
| | • **Schema's maken** | 22-10 | |
| November | | | |
| | **Solderen en aansluiten van**<br>• **Led strips**<br>• **Rolluik**<br>• **Verwarming**<br>• **Bewegingssensor**<br>• **Temperatuur sensor**<br>**Software schrijven voor meeste onderdelen** | 12-11 | |
| | **Achterstand inhalen** | 23-11 | |

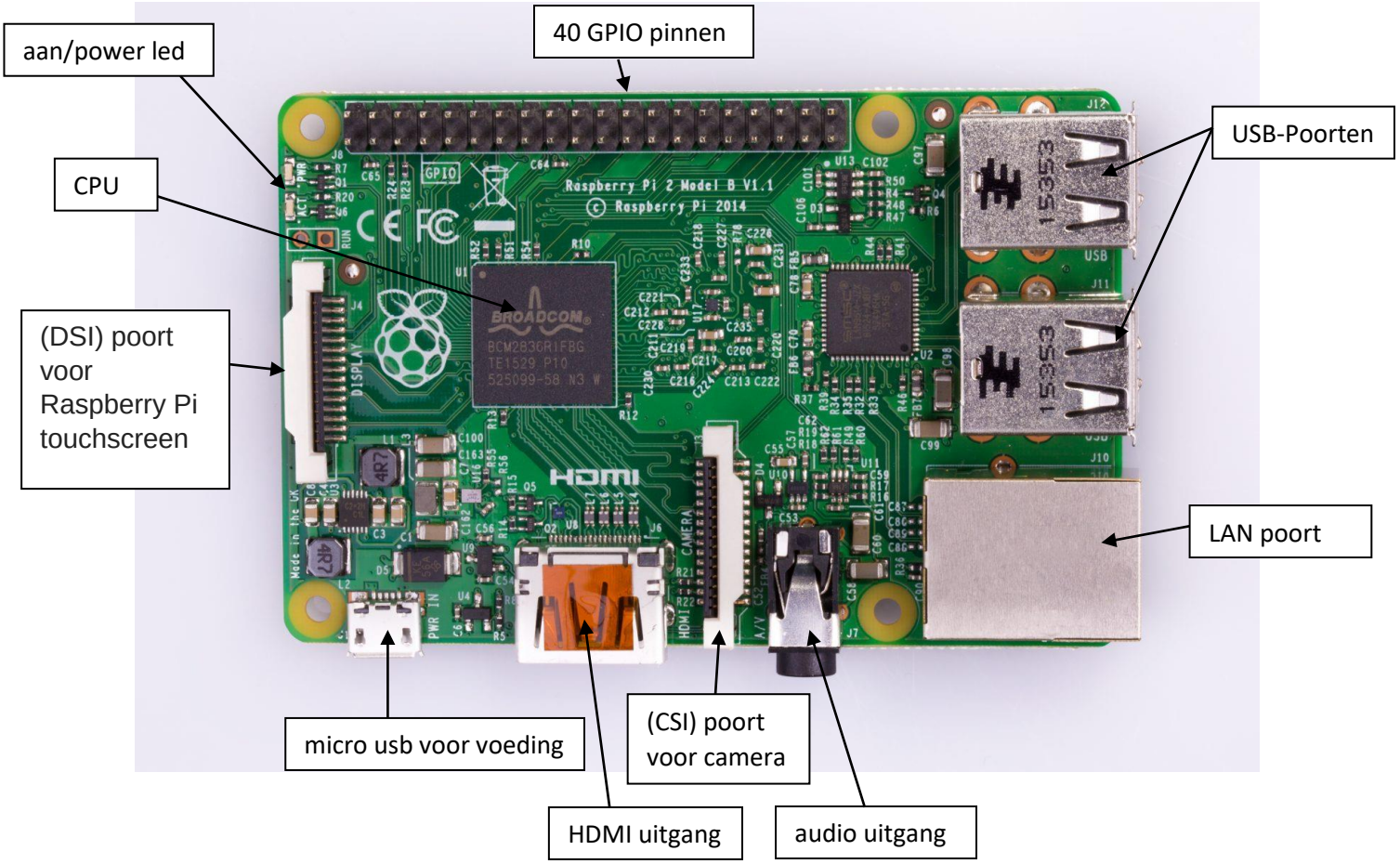
| | | | |
|---|---|---|---|
| December | | | |
| | **Kleine problemen oplossen** | 30-12 | |
| Januari | | | |
| | **Touchscreen verder uitwerken**<br>**Mechanische opstelling afwerken** | 14-01 | |
| | **Servo installeren en doen werken** | 28-01 | |
| Februari | | | |
| | **Scriptie verder schrijven**<br>**Site instellen** | 31-02 | |
| Maart | | | |
| | **Printplaat maken** | 31-03 | |
| April | | | |
| | **Scriptie afwerken en alles definitief monteren.** | 29-04 | |

# 3 Blokschema



# 4 Hardware

## 4.1 Raspberry PI



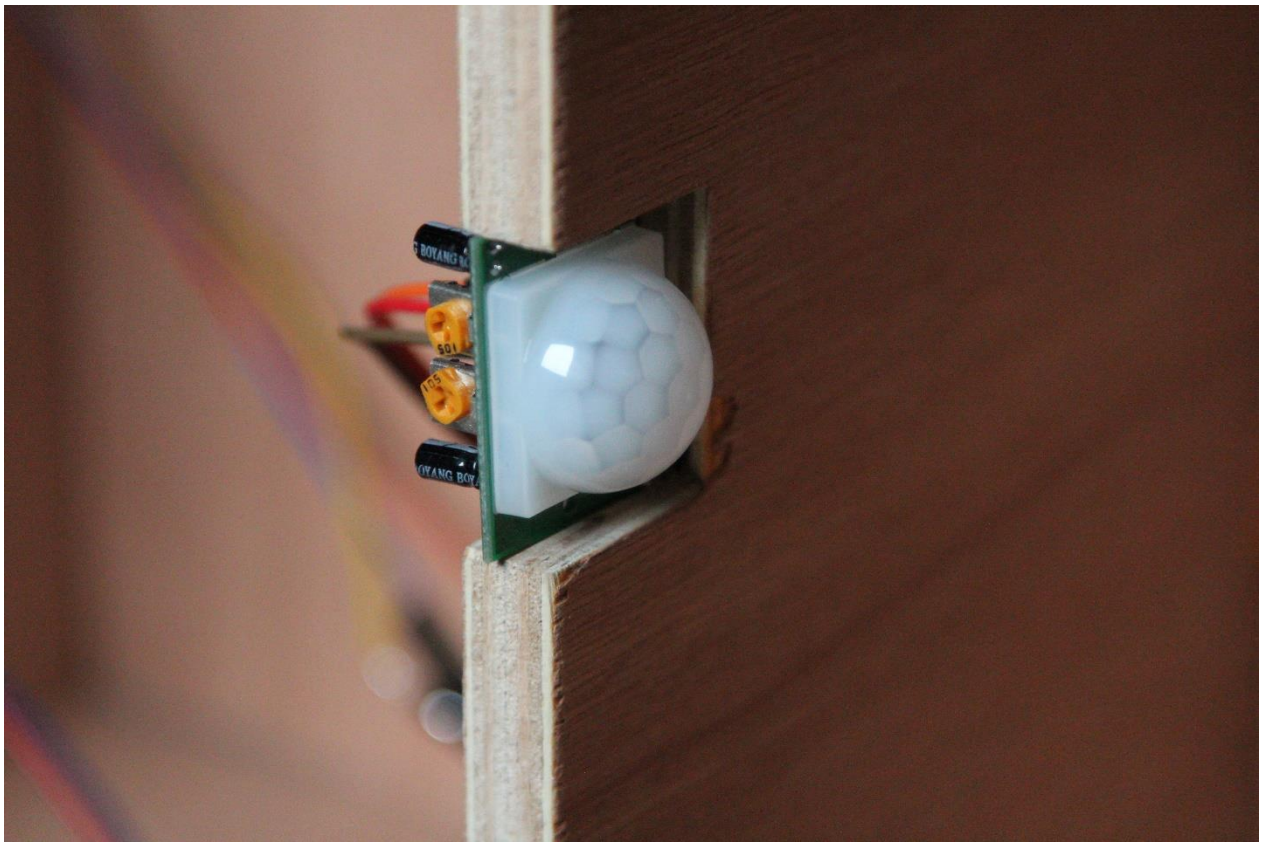

| | |
|---|---|
| CPU | Quad Core 1.2GHz Broadcom BCM2837 64bit CPU SoC ARMv8 architectuur |
| RAM | 1GB |
| Netwerk | BCM43438 wireless LAN and Bluetooth |
| GPIO | 40 pinnen |
| USB | 4 USB poorten |

Van de RPi gebruik ik de GPIO pinnen. Dit zijn programmeerbare I/O pinnen. Deze pinnen werken op een spanning van 3.3V. De max. uitgang stroom is max 50mA. Deze pinnen zijn hierdoor niet geschikt om een vermogen te sturen. De 5V pin is rechtstreeks verbonden met je bron. Ik gebruik ook de DSI-poort voor het touchscreen. Het touchscreen wordt nog apart gevoed door de 5V pin van de RPi.

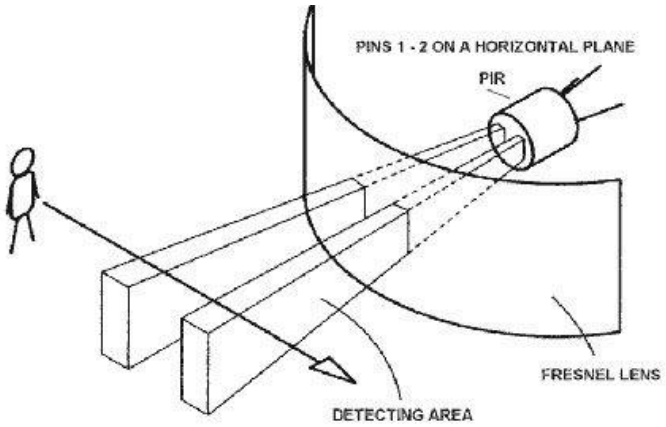
## 4.2 Bewegingsdetectie

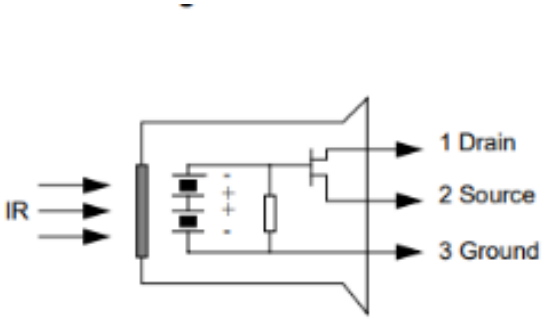
### a) Infraroodsensor

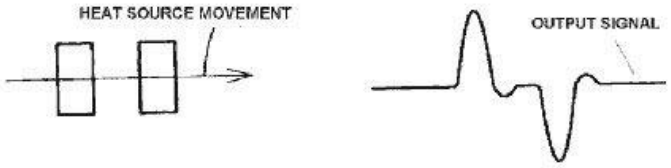
In kamer 2 staat een bewegingssensor die gebruikt kan worden voor het schakelen van verlichting of het alarm doen triggeren. M.a.w. als er in deze kamer beweging wordt gedetecteerd zal de verlichting in de kamer aan gaan. Maar als het alarmsysteem geactiveerd is zal het alarm getriggerd worden en dan zal de verlichting pinken en een geluid laten horen. Om het alarm uit te schakelen zal je een code moeten ingeven op het scherm.



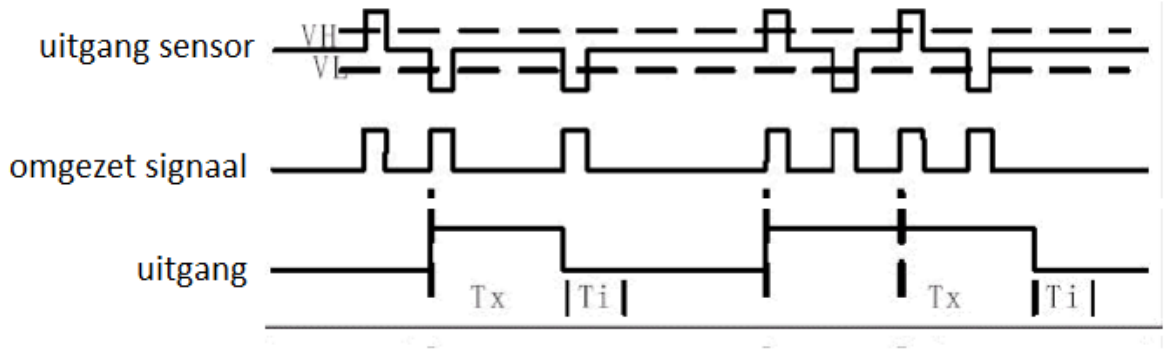
Elk object dat een temperatuur heeft boven het absoluut nulpunt straalt infrarood golven uit. Deze infraroodsensor kan de straling opvangen en omzetten in een spanning. Als dan een persoon of object met een hoog genoegen temperatuur langs het zicht van de 2 elementen beweegt zullen er 2 pulsen worden gegenereerd. Een positieve puls en een negatieve puls.



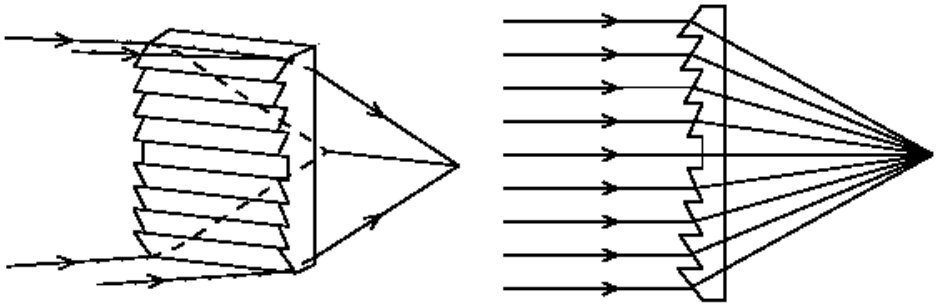

Deze sensor staat op een print met een decoder. De decoder zal de negatieve pulsen omzetten in positieve pulsen. Er zullen2 pulsen nodig zijn om een hoog signaal aan de uitgang te geven. Dit betekent dat er beweging is.




Tx = De tijdsduur wanneer uitgangspen (Vo) na triggeren hoog blijft.

Ti = Gedurende deze periode wordt triggeren geblokkeerd

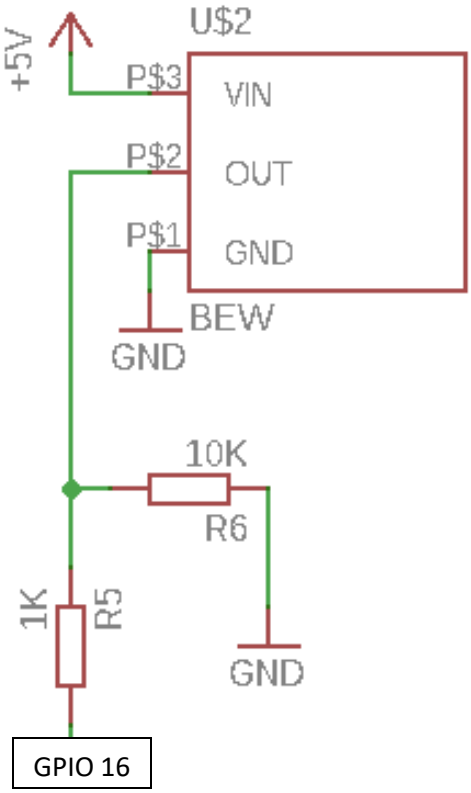
De frensel lens zorgt ervoor dat de straling op de sensor wordt gefocust. Dit zorgt voor de 120° hoek. Elke zeshoek focust de straling op de sensor.



Snelle spesificaties:

| |
|---|
| Detectie hoek: 120° |
| Detectie afstand: 0-7M |
| Voeding: 5V |

Schema bewegingssensor:



### b) Camera

De camera staat buiten het huis en observeert elke kamer. Deze camera wordt gebruikt als netwerkcamera zodat de website en het touchscreen beide het beeld kunnen ontvangen. De software dat ik hiervoor gebruik kan beweging detecteren en opnames maken.

Op het scherm zal je het huis kunnen observeren. Als het alarmsysteem actief is zal de camera zich gedragen als een bewegingssensor en het alarm triggeren als er beweging is. Als dan het alarm getriggerd wordt zal er een opname worden gemaakt die later kan geraadpleegd worden.

| |
|---|
| Logitech C170 |
| 1024 x 768  30fps |
| USB 2.0 |

## 4.3 Alarmsysteem

Het alarmsysteem maakt gebruik van de delen van bewegingsdetectie en een contact dat in de deur is gewerkt om te detecteren of er mensen in het huis komen. Het alarmsysteem moet eerst geactiveerd worden op het touchscreen.

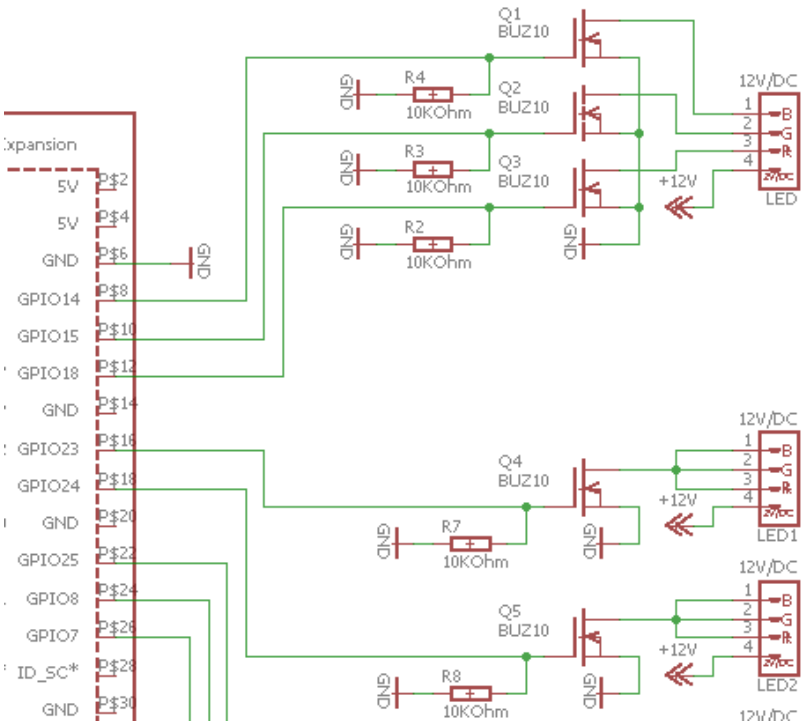
## 4.4 Verlichting

### a) Algemeen

Ik gebruik delen van een led strip (een deel = 3 leds, elke kamer één deel). Met deze realiseer ik de verlichting in mijn huis. Elk deel bestaat uit 3 leds. En elk deel heeft een vermogen van 0.72W. in kamer 1 is de verlichting RGB en in de andere niet.

### b) ansluiting

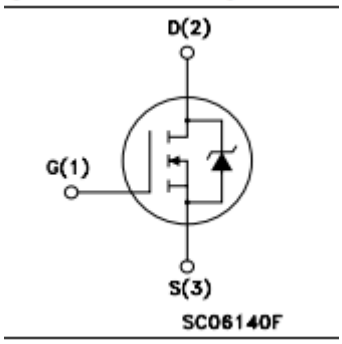
Schema Lichtsturing:




De verlichting wordt gestuurd door mosfets. Deze worden gebruikt als schakelaar. Ze onderbreken de connectie tussen de led strips en de Ground. De mosfets worden geschakeld door een positieve spanning van de Raspberry Pi GPIO pinnen. Deze spanning wordt over Ugs van de mosfet gezet. De spanning van 3.3 V is ruim voldoende om de mosfets te kunnen schakelen➔ (Ugs(th)max = 2.5V) Dit betekent dat als Ugs 2.5V is geeft de mosfet 100% zeker door van drain naar source. De N-kanaal mosfets moeten een positieve spanning aan de gate (Ugs) hebben om geschakeld te worden.

Mosfet: STP16NF06L N-kanaal

- N-kanaal
- Ugs(th)max = 2.5V: output Raspberry Pi 3.3V
- Ptot= 45W per mosfet




Meer informatie in de bijlage.

## 4.5 Verwarming

### a) Temperatuur sensor

De verwarming zal geschakeld worden door middel van een hysterese van 2°C. De verwarming zal 1° onder de ingestelde temperatuur toelaten vooraleer de verwarming aanslaat. En dan zal de verwarming de ruimte opwarmen 1° boven de ingestelde temperatuur.

De temperatuur wordt gemeten door een DS18B20+. Ik heb deze sensor gekozen omdat het 1-Wire systeem goed werkt met de RPi. En het bereik van deze sensor is ook goed (zie onderstaande tabel).

De sensor bevindt zich aan de andere kant van de kamer zo ver mogelijk van de weerstand weg en aan de onderkant van de kamer om een goede representatie te krijgen van de temperatuur in de kamer.

Deze temperatuursensor werkt op heet principe van een Silicon bandgap temperature sensor. De temperatuur wordt bepaald door de voorwaartse spanning van een silicium diode. Deze kan de base-emitter junctie van een transistor zijn.

De data wordt naar de RPi gestuurd door middel van het 1-Wire bus systeem. Dit is gelijkaardig aan I²C. In dit systeem is er altijd één master in mijn geval is dit de RPi. Met dit systeem kan je door middel van één data kabel en de ground met meerdere IC's (slaves) communiceren. De master geeft aan wanneer er data moet verstuurd worden. Elk apparaat heeft een 64-bit seriële nummer die uniek is aan de sensor. Deze bestaat uit 8-bits voor de 1-Wire familiecode(28h), 48-bit voor de unieke nummer en nog 8-bit voor een controle dat wordt uitgevoerd of de data juist is ontvangen. Dit systeem heeft een pull-up weerstand (R10 onderstaand schema) nodig om de data pin hoog te brengen. Dit systeem communiceert door deze pin laag te brengen. Het sturen van een binaire 1 gebeurt door de pin 1-15 µs laag te brengen en een 0 door de pin 60µs laag te brengen. Er word -+ 30µs na een negatieve flank uitgelezen om de bit te bepalen.

De temperatuur data wordt opgeslagen in de sensor in een 2 Byte register de 5 meest significante bits zijn voor het teken van de temperatuur (+ of -). De andere 11-bit geven het getal van de temperatuur weer.

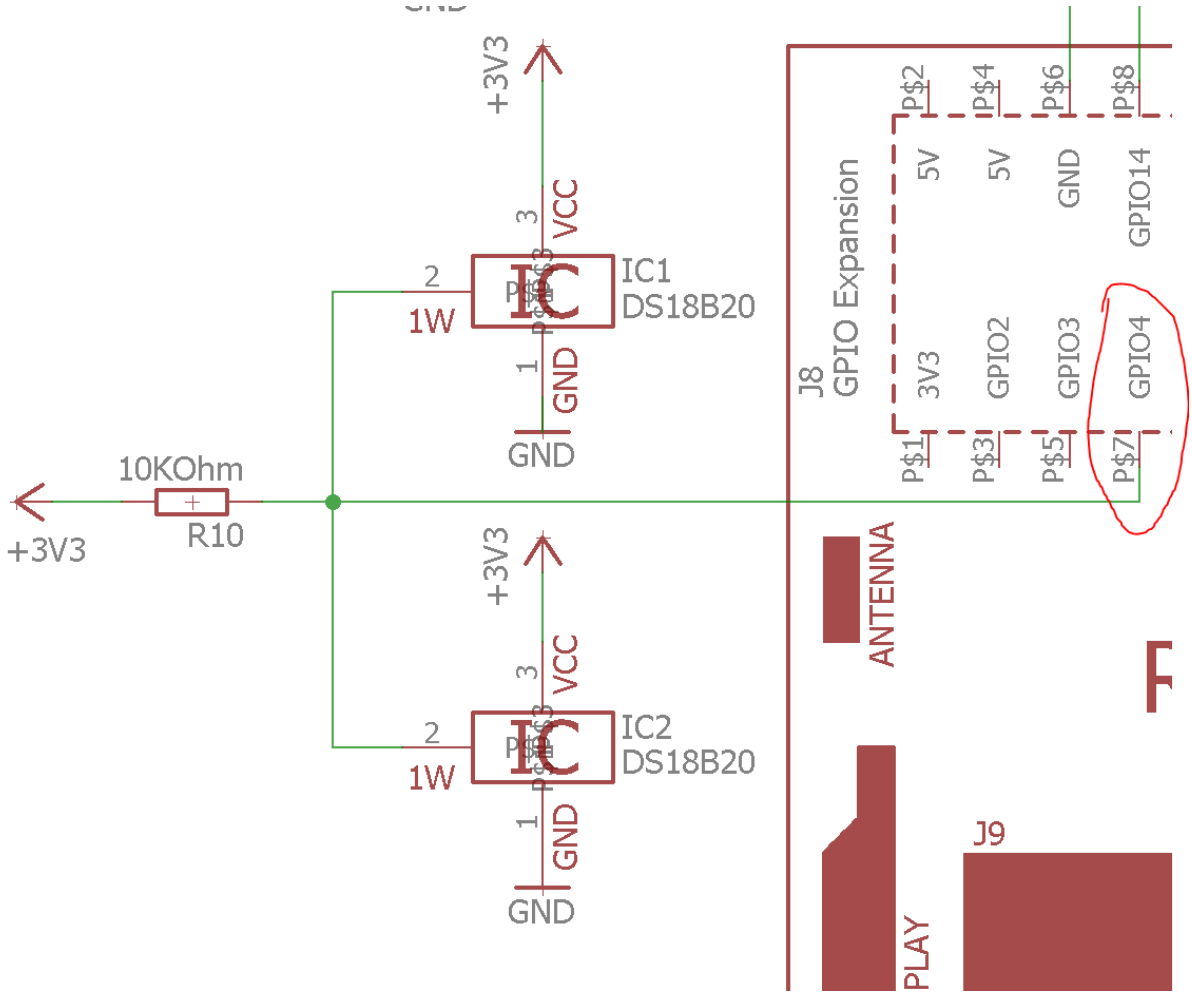
| | BIT 7 | BIT 6 | BIT 5 | BIT 4 | BIT 3 | BIT 2 | BIT 1 | BIT 0 |
|---|---|---|---|---|---|---|---|---|
| **LS BYTE** | $2^3$ | $2^2$ | $2^1$ | $2^0$ | $2^{-1}$ | $2^{-2}$ | $2^{-3}$ | $2^{-4}$ |
| | BIT 15 | BIT 14 | BIT 13 | BIT 12 | BIT 11 | BIT 10 | BIT 9 | BIT 8 |
| **MS BYTE** | S | S | S | S | S | $2^6$ | $2^5$ | $2^4$ |

S = SIGN

Specificaties:

| |
|---|
| DS18B20+ |
| Temperatuur range -55°C tot +125°C |
| Voeding 3V tot 5.5V |
| 12 bit resolutie |
| Tot +-0.5°C accuraat |

Schema:



## b) Verwarmingsweerstand

De weerstand mag niet op 100% vermogen geschakeld worden omdathet oppervlak van de verwarmingsweerstand 80°C kan worden. Dit kan mogelijk een brandgevaar met zich mee brengen. En men kan zich hier ook gemakelijk aan verbranden. Deze temperaturen zijn ook heel onrealistisch in een modern huis. De temperatuur van een moderne verwarming in een modern huis is ongeveer 40°C. Dit is te danken aan de isolatie die veel beter is dan vroeger. En nog vele andere aspecten.

Testen : verwarming op verschillende vermogens.

Ik heb met de thermometer van mijn multimeter de temperatuur gemeten. Ook heb ik (om zeker te zijn) de temperatuur sensor tegen de weerstand gezet om een 2$^{de}$ waarde te krijgen.

| Duty Cycle | Temperatuur | Spanning | Vermogen |
|---|---|---|---|
| 90% | 45°C | Ur=0.9*12V=10.8 | P=9.72W |
| 85% | 43°C | Ur=0.85*12V=10.2 | P=8.67W |
| 80% | 40°C | Ur=0.8*12V=9.6 | P=7.68W |

Specificaties:

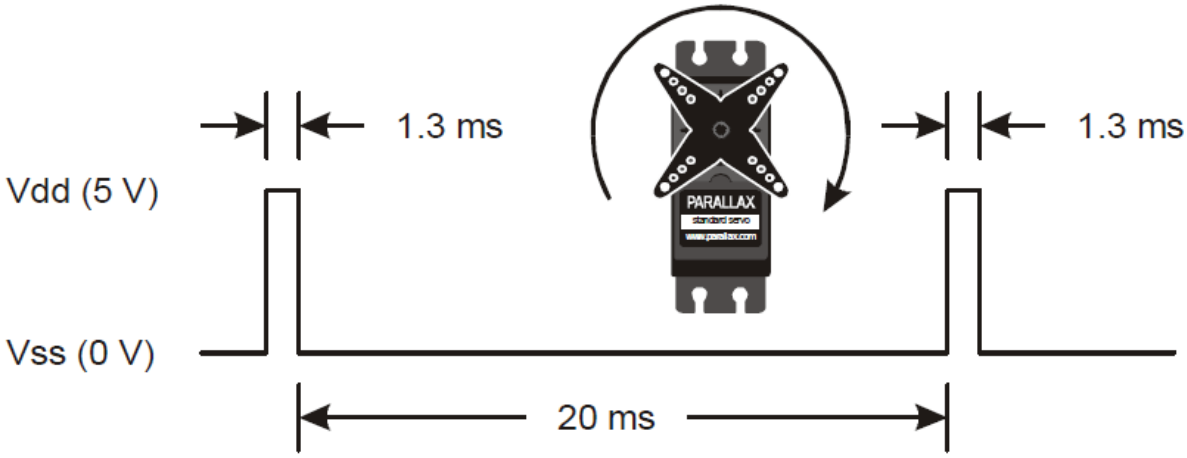
| Verwarming max. waarden |
|---|
| Vermogen 12W |
| Weerstand 12Ohm |
| Stroom 1A |
| Spanning 12V |
| Temperatuur = 80°C |

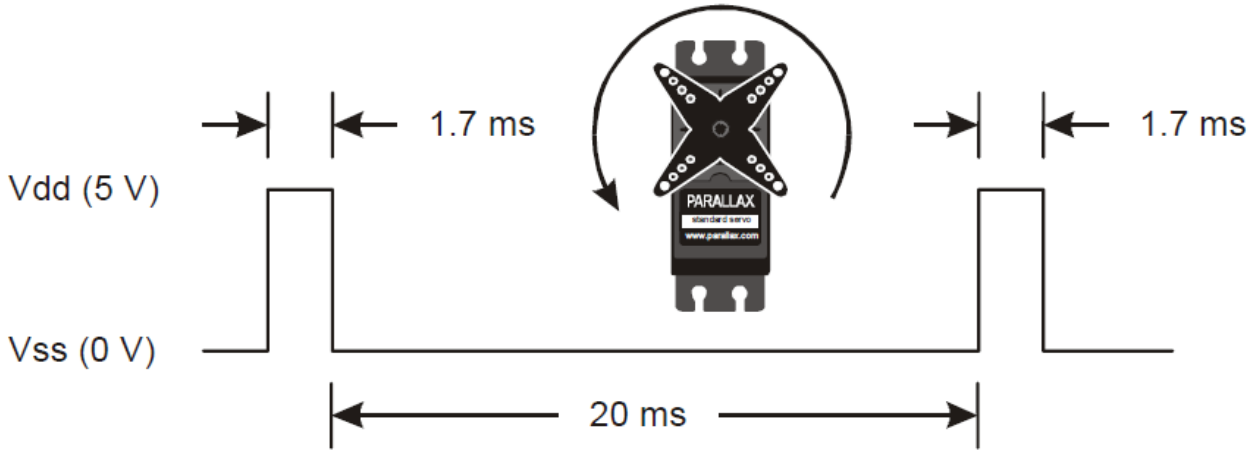
## 4.6 Rolluiken

### a) Servo motor

Het rolluik zal door een servo op en neer gaan. Ik heb een servo gekozen omdat deze gemakkelijk te sturen is met de RPi. En het koppel en de rotatiefrequentie blijven hetzelfde en worden niet beïnvloed door de reed contacten. De servo zorgt er ook voor dat de rolluik direct in beweging komt door de korte aanloop.

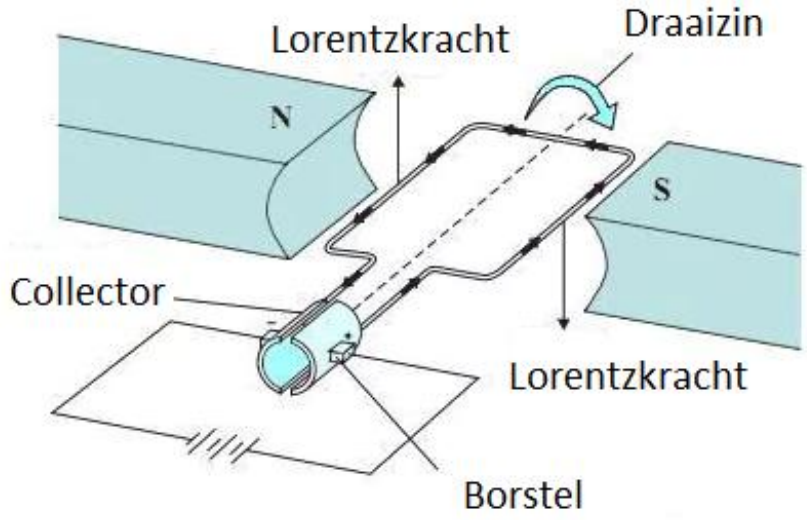
De servo motor wordt gestuurd door een PWM-signaal. De breedte van de pulsen bepalen of de motor naar rechts of naar links draait. En bepaalt ook de snelheid.






Een servo motor is opgebouwd in 2 delen: de regelelektronica en de motor.

De motor in dit geval een DC-motor. Deze maakt gebruik van permanente magneten en wikkelingen. De wikkelingen worden in een magnetisch veld gezet. Als er dan een stroom door die wikkeling zal vloeien zullen er lorentzkrachten ontstaan die de wikkeling zal verplaatsen. De zin waarin de wikkeling zich zal bewegen kan je bepalen met de linkerhand regel.

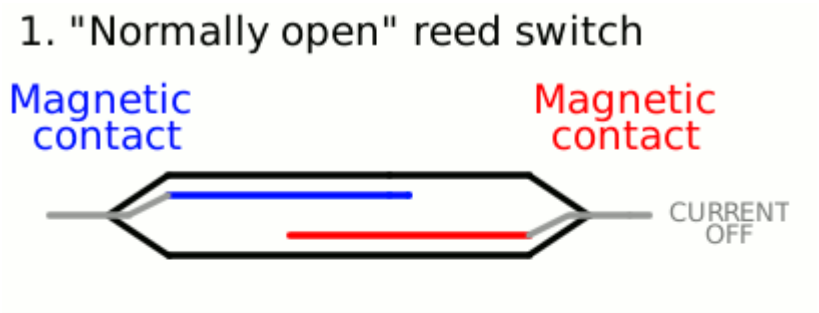


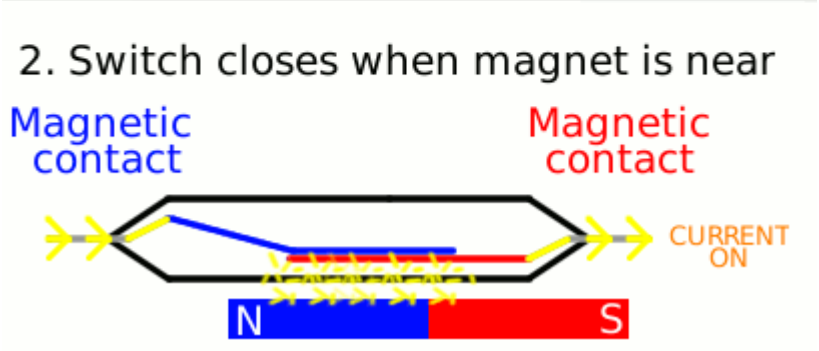
Het speciale aan een servo is dat deze DC-Motor gecontroleerd wordt. Hierdoor zal de motor bv. zijn toerental kunnen behouden ook al verandert het koppel.

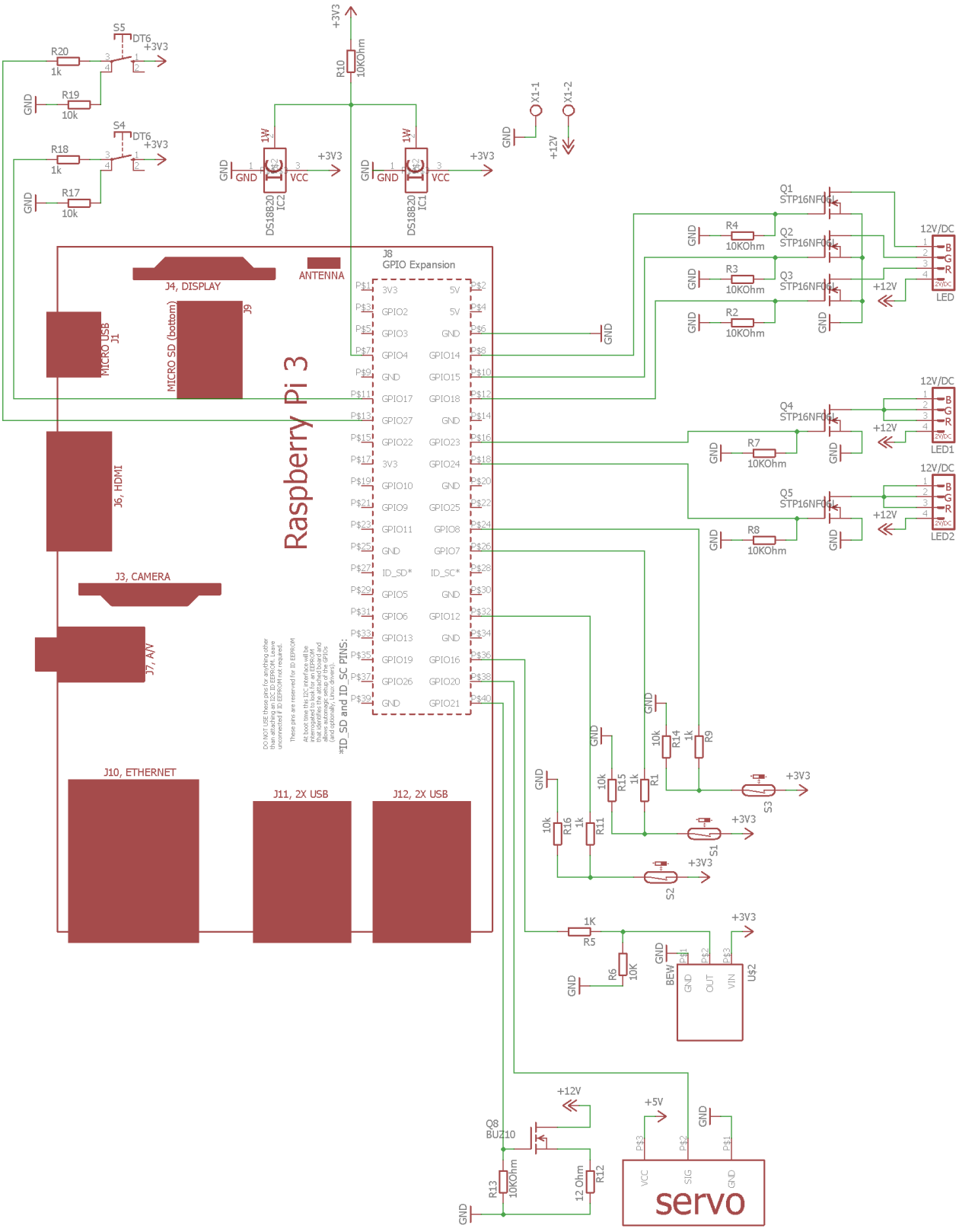
## b) Reed contact

Detectie van de stand van de rolluik gebeurt door een reed contact.

Een reed contact is een schakelaar die wordt geschakeld dooreen magneet. Deze magneet zal één van de twee beentjes van de schakelaar tegen het andere trekken en zo contact maken.



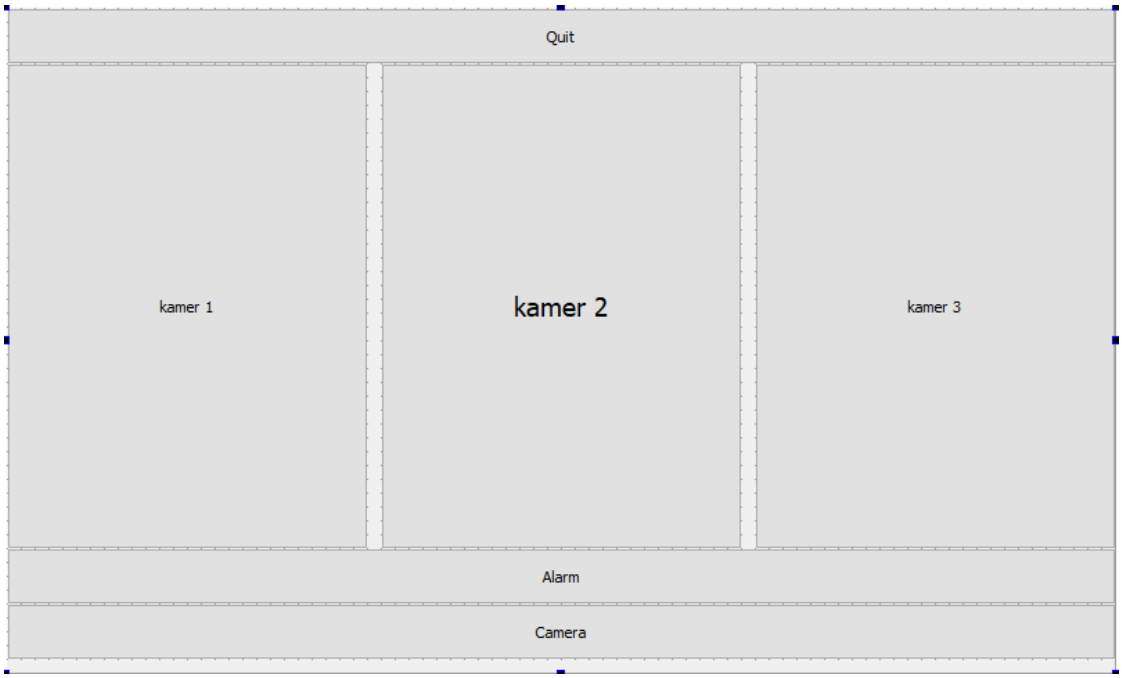

Raspberry Pi 3
J8
GPIO Expansion
J4, DISPLAY
ANTENNA
MICRO USB
J1
MICRO SD (bottom)
J9
J6, HDMI
J3, CAMERA
J7, A/V
J10, ETHERNET
J11, 2X USB
J12, 2X USB
*ID_SD and ID_SC PINS:
DO NOT USE these pins for anything other than attaching an I2C EEPROM. Leave unconnected if EEPROM not required.
These pins are reserved for ID EEPROM.
At boot time this I2C interface will be interrogated looking for an EEPROM that identifies the attached board and allows automatic configuration of the GPIOs (and optionally, Linux drivers).
S5
DT6
S4
DT6
R20
1k
R19
10k
R18
1k
R17
10k
+3V3
GND
R10
10KOhm
1W
IC2
DS18B20
IC1
DS18B20
VCC
X1-1
X1-2
+12V
Q1
STP16NF06L
Q2
STP16NF06L
Q3
STP16NF06L
R4
10KOhm
R3
10KOhm
R2
10KOhm
12V/DC
LED
Q4
STP16NF06L
R7
10KOhm
LED1
Q5
STP16NF06L
R8
10KOhm
LED2
R14
10k
1k
R9
R15
10k
1k
R1
R16
10k
1k
R11
S3
S1
S2
1K
R5
R6
10K
BEW
GND
OUT
VIN
U$2
Q8
BUZ10
R13
10KOhm
12 Ohm
R12
+5V
servo
VCC
SIG
GND

# 5 Software

## 5.1 Functionele analyse

### a) Touchscreen

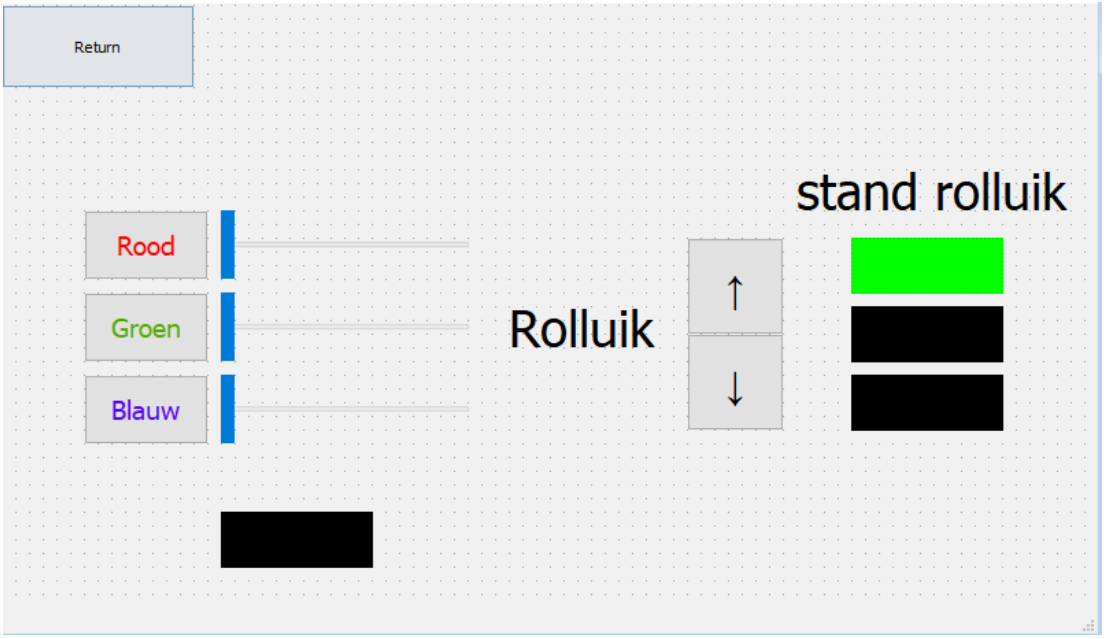
*1) kamer selectie*




Dit op dit scherm kan je kiezen welke kamer je wilt bedienen. Je kan ook het alarm activeren en naar een ander scherm van de camera gaan.

- Quit: sluit programma
- Kamer 1: ga naar kamer 1 (zie 2)
- Kamer 2: ga naar kamer 2 (zie 3)
- Kamer 3: ga naar kamer 3 (zie 4)
- Alarm: Zet alarm actief. Als het alarm actief is wordt het scherm vervangen door een codeslot om het alarm te deactiveren (zie 5)
- Camera: Ga naar camera scherm (zie 6)

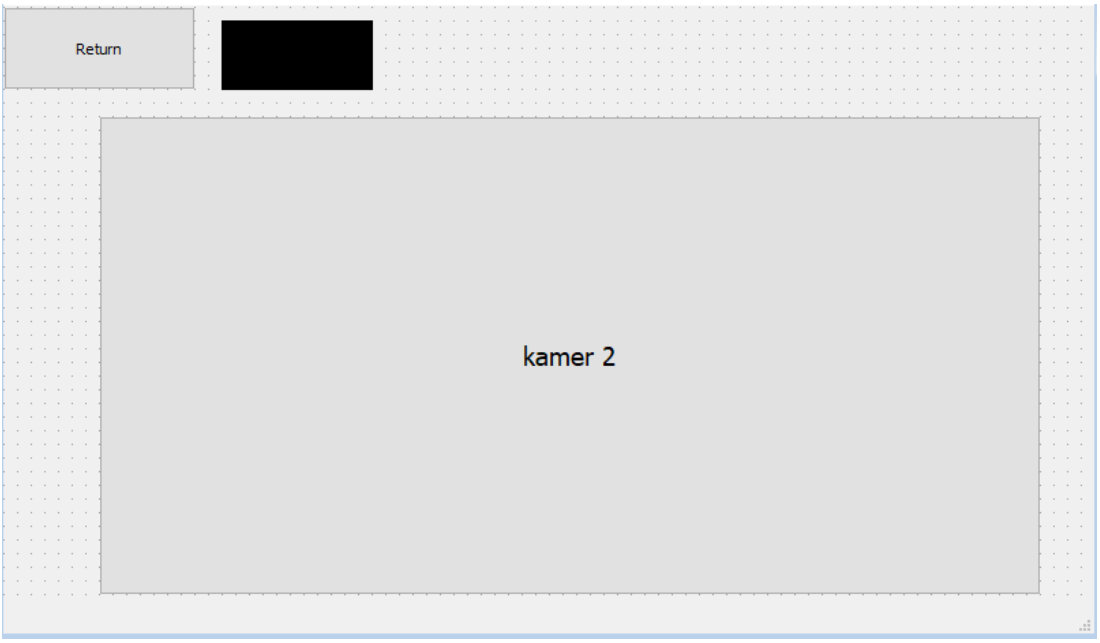
*2) Kamer1*




In kamer 1 kan elke kleur van de verlichtring apart bediend worden en gedimd worden. Hier kan ook de rolluik van stand veranderd worden en gezien worden waar de rolluik staat.

- Return: ga terug naar kamer selectie
- Kleuren: zet kleuren apart aan en uit
- Sliders: helderheid van kleur apart
- Rood vierkant onder de sliders: status verlichting, rood betekend dat de verlichting aan is en rood betekent dat de verlichting uit is
- ↑: rolluik naar boven
- ↓: rolluik naar boven
- Blokken naast de pijlen: stand van de rolluik, het groene blokje komt overeen met waar de rolluik is.

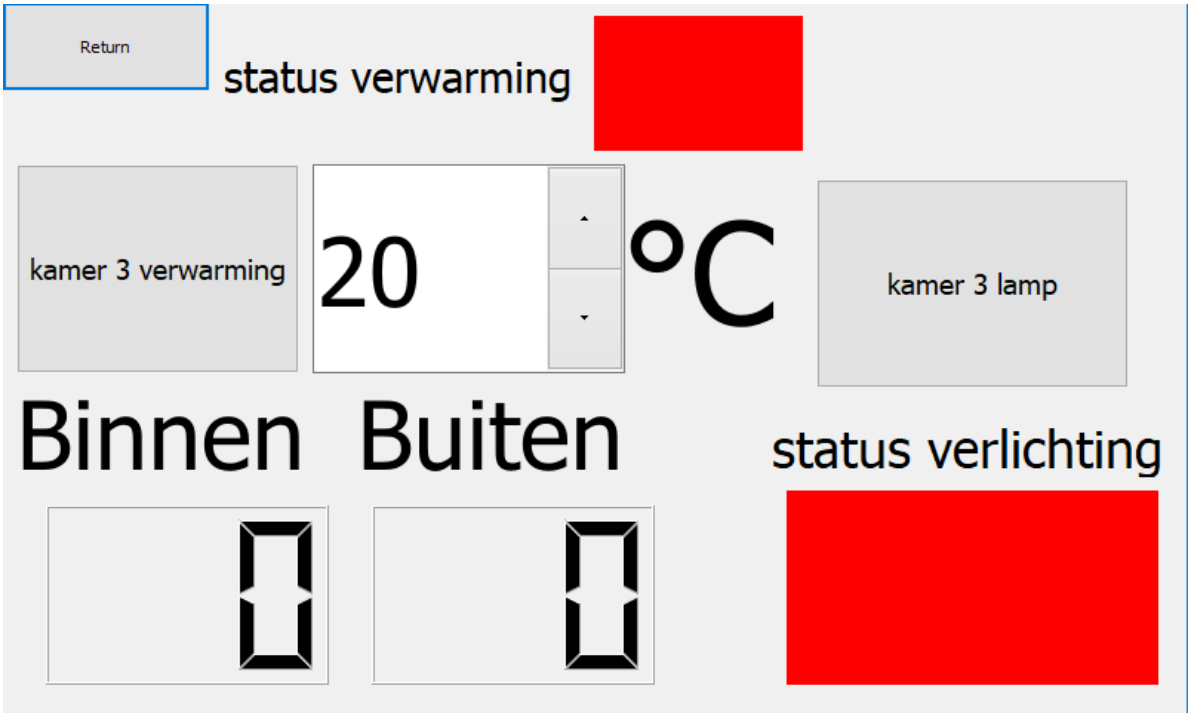
### *3) kamer 2*




In deze kamer gaat de verlichting automatisch aan maar ik heb hier toch nog een manuele knop.

- Return: ga terug naar kamer selectie
- Kamer 2: zet verlichting aan
- Blokje: status verlichting, rood betekend dat de verlichting aan is en rood betekent dat de verlichting uit is.

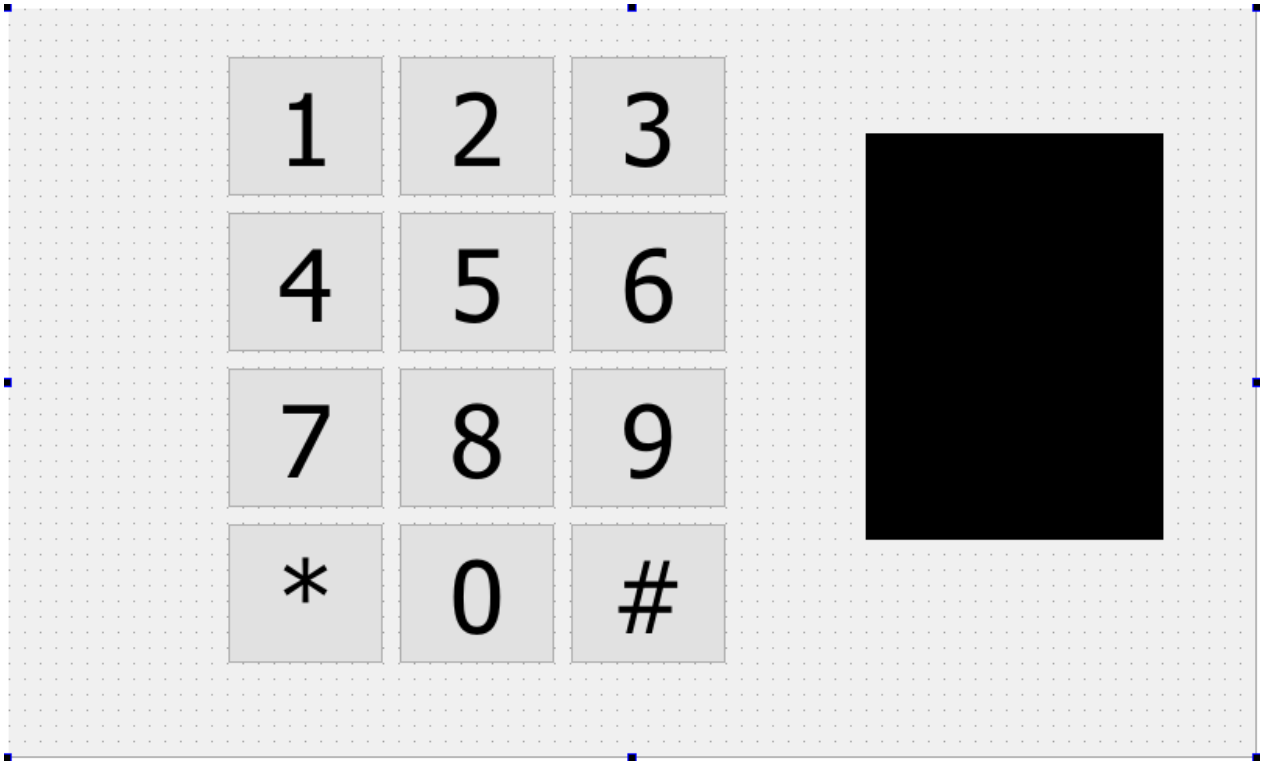
*4) kamer 3*




Op dit scherm kan je de verwarming in de kamer aan en uit schakelen en de temperatuur kiezen. De buiten en binnen temperatuur kan je aflezen. En je kan ook de verlichting aan en uit schakelen.

- Return: ga terug naar kamer selectie
- Kamer 3 verwarming: aan/uit knop voor verwarming
- Getal naast °C: ingestelde temperatuur voor verwarming
- Getallen onder binnen en buiten: de temperatuur binnen en buiten
- Kamer 3 lamp: aan/uit knop verlichting

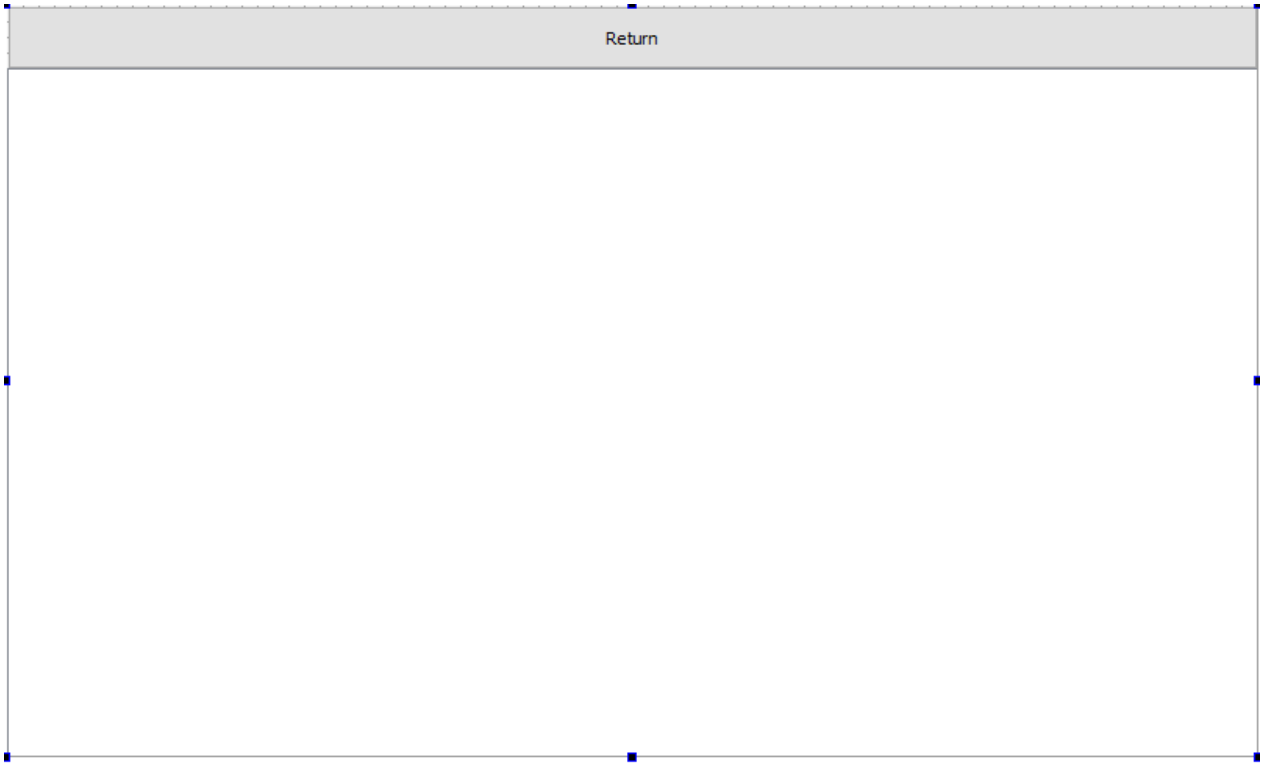
*5) alarm*




Op dit scherm moet je een code indrukken om het alarm te deactiveren.

- *: bevestig code
- #: begin opnieuw met het ingeven van de code
- Blok: als het rood wordt na bevestiging is de code fout. Als het groen wordt is de code juist en zal je terug naar de kamer selectie gaan.

### *6) camera*

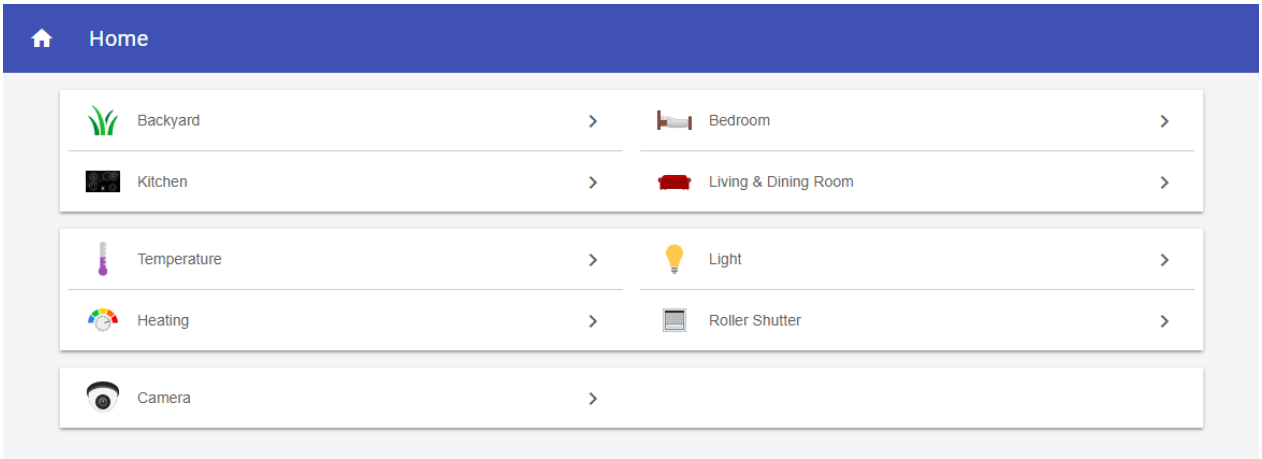



Op dit scherm kan je door de camera het huis observeren.

- Return: ga terug naar kamer selectie

### b) Website

Hier zijn de componenten opgedeeld in kamers en onderwerpen. Hier kan je evenveel als op het touchscreen.




## 5.2 Design

### a) gebruikte software

#### *1) Python*

Is de programmeertaal dat ik heb gekozen. Ik heb deze gekozen omdat het gemakkelijk in gebruik is en ik wou deze taal beter leren kennen.

### *2) PyQt 5*

PyQt 5 is een library die word gebruikt om UI's te maken voor computerapplicaties. Ik heb PyQt5 gekozen omdat het veel mogelijkheden heeft en het is uitgebreider dan andere bibliotheken.

#### Venster setup

```
class Main(QDialog):
    def __init__(self):
        super(Main, self).__init__()

        # setup scherm
        self.ui = Ui_Main()
        self.ui.setupUi(self)

        # conecties knoppen met functies
        self.ui.kamer1.clicked.connect(self.change_room1)
        self.ui.kamer2.clicked.connect(self.change_room2)
        self.ui.kamer3.clicked.connect(self.change_room3)
        self.ui.quit.clicked.connect(self.close_application)
        self.ui.camera.clicked.connect(self.change_camera)
        self.ui.alarm.clicked.connect(self.alarm)

        # toon scherm zonder rand
        self.setWindowFlags(QtCore.Qt.FramelessWindowHint)
        self.move(0, 0)
        self.show()
```

PyQt werkt met classes voor elk scherm. Dit is het hoofdscherm. Hier zeg ik welke knoppen verbonden zijn met welke functies. Hier kan je ook andere elementen toevoegen zoals een venster zonder rand.

#### Veranderen van scherm

```
def alarm(self):  # toon scherm alarm
    global w
    w = KeyPad()
    w.setWindowFlags(QtCore.Qt.FramelessWindowHint)
    w.move(0, 0)
    w.show()
```

Hier verander ik naar een ander scherm. In dit voorbeeld verander ik het scherm naar dat van het alarm.

#### Achtergrond veranderen voor signalen.

```
self.ui.boven.setStyleSheet("QFrame { background-color: %s }" %
                            green.name())
```

Ik gebruik QFrames als status blok. Ik verander de achtergrond kleur van de Qframe. In dit geval verander ik de kleur naar groen.

### *3) OpenHAB*

OpenHAB is open source software voor Domotica. Het kan connectie maken met veel verschillende Domotica toepassingen (Phillips Hue) en protocollen (mqtt). Ik gebruik dit voor mijn website. Er zijn verschillende panelen die je kan gebruiken. Eén van de panelen kan je zelf per persoon veranderen en de andere is voor iedereen hetzelfde.

### *4) Motion*

Motion is software waarmee je een usb webcam kan gebruiken als netwerkcamera. Zo kan ik de camera feed gemakkelijk laten zien op de site.

### *5) Database*

De database software dat ik gebruik is Mysql ik gebruik deze omdat meerdere programma's gemakkelijk data kunnen schrijven naar een database. In deze database komen de toestanden van de onderdelen terecht. En de temperaturen worden bijgehouden. OpenHAB maakt ook gebruik van een database om bepaalde configuraties bij te houden.

Structuur:

Tabel temp:

| Locatie(TEKST) | Datum (DATETIME) | Temp(FLOAT) |
|---|---|---|
| binnen | 2018-02-23 20:00:07 | 32 |

In deze database gaat de temperatuur bijgehouden worden.

Tabel device:

| Datum(DATETIME) | Device(TEXT) | Status(INT) |
|---|---|---|
| 2018-02-23 20:00:03 | verwarmingset | 20 |

Hier ga ik de waardes updaten naargelang er knoppen worden ingedrukt. Ik voeg niets bij in deze tabel.

### Database sturing

```
c2.execute("SELECT value FROM devices WHERE device = 'reeddown'")
for row in c2.fetchall():
    reeddown = row[0]
```

Ik lees uit een database wat de status van een onderdeel is.

```
c.execute("INSERT INTO temp (locatie, tijd, temp) VALUES (%s, %s, %s)",
          (locatie, date, int(temp2)))
conn.commit()
```

Ik schrijf ook naar mijn database Bv. wat de temperatuur is voor de site en het touchscreen.

### *6) MQTT*

Dit is een communicatieprotocol dat vaak gebruikt wordt door IoT controllers. Het werkt op het principe van een broker en cliënt. De broker is een server dat als hij data krijgt het zal sturen naar alle cliënts die geconnecteerd zijn. Als je data wilt ontvangen moet je bij de cliënt subscriben op een onderwerp. Dit onderwerp heeft data dat zowel een getal als een woord kan zijn.

| Onderwerp: Huis/keuken/verlichting |
| --- |
| Data: ON |

#### MQTT-sturing

```
mqttclient.publish("home/outdoor/temp", temp2)
```

Ik stuur naar de broker het onderwerp met de temperatuur in dit voorbeeld.

```
mqttclient.message_callback_add("home/living/light", on_message_living_light)
```

Dit is om te abonneren op een onderwerp. Als er dan een bericht komt met dat onderwerp zal de data worden doorgegeven naar een functie.

```
def on_message_living_light(client, userdata, message):
```

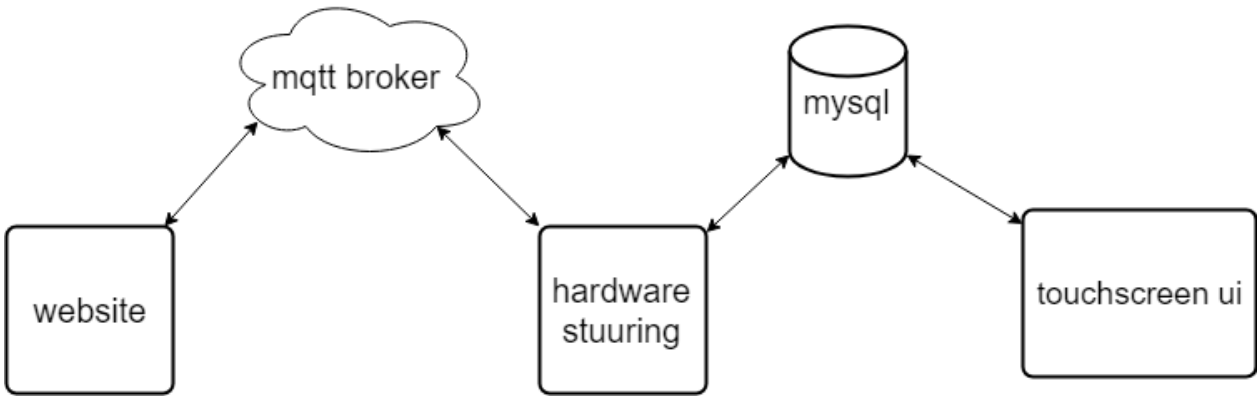
### *7) Interrupt:*

Interrupts gebruik ik voor mijn hardware input. Ik doe dit omdat dit veel gemakkelijker is. En ik kan niet alle inputs blijven bekijken omdat anders mijn ui niet meer reageert. Dit betekend dat mijn hardware altijd voorrang heeft op mijn software. Om een interrupt toe te voegen doe ik dit commando.

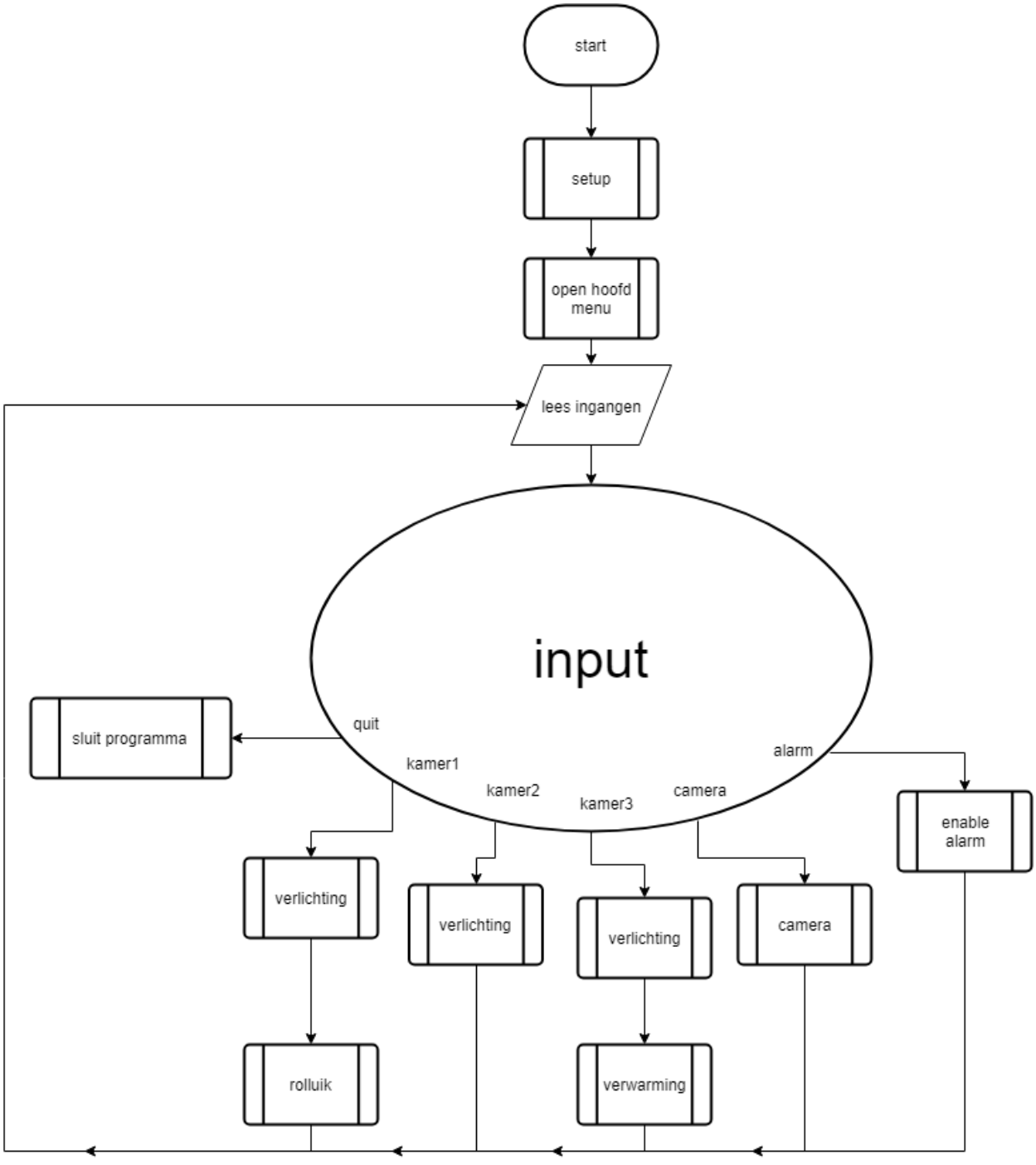
```
GPIO.add_event_detect(rolluik_naar_beneden, callback=rolluikup,
bouncetime=500)*
```

In dit commando zeg ik dat de input een interrupt triggert als de pin hoog word. De interupt handler is de functie rolluikup en de input wordt softwarematig ontdendert.

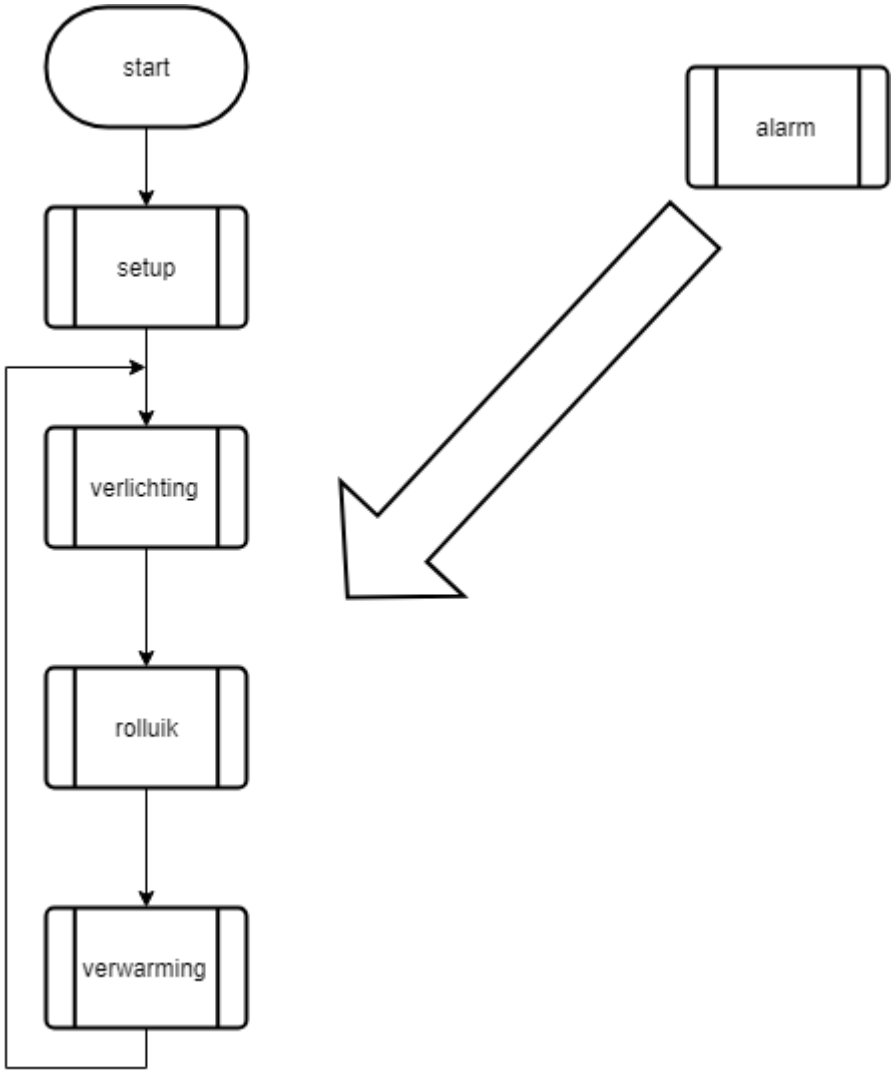
### *8) Communicatie schema:*



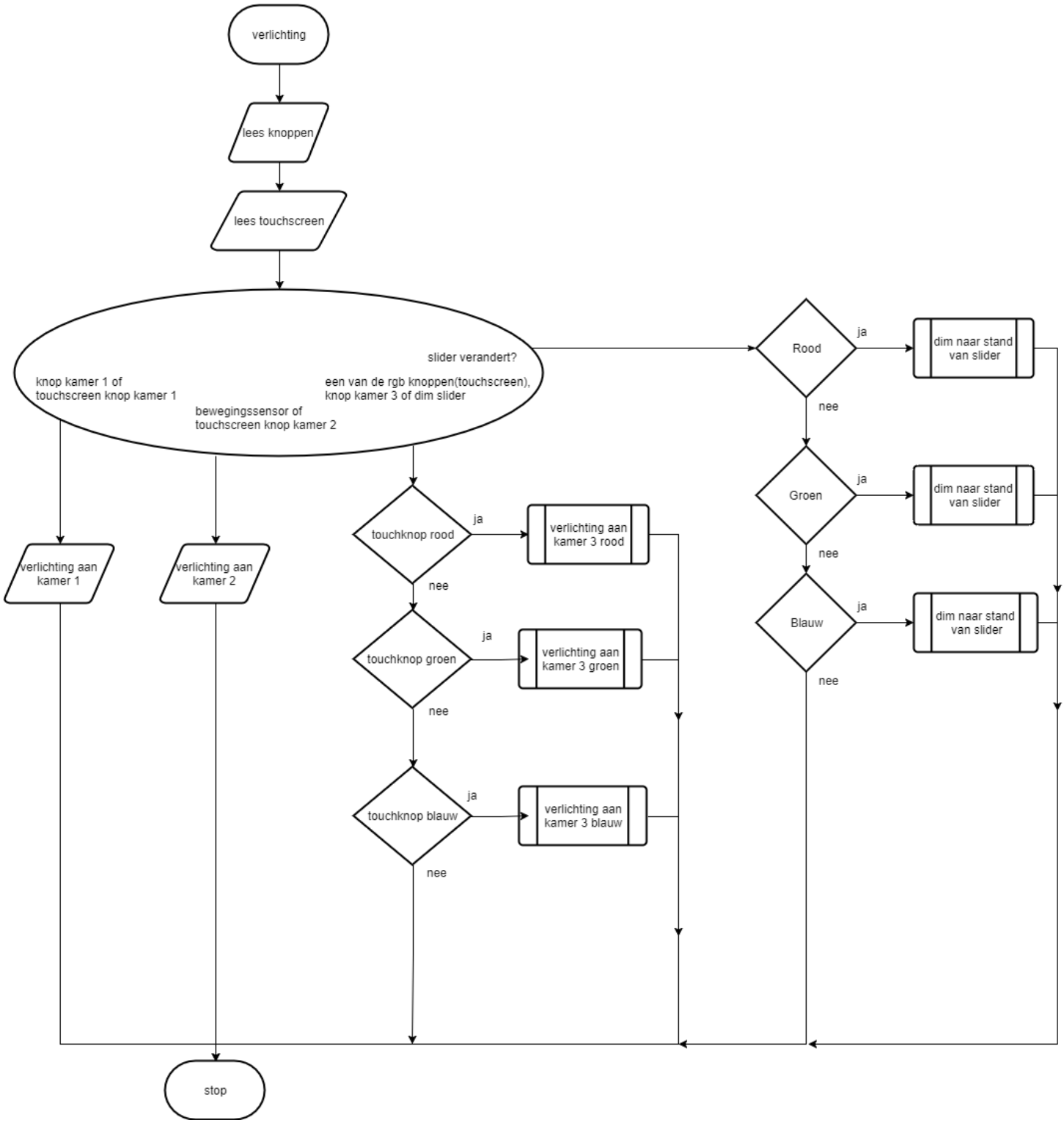
## b) Hoofd flowchart touchscreen



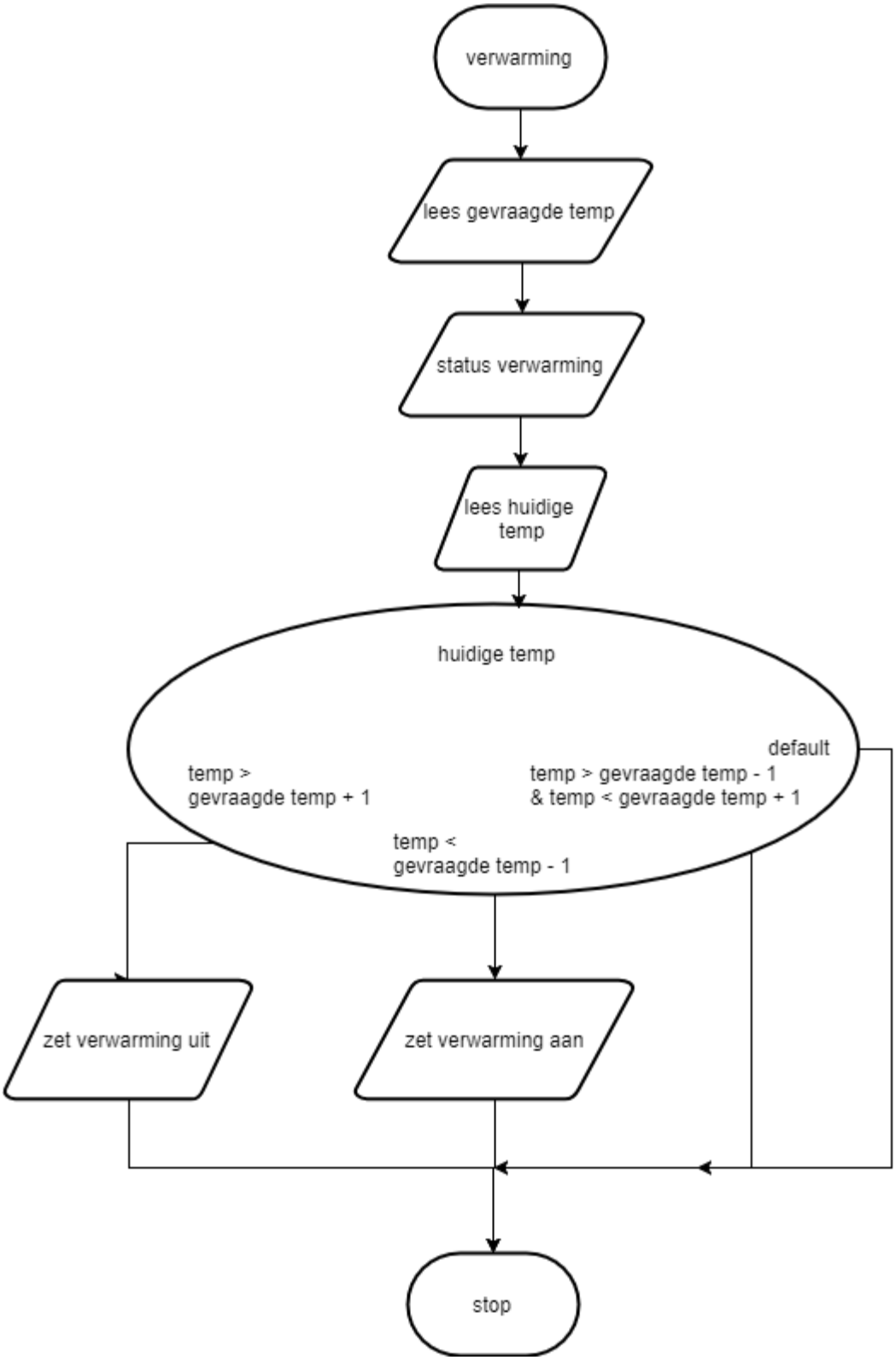
## c) Hoofd flowchart loop



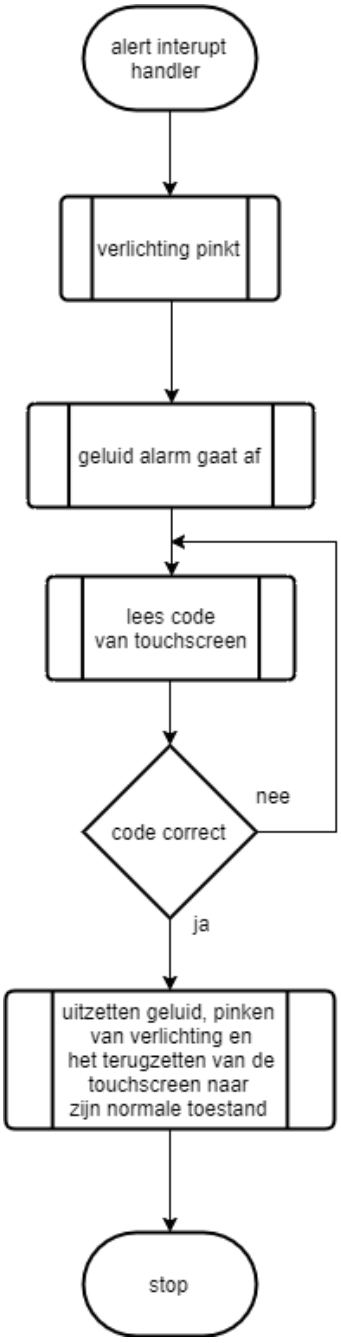
## d) Verlichting flowchart



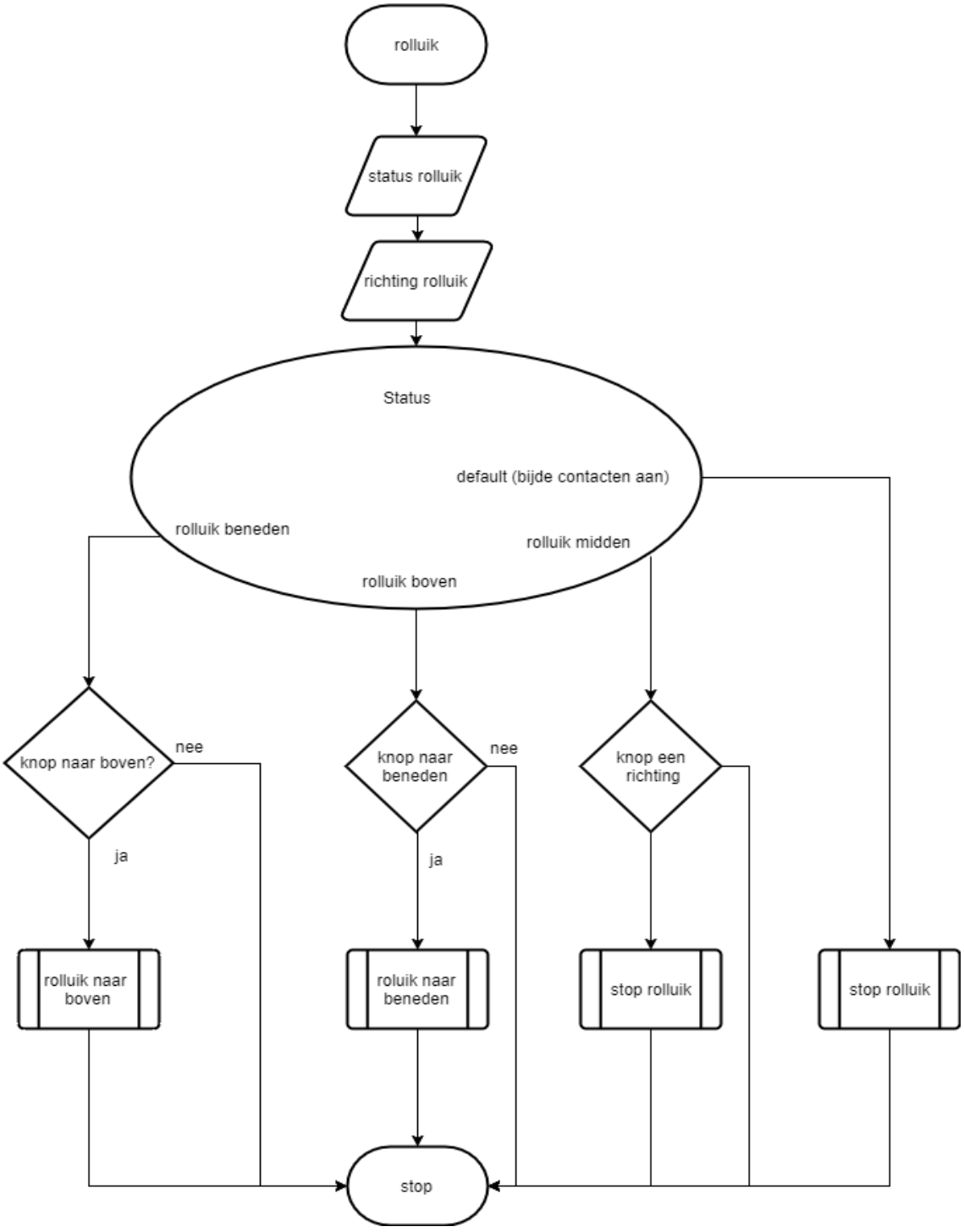
## e) Verwarming flowchart



## f) Beveiliging flowchart



## g) Rolluik flowchart



## 5.3 Implementatie

### Touchscreen

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import QtCore, QtGui
import mysql.connector as mysql
import time
import datetime
from main import Ui_Main
from room1 import Ui_Room1
from room2 import Ui_Room2
from room3 import Ui_Room3
from camera2 import Ui_Camera
from keypad import Ui_KeyPad

# connectie maken met de database
conn = mysql.connect(user='root', password='masta', database='gip')
c = conn.cursor()

# veriabele voor de tijd
unix = int(time.time())
date = str(datetime.datetime.fromtimestamp(unix).strftime('%Y-%m-%d %H:%M:%S'))

# globale variabelen voor gebruik tussen functies
Ron = False
Gon = False
Bon = False
room3 = False
room2 = False
temp1 = 0
temp2 = 0
asked = 20
Von = False
digit = 0
code = []

# PyQt kleuren
green = QtGui.QColor(0, 255, 0)
red = QtGui.QColor(255, 0, 0)
black = QtGui.QColor(0, 0, 0)


class Main(QDialog):
    def __init__(self):
        super(Main, self).__init__()

        # setup scherm
        self.ui = Ui_Main()
        self.ui.setupUi(self)

        # connecties knoppen met functies
        self.ui.kamer1.clicked.connect(self.change_room1)
        self.ui.kamer2.clicked.connect(self.change_room2)
        self.ui.kamer3.clicked.connect(self.change_room3)
        self.ui.quit.clicked.connect(self.close_application)
        self.ui.camera.clicked.connect(self.change_camera)
        self.ui.alarm.clicked.connect(self.alarm)

        # toon scherm zonder rand
        self.setWindowFlags(QtCore.Qt.FramelessWindowHint)
        self.move(0, 0)
```

```
        self.show()

    def alarm(self):  # toon scherm alarm
        global w
        w = KeyPad()
        w.setWindowFlags(QtCore.Qt.FramelessWindowHint)
        w.move(0, 0)
        w.show()

    def change_room1(self):  # toon scherm kamer1
        global w
        w = Room1()
        w.setWindowFlags(QtCore.Qt.FramelessWindowHint)
        w.move(0, 0)
        w.show()

    def change_room2(self):  # toon scherm kamer2
        global w
        w = Room2()
        w.setWindowFlags(QtCore.Qt.FramelessWindowHint)
        w.move(0, 0)
        w.show()

    def change_room3(self):  # toon scherm kamer3
        global w
        w = Room3()
        w.setWindowFlags(QtCore.Qt.FramelessWindowHint)
        w.move(0, 0)
        w.show()

    def change_camera(self):  # toon scherm kamer1
        global w
        w = Camera()
        w.setWindowFlags(QtCore.Qt.FramelessWindowHint)
        w.move(0, 0)
        w.show()

    def close_application(self):  # sluit programma
        print("bye")
        c.execute("UPDATE devices SET value = 1 WHERE device = 'stop'")
        conn.commit()
        c.close()
        conn.close()
        sys.exit()


class Room1(QDialog):
    def __init__(self):
        super(Room1, self).__init__()

        # setup scherm
        self.ui = Ui_Room1()
        self.ui.setupUi(self)

        # timer om de functie te laten herhalen
        self.ui.rolluik_timer = QtCore.QTimer()
        self.ui.rolluik_timer.timeout.connect(self.rolluik_stand)
        self.ui.rolluik_timer.start(100)

        # conecties knoppen met functies
        self.ui.home.clicked.connect(change_main)
        self.ui.rood.clicked.connect(self.setColor)
```

```
        self.ui.blauw.clicked.connect(self.setColor)
        self.ui.groen.clicked.connect(self.setColor)
        self.ui.blauwsld.valueChanged[int].connect(self.dimBlue)
        self.ui.roodsld.valueChanged[int].connect(self.dimRed)
        self.ui.groensld.valueChanged[int].connect(self.dimGreen)
        self.ui.up.clicked.connect(self.rolluik_ts)
        self.ui.down.clicked.connect(self.rolluik_ts)

    def rolluik_stand(self):  # verandert de kelueren van de status blokken
        # lees de stand uit
        conn2 = mysql.connect(user='root', password='masta', database='gip')
        c2 = conn2.cursor()
        c2.execute("SELECT value FROM devices WHERE device = 'reeddown'")
        for row in c2.fetchall():
            reeddown = row[0]
        c2.execute("SELECT value FROM devices WHERE device = 'reedup'")
        for row in c2.fetchall():
            reedup = row[0]

        # verandert de kleur naar de stand van de rolluik
        if reeddown:
            self.ui.boven.setStyleSheet("QFrame { background-color: %s }" %
                                        green.name())
            self.ui.beneden.setStyleSheet("QFrame { background-color: %s }" %
                                          red.name())
            self.ui.mid.setStyleSheet("QFrame { background-color: %s }" %
                                      red.name())
        elif reedup:
            self.ui.beneden.setStyleSheet("QFrame { background-color: %s }" %
                                          green.name())
            self.ui.boven.setStyleSheet("QFrame { background-color: %s }" %
                                        red.name())
            self.ui.mid.setStyleSheet("QFrame { background-color: %s }" %
                                      red.name())
        else:
            self.ui.mid.setStyleSheet("QFrame { background-color: %s }" %
                                      green.name())
            self.ui.beneden.setStyleSheet("QFrame { background-color: %s }" %
                                          red.name())
            self.ui.boven.setStyleSheet("QFrame { background-color: %s }" %
                                        red.name())
        c2.close()
        conn2.close()

    def rolluik_ts(self):  # schrijf naar de database als een knop van de roluik op het
scherm wordt gebruikt
        source = self.sender()
        if source.text() == '↑':
            c.execute("UPDATE devices SET value = 1 WHERE device = 'servoup'")
        else:
            c.execute("UPDATE devices SET value = 1 WHERE device = 'servodown'")
        conn.commit()

    def setColor(self):  # schrijft input van de RGB knoppen in de database

        source = self.sender()

        global Ron, Gon, Bon

        if source.text() == "Rood":
            if Ron:
                c.execute("UPDATE devices SET value = 0 WHERE device = 'led1ron'")
                Ron = False
```

```
            else:
                c.execute("UPDATE devices SET value = 1 WHERE device = 'led1ron'")
                Ron = True
        elif source.text() == "Groen":
            if Gon:
                c.execute("UPDATE devices SET value = 0 WHERE device = 'led1gon'")
                Gon = False
            else:
                c.execute("UPDATE devices SET value = 1 WHERE device = 'led1gon'")
                Gon = True
        else:
            if Bon:
                c.execute("UPDATE devices SET value = 0 WHERE device = 'led1bon'")
                Bon = False
            else:
                c.execute("UPDATE devices SET value = 1 WHERE device = 'led1bon'")
                Bon = True

        conn.commit()

        # geeft het statusblokje de juiste kleur
        if Bon or Ron or Gon:
            self.ui.status.setStyleSheet("QFrame { background-color: %s }" %
                                         green.name())
        else:
            self.ui.status.setStyleSheet("QFrame { background-color: %s }" %
                                         red.name())

    def dimBlue(self, value):  # schruift de input van de slider voor blauw in de
database
        c.execute("UPDATE devices SET value = %s WHERE device = %s", (value,
'led1blue'))
        conn.commit()

    def dimRed(self, value):  # schruift de input van de slider voor rood in de
database
        c.execute("UPDATE devices SET value = %s WHERE device = %s", (value,
'led1red'))
        conn.commit()

    def dimGreen(self, value):  # schruift de input van de slider voor groen in de
database
        c.execute("UPDATE devices SET value = %s WHERE device = %s", (value,
'led1green'))
        conn.commit()


class Room2(QDialog):
    def __init__(self):
        super(Room2, self).__init__()

        # setup scherm
        self.ui = Ui_Room2()
        self.ui.setupUi(self)

        # timer om de functie te laten herhalen
        self.ui.stat = QtCore.QTimer()
        self.ui.stat.timeout.connect(self.room2stat)
        self.ui.stat.start(100)

        # conecties knoppen met functies
        self.ui.home.clicked.connect(change_main)
        self.ui.light.clicked.connect(self.room2)
```

```
    def room2(self):  # schrijf naar de database als de knop van de verlichting op het
scherm wordt gebruikt
        conn2 = mysql.connect(user='root', password='masta', database='gip')
        c2 = conn2.cursor()
        global room2
        c2.execute("SELECT value FROM devices WHERE device = 'led2'")
        for row in c2.fetchall():
            led2 = row[0]

        if led2:
            room2 = False
        else:
            room2 = True

        if room2:
            c.execute("UPDATE devices SET value = 1 WHERE device = 'led2'")

        else:
            c.execute("UPDATE devices SET value = 0 WHERE device = 'led2'")

        conn.commit()

    def room2stat(self):  # verandert de kelueren van de status blok
        conn2 = mysql.connect(user='root', password='masta', database='gip')
        c2 = conn2.cursor()
        c2.execute("SELECT value FROM devices WHERE device = 'led2'")
        for row in c2.fetchall():
            led2 = row[0]

        c2.execute("SELECT value FROM devices WHERE device = 'led2bew'")
        for row in c2.fetchall():
            led2bew = row[0]

        if led2 or led2bew:
            self.ui.aanuit.setStyleSheet("QFrame { background-color: %s }" %
                                         green.name())
        else:
            self.ui.aanuit.setStyleSheet("QFrame { background-color: %s }" %
                                         red.name())
        c2.close()
        conn2.close()


class Camera(QDialog):
    def __init__(self):
        super(Camera, self).__init__()

        # setup scherm
        self.ui = Ui_Camera()
        self.ui.setupUi(self)

        # conecties knoppen met functies
        self.ui.home.clicked.connect(change_main)


class Room3(QDialog):
    def __init__(self):
        super(Room3, self).__init__()

        # setup scherm
        self.ui = Ui_Room3()
        self.ui.setupUi(self)
```

```
        self.temp()

        # timer om de functie te laten herhalen
        self.ui.stat = QtCore.QTimer()
        self.ui.stat.timeout.connect(self.room3stat)
        self.ui.stat.timeout.connect(self.verwarmingstat)
        self.ui.stat.start(100)

        # conecties knoppen met functies
        self.ui.home.clicked.connect(change_main)
        self.ui.verlivht.clicked.connect(self.room3)
        self.ui.verwarm.clicked.connect(self.verwarm_knop)
        self.ui.temp.setValue(verwarming)
        self.ui.temp.valueChanged[int].connect(self.verwarm)

    def room3(self):  # schrijf naar de database als de knop van de verlichting op het
scherm wordt gebruikt
        conn2 = mysql.connect(user='root', password='masta', database='gip')
        c2 = conn2.cursor()
        global room3
        c2.execute("SELECT value FROM devices WHERE device = 'led3'")
        for row in c2.fetchall():
            led3 = row[0]

        if led3:
            room3 = False
        else:
            room3 = True

        if room3:
            c.execute("UPDATE devices SET value = 1 WHERE device = 'led3'")

        else:
            c.execute("UPDATE devices SET value = 0 WHERE device = 'led3'")

        conn.commit()
        c2.close()
        conn2.close()

    def room3stat(self):  # verandert de kelueren van de status blok voor de
verlichting
        conn2 = mysql.connect(user='root', password='masta', database='gip')
        c2 = conn2.cursor()
        c2.execute("SELECT value FROM devices WHERE device = 'led3'")
        for row in c2.fetchall():
            led3 = row[0]

        if led3:
            self.ui.stat_verlicht.setStyleSheet("QFrame { background-color: %s }" %
                                                green.name())
        else:
            self.ui.stat_verlicht.setStyleSheet("QFrame { background-color: %s }" %
                                                red.name())
        c2.close()
        conn2.close()

    def verwarm(self, value):  # schrijf naar de database als de slider van de
verwarming op het scherm wordt verandert
        global verwarming
        c.execute("UPDATE devices SET value = %s WHERE device = %s", (value,
'verwarmingset'))
        conn.commit()
        verwarming = value
```

```
    def verwarm_knop(self):   # schrijf naar de database als de knop van de verwarming
op het scherm wordt gebruikt
        conn2 = mysql.connect(user='root', password='masta', database='gip')
        c2 = conn2.cursor()
        c2.execute("SELECT value FROM devices WHERE device = 'verwarming'")
        for row in c2.fetchall():
            heat = row[0]

        if heat:
            Von = False
        else:
            Von = True

        if Von:
            c.execute("UPDATE devices SET value = 1 WHERE device = 'verwarming'")
        else:
            c.execute("UPDATE devices SET value = 0 WHERE device = 'verwarming'")
        conn.commit()
        c2.close()
        conn2.close()

    def verwarmingstat(self):  # verandert de kelueren van de status blok voor de
verwarming
        conn2 = mysql.connect(user='root', password='masta', database='gip')
        c2 = conn2.cursor()
        c2.execute("SELECT value FROM devices WHERE device = 'verwarming'")
        for row in c2.fetchall():
            heat = row[0]

        if heat:
            self.ui.stat_verwarm.setStyleSheet("QFrame { background-color: %s }" %
                                               green.name())
        else:
            self.ui.stat_verwarm.setStyleSheet("QFrame { background-color: %s }" %
                                               red.name())
        c2.close()
        conn2.close()

    def temp(self):  # lees de temperatuur voor het touchscreen
        conn2 = mysql.connect(user='root', password='masta', database='gip')
        c2 = conn2.cursor()
        global temp1

        c2.execute("SELECT temp FROM temp WHERE locatie = 'binnen'")
        for row in c2.fetchall():
            temp1 = row[0]

        self.ui.Tbinnen.display(temp1)

        global temp2

        c2.execute("SELECT temp FROM temp WHERE locatie = 'buiten'")
        for row in c2.fetchall():
            temp2 = row[0]
        self.ui.Tbuiten.display(temp2)
        c2.close()
        conn2.close()


class KeyPad(QDialog):
    def __init__(self):
        super(KeyPad, self).__init__()
```

```
        # setup scherm
        self.ui = Ui_KeyPad()
        self.ui.setupUi(self)

        # conecties knoppen met functies
        self.ui.een.clicked.connect(self.disable)
        self.ui.twee.clicked.connect(self.disable)
        self.ui.drie.clicked.connect(self.disable)
        self.ui.vier.clicked.connect(self.disable)
        self.ui.vijf.clicked.connect(self.disable)
        self.ui.zes.clicked.connect(self.disable)
        self.ui.zeven.clicked.connect(self.disable)
        self.ui.acht.clicked.connect(self.disable)
        self.ui.negen.clicked.connect(self.disable)
        self.ui.nul.clicked.connect(self.disable)
        self.ui.ster.clicked.connect(self.disable)
        self.ui.hek.clicked.connect(self.disable)

        c.execute("UPDATE devices SET value = 1 WHERE device = 'alarm'")
        conn.commit()

    def disable(self):  # functie voor het code om het alarm uit te schakelen
        source = self.sender()
        global digit, code

        self.ui.fout.setStyleSheet("QFrame { background-color: %s }" %
                                   black.name())

        if source.text() == '*':
            if code == ['6', '5', '4', '3', '2', '1']:
                global w
                c.execute("UPDATE devices SET value = 0 WHERE device = 'alarm'")
                conn.commit()
                w = Main()
                w.setWindowFlags(QtCore.Qt.FramelessWindowHint)
                w.move(0, 0)
                w.show()
                code.clear()
                digit = 0

            else:
                code.clear()
                self.ui.fout.setStyleSheet("QFrame { background-color: %s }" %
                                           red.name())

        else:
            code.insert(0, source.text())
            digit = digit + 1

        if source.text() == '#':
            code.clear()

        print(code)


def change_main():  # verandert het scherm naar het hoofdmenu
    global w
    w = Main()
    w.setWindowFlags(QtCore.Qt.FramelessWindowHint)
    w.move(0, 0)
    w.show()
```

```
app = QApplication(sys.argv)
w = Main()
sys.exit(app.exec_())
```

## Hardware sturen en uitlezen

```
import mysql.connector as mysql
import time
import datetime
import RPi.GPIO as GPIO
import paho.mqtt.client as mqtt

# pinnen
RGBred = 15
RGBgreen = 18
RGBblue = 14
ledkamer2 = 23
ledkamer3 = 24
bewegingssensor = 16
reed_deur = 12
reed_boven = 8
reed_beneden = 7
servo = 20
verwarmingpin = 21
rolluik_naar_boven = 27
rolluik_naar_beneden = 17

# temperatuur sensor id
sensorid1 = "28-000009acebe1"
sensorid2 = "28-0000097bf4f6"

# globale variabelen voor gebruik tussen functies
temp1 = 0
temp2 = 0
servo_beneden = 8.5
servo_boven = 6.5
richting = 'geen'
last2 = 'led'
last3 = 'led'
lastV = 'von'

# setup voor GPIO pinnen
GPIO.setmode(GPIO.BCM)
GPIO.setup(RGBred, GPIO.OUT)
GPIO.setup(RGBgreen, GPIO.OUT)
GPIO.setup(RGBblue, GPIO.OUT)
GPIO.setup(ledkamer2, GPIO.OUT)
GPIO.setup(ledkamer3, GPIO.OUT)
GPIO.setup(servo, GPIO.OUT)
GPIO.setup(verwarmingpin, GPIO.OUT)
GPIO.setup(bewegingssensor, GPIO.IN)
GPIO.setup(reed_boven, GPIO.IN)
GPIO.setup(reed_beneden, GPIO.IN)
GPIO.setup(rolluik_naar_boven, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(rolluik_naar_beneden, GPIO.IN, pull_up_down=GPIO.PUD_UP)
GPIO.setup(reed_boven, GPIO.IN)
GPIO.setup(reed_beneden, GPIO.IN)
pb = GPIO.PWM(RGBblue, 100)
pb.start(0)
pr = GPIO.PWM(RGBred, 100)
pr.start(0)
pg = GPIO.PWM(RGBgreen, 100)
pg.start(0)
heat = GPIO.PWM(verwarmingpin, 50)
heat.start(0)
pservo = GPIO.PWM(servo, 50)
pservo.start(0)
```

```
# connectie maken met de database
conn = mysql.connect(user='root', password='masta', database='gip')
c = conn.cursor()

# connectie maken met de MQTT broker
mqttclient = mqtt.Client(client_id="ian159159", clean_session=True, transport="tcp")
mqttclient.connect('localhost', port=1883)

# veriabele voor de tijd
unix = int(time.time())
date = str(datetime.datetime.fromtimestamp(unix).strftime('%Y-%m-%d %H:%M:%S'))


def temp():  # leest de temperatuur en schrijft het naar de database, verzend
temperatuur naar de MQTT broker
    global temp1, temp2
    tfile = open("/sys/bus/w1/devices/" + sensorid1 + "/w1_slave")
    text = tfile.read()
    tfile.close()
    secondline = text.split("\n")[1]
    temperaturedata = secondline.split(" ")[9]
    temperature = float(temperaturedata[2:])
    temp1 = temperature / 1000
    locatie = 'binnen'
    c.execute("INSERT INTO temp (locatie, tijd, temp) VALUES (%s, %s, %s)",
              (locatie, date, int(temp1)))
    mqttclient.publish("home/indoor/temp", temp1)

    tfile2 = open("/sys/bus/w1/devices/" + sensorid2 + "/w1_slave")
    text2 = tfile2.read()
    tfile2.close()
    secondline2 = text2.split("\n")[1]
    temperaturedata2 = secondline2.split(" ")[9]
    temperature2 = float(temperaturedata2[2:])
    temp2 = temperature2 / 1000
    locatie = 'buiten'
    c.execute("INSERT INTO temp (locatie, tijd, temp) VALUES (%s, %s, %s)",
              (locatie, date, int(temp2)))
    conn.commit()
    mqttclient.publish("home/outdoor/temp", temp2)


def on_message_verwarming(client, userdata, message):
    mqttconn = mysql.connect(user='root', password='masta', database='gip')
    mqttc = mqttconn.cursor()

    if str(message.payload) == "b'ON'":
        mqttc.execute("UPDATE devices SET value = 1 WHERE device = 'verwarming'")
    else:
        mqttc.execute("UPDATE devices SET value = 0 WHERE device = 'verwarming'")
    mqttconn.commit()
    mqttc.close()
    mqttconn.close()


def verwarming():  # zet verwarming aan of uit naargelang de temperatuur
    global lastV
    c.execute("SELECT value FROM devices WHERE device = 'verwarming'")
    for row in c.fetchall():
        von = row[0]

    c.execute("SELECT value FROM devices WHERE device = 'verwarmingset'")
    for row in c.fetchall():
```

```
        asked = row[0]

    mqttclient.publish("home/goal/temp", asked)

    if von:
        if temp1 > (asked + 1):
            heat.ChangeDutyCycle(0)
        elif temp1 < (asked - 1):
            heat.ChangeDutyCycle(90)

        if lastV != 'verwarmon':
            mqttclient.publish("home/verwarming/on/of", "ON")
        lastV = 'verwarmon'
    else:
        heat.ChangeDutyCycle(0)
        if lastV != 'verwarmof':
            mqttclient.publish("home/verwarming/on/of", "OFF")
        lastV = 'verwarmof'


def verlivhtingRGB():  # zet de RGB verlichting aan naargelang de database
    c.execute("SELECT value FROM devices WHERE device = 'led1'")
    for row in c.fetchall():
        led1 = row[0]
    c.execute("SELECT value FROM devices WHERE device = 'led1red'")
    for row in c.fetchall():
        dimRed = row[0]
    c.execute("SELECT value FROM devices WHERE device = 'led1green'")
    for row in c.fetchall():
        dimGreen = row[0]
    c.execute("SELECT value FROM devices WHERE device = 'led1blue'")
    for row in c.fetchall():
        dimBlue = row[0]
    c.execute("SELECT value FROM devices WHERE device = 'led1ron'")
    for row in c.fetchall():
        ron = row[0]
    c.execute("SELECT value FROM devices WHERE device = 'led1gon'")
    for row in c.fetchall():
        gon = row[0]
    c.execute("SELECT value FROM devices WHERE device = 'led1bon'")
    for row in c.fetchall():
        bon = row[0]

    if led1:
        if ron:
            pr.ChangeDutyCycle(dimRed)
        else:
            pr.ChangeDutyCycle(0)
        if gon:
            pg.ChangeDutyCycle(dimGreen)
        else:
            pg.ChangeDutyCycle(0)
        if bon:
            pb.ChangeDutyCycle(dimBlue)
        else:
            pb.ChangeDutyCycle(0)
    else:
        pr.ChangeDutyCycle(0)
        pg.ChangeDutyCycle(0)
        pb.ChangeDutyCycle(0)


def verlichting():  # zet de normale verlichting aan/uit naargelang de database
```

```
    global last2, last3
    c.execute("SELECT value FROM devices WHERE device = 'led2'")
    for row in c.fetchall():
        led2 = row[0]

    c.execute("SELECT value FROM devices WHERE device = 'led2bew'")
    for row in c.fetchall():
        led2bew = row[0]

    c.execute("SELECT value FROM devices WHERE device = 'led3'")
    for row in c.fetchall():
        led3 = row[0]

    if led2 or led2bew:
        GPIO.output(ledkamer2, True)
        if last2 != 'led2on' and led2bew != 1:
            mqttclient.publish("home/kitchen/light", "ON")
        last2 = 'led2on'
    else:
        GPIO.output(ledkamer2, False)
        if last2 != 'led2off':
            mqttclient.publish("home/kitchen/light", "OFF")
        last2 = 'led2off'

    if led3:
        GPIO.output(ledkamer3, True)
        if last3 != 'led3on':
            mqttclient.publish("home/bedroom/light", "ON")
        last3 = 'led3on'
    else:
        GPIO.output(ledkamer3, False)
        if last3 != 'led3off':
            mqttclient.publish("home/bedroom/light", "OFF")
        last3 = 'led3off'


def bew():  # funcrie voor de bewegingssensor
    c.execute("SELECT value FROM devices WHERE device = 'led2'")
    for row in c.fetchall():
        led2 = row[0]

    if led2:
        pass
    else:
        if GPIO.input(bewegingssensor):
            c.execute("UPDATE devices SET value = 1 WHERE device = 'led2bew'")
            GPIO.output(led2, True)
        else:
            c.execute("UPDATE devices SET value = 0 WHERE device = 'led2bew'")
            GPIO.output(led2, False)


def on_message_living_light(client, userdata, message):
    mqttconn = mysql.connect(user='root', password='masta', database='gip')
    mqttc = mqttconn.cursor()

    if str(message.payload) == "b'ON'":
        mqttc.execute("UPDATE devices SET value = 1 WHERE device = 'led1'")
    else:
        mqttc.execute("UPDATE devices SET value = 0 WHERE device = 'led1'")
    mqttconn.commit()
    mqttc.close()
    mqttconn.close()
```

```
def on_message_kitchen_light(client, userdata, message):
    mqttconn = mysql.connect(user='root', password='masta', database='gip')
    mqttc = mqttconn.cursor()

    if str(message.payload) == "b'ON'":
        mqttc.execute("UPDATE devices SET value = 1 WHERE device = 'led2'")
    else:
        mqttc.execute("UPDATE devices SET value = 0 WHERE device = 'led2'")
    mqttconn.commit()
    mqttc.close()
    mqttconn.close()


def on_message_bedroom_light(client, userdata, message):
    mqttconn = mysql.connect(user='root', password='masta', database='gip')
    mqttc = mqttconn.cursor()

    if str(message.payload) == "b'ON'":
        mqttc.execute("UPDATE devices SET value = 1 WHERE device = 'led3'")
    else:
        mqttc.execute("UPDATE devices SET value = 0 WHERE device = 'led3'")
    mqttconn.commit()
    mqttc.close()
    mqttconn.close()


def rolluikup(rolluik_naar_boven):  # schrijf hardwareknop voor de rolluik input in de
database
    c.execute("UPDATE devices SET value = 1 WHERE device = 'servoup'")
    conn.commit()


def rolluikdown(rolluik_naar_beneden):  # schrijf hardwareknop voor de rolluik input in
de database
    c.execute("UPDATE devices SET value = 1 WHERE device = 'servodown'")
    conn.commit()


def on_message_shutter(client, userdata, message):
    mqttconn = mysql.connect(user='root', password='masta', database='gip')
    mqttc = mqttconn.cursor()

    if message.payload == "b'UP'":
        mqttc.execute("UPDATE devices SET value = 1 WHERE device = 'servoup'")
    elif message.payload == "b'DOWN'":
        mqttc.execute("UPDATE devices SET value = 1 WHERE device = 'servodown'")
    else:
        mqttc.execute("UPDATE devices SET value = 1 WHERE device = 'servostop'")
    mqttconn.commit()
    mqttc.close()
    mqttconn.close()


def rolluik():  # sturing van de rolluik
    global richting
    down = 0
    up = 0
    stop = 0
    c.execute("SELECT value FROM devices WHERE device = 'servodown'")
    for row in c.fetchall():
        down = row[0]
```

```
    c.execute("SELECT value FROM devices WHERE device = 'servoup'")
    for row in c.fetchall():
        up = row[0]

    c.execute("SELECT value FROM devices WHERE device = 'servostop'")
    for row in c.fetchall():
        stop = row[0]

    if GPIO.input(reed_beneden) and up:
        pservo.ChangeDutyCycle(servo_boven)
        richting = 'boven'

    elif GPIO.input(reed_boven) and down:
        pservo.ChangeDutyCycle(servo_beneden)
        richting = 'beneden'

    elif richting == 'beneden' and GPIO.input(reed_beneden):
        pservo.ChangeDutyCycle(0)
        richting = 'geen'

    elif richting == 'boven' and GPIO.input(reed_boven):
        pservo.ChangeDutyCycle(0)
        richting = 'geen'

    elif (not GPIO.input(reed_beneden)) and (not GPIO.input(reed_boven)):

        if (richting == 'boven' or richting == 'beneden') and (up or down or stop):
            pservo.ChangeDutyCycle(0)
            richting = 'geen'

        elif richting == 'geen' and up:
            pservo.ChangeDutyCycle(servo_boven)
            richting = 'boven'

        elif richting == 'geen' and down:
            pservo.ChangeDutyCycle(servo_beneden)
            richting = 'beneden'

    elif GPIO.input(reed_beneden) and GPIO.input(reed_boven):
        pservo.ChangeDutyCycle(0)
        richting = 'geen'

    if down:
        c.execute("UPDATE devices SET value = 0 WHERE device = 'servodown'")

    if up:
        c.execute("UPDATE devices SET value = 0 WHERE device = 'servoup'")

    if stop:
        c.execute("UPDATE devices SET value = 1 WHERE device = 'servostop'")
    conn.commit()


def handler(i):  # functie voor als het alarm getrigert wordt
    time.sleep(20)
    c.execute("SELECT value FROM devices WHERE device = 'alarm'")
    for row in c.fetchall():
        alarm = row[0]

    while 1:
        if alarm:
            pass
```

```
        else:
            break


def alarm_enable():  # set interupt voor alarm
    GPIO.add_event_detect(bewegingssensor, GPIO.RISING, callback=handler,
bouncetime=10)


def alarm_disable():  # verwijder interupt voor alarm
    GPIO.remove_event_detect(bewegingssensor)


# def on_message(mosq, obj, msg):
#   print(msg.topic+" "+str(msg.qos)+" "+str(msg.payload))

# interupts voor hardware knoppen
GPIO.add_event_detect(rolluik_naar_beneden, GPIO.RISING, callback=rolluikup,
bouncetime=500)
GPIO.add_event_detect(rolluik_naar_boven, GPIO.RISING, callback=rolluikdown,
bouncetime=500)

# subscripties voor MQTT
mqttclient.message_callback_add("home/living/light", on_message_living_light)
mqttclient.message_callback_add("home/kitchen/light", on_message_kitchen_light)
mqttclient.message_callback_add("home/bedroom/light", on_message_bedroom_light)
mqttclient.message_callback_add("home/verwarming/on/of", on_message_verwarming)
mqttclient.message_callback_add("home/living/shutter", on_message_shutter)
on_message = mqttclient.on_message
mqttclient.subscribe("home/#")

mqttclient.loop_start()
```

```
while 1:  # alle functies worden doorlopen met bepaalde timing

    temp()
    verwarming()

    for i in range(0, 11):  # elke 5 seconde worden deze functies doorlopen
        c.execute("SELECT value FROM devices WHERE device = 'stop'")
        for row in c.fetchall():
            stop = row[0]

        c.execute("SELECT value FROM devices WHERE device = 'alarm'")
        for row in c.fetchall():
            alarm = row[0]

        if alarm:
            time.sleep(20)
            alarm_enable()
        else:
            alarm_disable()

        if stop:
            c.execute("UPDATE devices SET value = 0 WHERE device = 'stop'")
            conn.commit()
            mqttclient.loop_stop()
            c.close()
            conn.close()
            GPIO.cleanup()
            quit()

        for j in range(0, 499):  # elke 0.1 seconde word deze functies doorlopen
            verlivhtingRGB()
            verlichting()
            bew()
            rolluik()
            time.sleep(0.01)
```

# 6 Problemen en Oplossingen

## 6.1 RPI.GPIO PWM start stop bug

In de RPI.GPIO library kan er een bug voorkomen dat als je een PWM-signaal op een pin te veel start en stopt. Dan zal de pin uiteindelijk hoog blijven. Dit heb ik opgelost door de duty cycle naar 0 te zetten i.p.v. het te stoppen.

## 6.2 Veranderen van scherm

Het veranderen van een scherm in PyQt5 was niet gemakkelijk. Wat hij eigenlijk doet is het scherm sluiten en een nieuw openen. Dit zorgt ervoor dat elk scherm apart is. Dit heb ik opgelost door globale variabelen te gebruiken zodanig dat ik gemakkelijker tussen classes kan communiceren.

## 6.3 Database cache

Omdat de database gecached word door de mysql connector moet ik elke keer een nieuwe connectie maken om de database up to date te houden in het programma.

# 7 Besluit

Get maken van dit eindwerk was niet gemakkelijk maar zeker wel interessant. Ik heb veel geleerd over domotica en software.

Ik had het mezelf niet gemakkelijk gemaakt door zo veel mogelijk zelf te ontwerpen zonder veel van het internet te kopiëren. Maar dat vind ik net het leukste, het ontwerpen en realiseren van iets dat je nog nooit eerder hebt gedaan.

Het moeilijkste was waarschijnlijk het schrijven van de software. Ik had al een programmeertaal gekozen waar ik amper ervaring mee had en ik wou de GUI zelf maken. Daar is zeker het meeste tijd in gekropen.

Uiteindelijk heb ik veel bijgeleerd en was dit een heel interessant eindwerk.

# 8 Prijsberekening

| Onderdeel | Hoeveelheid | Prijs (totaal) |
|---|---|---|
| Conrectoren/klemmen | 3 | € 12.58 |
| Printplaat | 1 | € 3.99 |
| Kabel | x | € 7.20 |
| Weerstanden | 20 | € 1 |
| Led strip | 3 | € 1.80 |
| Temperatuur sensor | 2 | € 8.50 |
| Verwarmingsweerstand | 1 | € 7.63 |
| PIR-bewegingssensor | 1 | € 7.07 |
| Reedcontacten | 3 | € 1.89 |
| Drukknoppen | 2 | € 0.36 |
| MOSFET | 6 | € 2.70 |
| Magneten | 3 | € 2.16 |
| Servo motor | 1 | € 16.93 |
| Houd voor maquette | x | € 50 |
| Raspberry Pi + touchscreen | 1 | € 115.44 |
| Totaal | x | € 242.32 |

# 9 Beschrijving voor niet-technisch publiek

## 9.1 Domoticasysteem

Mijn gip is een domotica systeem opgebouwd en ontworpen door mijzelf. De hersenen van mij gip is de Raspberry Pi (dit is een soort van kleine computer). Deze zal alle onderdelen van mijn gip verbinden. Ik heb 3 kamers in elke kamer worden er 1 of meerdere onderdelen voorgesteld. Hieronder vind je een plan van wat waar staat.

| | Touchscreen/speaker | |
|---|---|---|
| Speciale verlichting<br>Rolluik | Verlichting<br><br>Bewegingssensor<br>Deurcontact | Verlichting<br><br>Verwarming<br>Temperatuursensor |

- In de eerste kamer heb ik speciale verlichting. Hiermee bedoel ik dat de verlichting van kleur kan veranderd worden. En de helderheid kan aangepast worden. In deze kamer is ook een rolluik dat kan bediend worden.
- In de 2de kamer heb ik standaard verlichting dat ik van het scherm en de website kan bedienen. De bewegingssensor dient om de verlichting automatisch aan en uit te laten gaan. Deze zal na detectie van beweging de verlichting een aantal seconden laten branden.
- In de 3de kamer is de verlichting ook normaal maar is er ook nog een verwarmingssysteem dat de kamer op temperatuur zal houden.
- Ik heb ook nog een alarmsysteem dat je kan activeren op het touchscreen. Het alarm zal dan op twee manieren kunnen worden getriggerd. De eerste manier is door een reed contact (schakelaar dat wordt geschakeld door er een magneet tegen te houden) in de deur. Als de deur wordt geopend terwijl het alarmsysteem actief is gaat het alarm af. De tweede manier is door de bewegingssensor. Als het alarm dan afgaat knippert de verlichting en zal de speaker een geluid maken. Om het alarm te deactiveren moet je een code ingeven op het scherm.
- Op het scherm kan je alles bedienen en de status zien van alle onderdelen. Op de site kan je ook de status zien van alle onderdelen op de rolluik na. Er is ook nog een camera waarmee je het huis kan observeren

# Bijlage

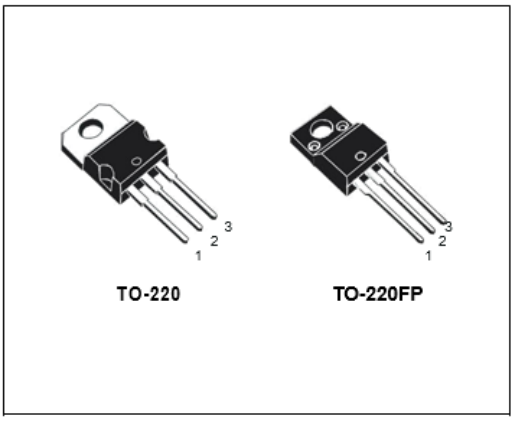
## MOSFET

# STP16NF06L
# STP16NF06LFP

## N-CHANNEL 60V - 0.07 Ω - 16A TO-220/TO-220FP
## STripFET™ II POWER MOSFET

| TYPE | $V_{DSS}$ | $R_{DS(on)}$ | $I_D$ |
|---|---|---|---|
| STP16NF06L | 60 V | <0.09 Ω | 16 A |
| STP60NF06LFP | 60 V | <0.09 Ω | 11 A |

- TYPICAL $R_{DS}(on)$ = 0.07Ω
- EXCEPTIONAL dv/dt CAPABILITY
- LOW GATE CHARGE AT 100 °C
- LOW THRESHOLD DRIVE




### DESCRIPTION

This Power MOSFET is the latest development of STMicroelectronis unique "Single Feature Size™" strip-based process. The resulting transistor shows extremely high packing density for low on-resistance, rugged avalanche characteristics and less critical alignment steps therefore a remarkable manufacturing reproducibility.

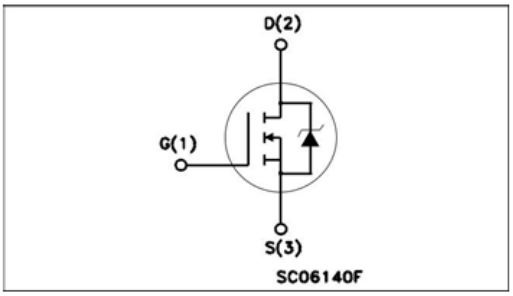
### APPLICATIONS

- MOTOR CONTROL, AUDIO AMPLIFIERS
- HIGH CURRENT, HIGH SPEED SWITCHING
- SOLENOID AND RELAY DRIVERS

▪DC-DC & DC-AC CONVERTERS▪

AUTOMOTIVE ENVIRONMENT

### INTERNAL SCHEMATIC DIAGRAM




### ABSOLUTE MAXIMUM RATINGS

| Symbol | Parameter | Value | | Unit |
|---|---|---|---|---|
| | | STP16NF06L | STP16NF06LFP | |
| $V_{DS}$ | Drain-source Voltage ($V_{GS}$ = 0) | 60 | | V |
| $V_{DGR}$ | Drain-gate Voltage ($R_{GS}$ = 20 kΩ) | 60 | | V |
| $V_{GS}$ | Gate- source Voltage | ± 16 | | V |
| $I_D$ | Drain Current (continuous) at $T_C$ = 25°C | 16 | 11(*) | A |
| $I_D$ | Drain Current (continuous) at $T_C$ = 100°C | 11 | 7.5(*) | A |
| $I_{DM}$(•) | Drain Current (pulsed) | 64 | 44(*) | A |
| $P_{tot}$ | Total Dissipation at $T_C$ = 25°C | 45 | 25 | W |
| | Derating Factor | 0.3 | 0.17 | W/°C |
| dv/dt (1) | Peak Diode Recovery voltage slope | 23 | | V/ns |
| $E_{AS}$ (2) | Single Pulse Avalanche Energy | 127 | | mJ |
| $V_{ISO}$ | Insulation Withstand Voltage (DC) | -------- | 2500 | V |
| $T_{stg}$ | Storage Temperature | -55 to 175 | | °C |
| $T_j$ | Operating Junction Temperature | | | |

(□) Pulse width limited by safe operating area.

(*) Current Limited by package's thermal resistance

(1) ISD ≤ 16A, di/dt ≤ 210A/µs, VDD ≤ V(BR)DSS, Tj ≤ TJMAX.
(2) Starting $T_j$ = 25 °C, $I_D$ = 8A, $V_{DD}$ = 30V

Verwarmingsweerstand

Conrad Electronic
Artikelnummer: 532878

# Datenblatt **Heizfolie**

# *Datasheet* ***heater***

| | |
|---|---|
| **Nennspannung:**<br>*Nominal Voltage:* | 12 V |
| **Nennleistung:**<br>*Effective Output:* | 12 W +/-10% |
| **Abmessungen [mm/Zoll]:**<br>*Dimensions [mm/inch]:* | 77 mm x 110 mm<br>*3.03" x 4.33"* |
| **Oberflächentemperatur*:**<br>*Surface temperature**:* | ca. 80 °C<br>*approx. 176 °F* |
| **Regelthermostat:**<br>*Thermostat:* | ohne<br>*without* |
| **Sicherheitsthermostat:**<br>*Safety Thermostat:* | ohne<br>*without* |
| **Dicke:**<br>*Thickness:* | ca. 0,4 mm (0.016")<br>*approx. 0.4 mm (0.016")* |
| **Träger:**<br>*Carrier:* | 125 µm Polyesterfolie<br>*125 microns polyester foil* |
| **Klebeband:**<br>*Adhesive Tape:* | Flammhemmendes Polyesterklebeband m. Silikon-Schutzpapier<br>*Flame-retardant polyester foil with silicon protective paper* |
| **Klebstoff:**<br>*Adhesive:* | Modifiziertes Acrylat<br>*Modified Acrylate* |
| **Versiegelung:**<br>*Sealing:* | Hochtemperaturbeständiger Dichtstoff<br>*High-temperature resistant sealant* |
| **Elektro-Anschluss:**<br>*Electrical Connection:* | Zwillingskabel 2x 0,75 mm²<br>*Double cable 2x 0.75 mm²* |
| **Temperaturbereich:**<br>*Temperature Range:* | Dauerbelastung -40 °C (-40 °F) bis max. +95 °C (+203 °F)<br>*Long term loading -40 °C (-40 °F) up to max. +95 °C (+203 °F)* |
| **RoHS konform:**<br>*RoHS compliant:* | Ja<br>*yes* |
| **Schutzrad:**<br>*Degree of Protection:* | IP X4<br>*IP X4* |
| **Bemerkung:** | **Achtung:** Aufgrund hoher Heizleistung, bezogen auf die Fläche, kann die Heizfolie je nach Einbausituation, ohne ausreichende Kühlung oder Temperaturregelung, überhitzen und dadurch zerstört werden! |
| *Comment:* | **Attention:** Overheating and the resulting destruction, as a consequence of high heating power of the heating foil, can be prevented by providing enough cooling or temperature control, depending on the positioning of the high power heating foil. |

* Heizfolie frei in der Luft hängend, die Wärmeabgabe erfolgt nur an die Umgebungsluft

** Temperature was measured with the heater suspended freely in the air, the heat was only given off to the ambient air

Rev. B

maxim integrated™

# DS18B20
# Programmable Resolution 1-Wire Digital Thermometer

## DESCRIPTION

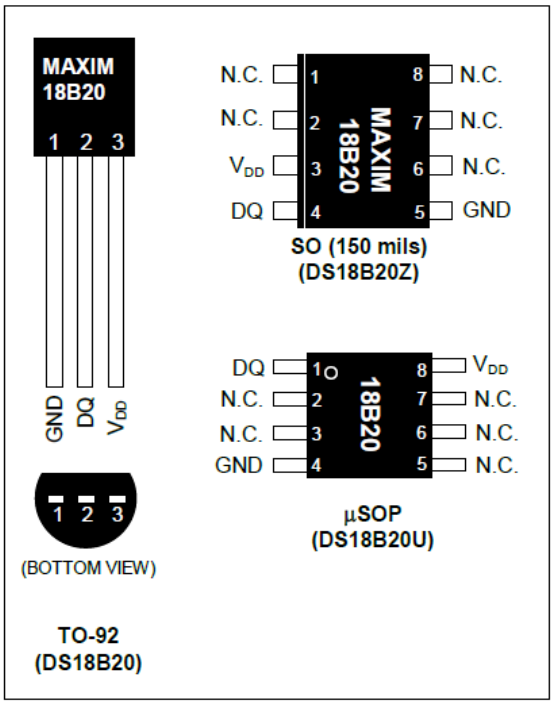
The DS18B20 digital thermometer provides 9-bit to 12-bit Celsius temperature measurements and has an alarm function with nonvolatile user-programmable upper and lower trigger points. The DS18B20 communicates over a 1-Wire bus that by definition requires only one data line (and ground) for communication with a central microprocessor. It has an operating temperature range of -55°C to +125°C and is accurate to ±0.5°C over the range of -10°C to +85°C. In addition, the DS18B20 can derive power directly from the data line ("parasite power"), eliminating the need for an external power supply.

Each DS18B20 has a unique 64-bit serial code, which allows multiple DS18B20s to function on the same 1-Wire bus. Thus, it is simple to use one microprocessor to control many DS18B20s distributed over a large area. Applications that can benefit from this feature include HVAC environmental controls, temperature monitoring systems inside buildings, equipment, or machinery, and process monitoring and control systems.

## FEATURES

- Unique 1-Wire® Interface Requires Only One Port Pin for Communication
- Each Device has a Unique 64-Bit Serial Code Stored in an On-Board ROM
- Multidrop Capability Simplifies Distributed Temperature-Sensing Applications
- Requires No External Components
- Can Be Powered from Data Line; Power Supply Range is 3.0V to 5.5V
- Measures Temperatures from -55°C to +125°C (-67°F to +257°F)
- ±0.5°C Accuracy from -10°C to +85°C
- Thermometer Resolution is User Selectable from 9 to 12 Bits
- Converts Temperature to 12-Bit Digital Word in 750ms (Max)
- User-Definable Nonvolatile (NV) Alarm Settings
- Alarm Search Command Identifies and Addresses Devices Whose Temperature is Outside Programmed Limits (Temperature Alarm Condition)
- Available in 8-Pin SO (150 mils), 8-Pin µSOP, and 3-Pin TO-92 Packages
- Software Compatible with the DS1822
- Applications Include Thermostatic Controls, Industrial Systems, Consumer Products, Thermometers, or Any Thermally Sensitive System

## PIN CONFIGURATIONS




*1-Wire is a registered trademark of Maxim Integrated Products, Inc.*

**For pricing, delivery, and ordering information, please contact Maxim Direct at 1-888-629-4642, or visit Maxim's website at www.maximintegrated.com.**

REV: 042208

## Table 1. Temperature/Data Relationship

| TEMPERATURE (°C) | DIGITAL OUTPUT (BINARY) | DIGITAL OUTPUT (HEX) |
|---|---|---|
| +125 | 0000 0111 1101 0000 | 07D0h |
| +85* | 0000 0101 0101 0000 | 0550h |
| +25.0625 | 0000 0001 1001 0001 | 0191h |
| +10.125 | 0000 0000 1010 0010 | 00A2h |
| +0.5 | 0000 0000 0000 1000 | 0008h |
| 0 | 0000 0000 0000 0000 | 0000h |
| -0.5 | 1111 1111 1111 1000 | FFF8h |
| -10.125 | 1111 1111 0101 1110 | FF5Eh |
| -25.0625 | 1111 1110 0110 1111 | FE6Fh |
| -55 | 1111 1100 1001 0000 | FC90h |

**The power-on reset value of the temperature register is +85°C.*

## AC ELECTRICAL CHARACTERISTICS (-55°C to +125°C; $V_{DD}$ = 3.0V to 5.5V)

| PARAMETER | SYMBOL | CONDITIONS | MIN | TYP | MAX | UNITS | NOTES |
|---|---|---|---|---|---|---|---|
| Temperature Conversion Time | $t_{CONV}$ | 9-bit resolution | | | 93.75 | ms | 1 |
| | | 10-bit resolution | | | 187.5 | | |
| | | 11-bit resolution | | | 375 | | |
| | | 12-bit resolution | | | 750 | | |
| Time to Strong Pullup On | $t_{SPON}$ | Start Convert T Command Issued | | | 10 | µs | |
| Time Slot | $t_{SLOT}$ | | 60 | | 120 | µs | 1 |
| Recovery Time | $t_{REC}$ | | 1 | | | µs | 1 |
| Write 0 Low Time | $t_{LOW0}$ | | 60 | | 120 | µs | 1 |
| Write 1 Low Time | $t_{LOW1}$ | | 1 | | 15 | µs | 1 |
| Read Data Valid | $t_{RDV}$ | | | | 15 | µs | 1 |
| Reset Time High | $t_{RSTH}$ | | 480 | | | µs | 1 |
| Reset Time Low | $t_{RSTL}$ | | 480 | | | µs | 1,2 |
| Presence-Detect High | $t_{PDHIGH}$ | | 15 | | 60 | µs | 1 |
| Presence-Detect Low | $t_{PDLOW}$ | | 60 | | 240 | µs | 1 |
| Capacitance | $C_{IN/OUT}$ | | | | 25 | pF | |

Servo motor

PARALLAX

**Web Site:** www.parallax.com
**Forums:** forums.parallax.com
**Sales:** sales@parallax.com
**Technical:** support@parallax.com

**Office:** (916) 624-8333
**Fax:** (916) 624-8003
**Sales:** (888) 512-1024
**Tech Support:** (888) 997-8267

# Parallax Continuous Rotation Servo (#900-00008)

The Parallax Standard Servo is ideal for adding bidirectional continuous rotation to your robotics projects.

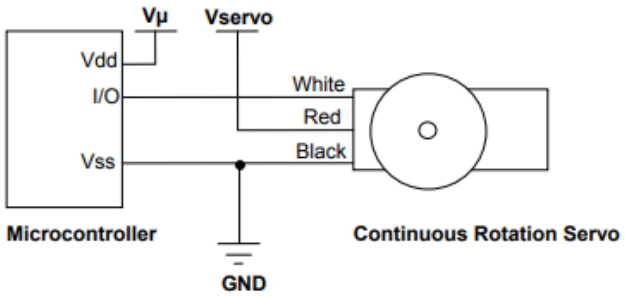
## Features


- Bidirectional continuous rotation
- 0 to 50 RPM, with a linear response to PWM for easy ramping
- Accepts four mounting screws
- Easy to interface with any Parallax microcontroller or PWM-capable device
- Very easy to control with the PULSOUT command in PBASIC or SX/B
- Weighs only 1.50 oz (42.5 g)
- 38 oz-in torque @ 6 V

## Key Specifications

- Power requirements: 4 to 6 VDC; Maximum current draw 140 +/- 50 mA at 6 VDC when operating in no load conditions, 15 mA when in static state
- Communication: pulse-width modulation
- Dimensions: approx 2.2 x 0.8 x 1.6 in (5.58x 1.9 x 4.06 cm) excluding servo horn
- Operating temperature range: 14 to 122 °F (-10 to +50 °C)

## Quick-Start Circuit




**Vμ** = microcontroller voltage supply

**Vservo** = 4 to 6 VDC, regulated or battery

**I/O** = PWM TTL or CMOS output signal, 3.3 to 5 V; < Vservo + 0.2 V

PIR-sensor gebruikt in infraroodsensor module

# Pyroelectric Passive Infrared Sensor

## General Description

The RE 200B is a passive infrared sensor designed to pick up heat radiation of wave lengths in a band around 10 microns. It contains two active elements configured as balanced differential series opposed type. This results in good compensation of environmental temperature and excellent sensitivity for small changes of a spatial temperature pattern. Thermal signals far below one microwatt are sufficient to trigger a sufficient output voltage change.

## Functional Description

If the active elements of the PIR sensor are exposed to a change in the surrounding temperature field, electrical charges are separated within the sensor elements. The voltage across the sensors controls a J-FET source follower impedance converter and thus modulates the output current of the PIR detector.

The spectral sensitivity of the sensor is controlled by the optical transfer characteristics of the window in the case and has been optimized to pick up radiation of the human body.

## Applications

- alarm systems
- consumer electronics
- human body detection
- automatic switches

## Operating Conditions

| | | |
|---|---|---|
| Operating Temperature | -20 °C to +70 °C | |
| Storage Temperature | -30 °C to +80 °C | |
| Operating Voltage | 3 to 10 V | at Rs = 47 kΩ |

## Electrical and Optical Characteristics at 25C

| | | |
|---|---|---|
| Circuit Configuration | Three-terminal sensor with source follower | |
| Appearance | TO-5 metal case with hermetic seal | |
| Output balance between elements | 15% max | at 1 Hz |
| Spectral Response | | determined by filter |
| Filter Substrate | Silicon | |
| Transmission | > 70% | average in 7 ... 14 µm range |
| Cut on wavelength | 5.0 ± 0.5 µm | at 5% T abs. |

Test Conditions: VDD = 5.0 V, Tamb = 25 C, unless otherwise noted.

| Symbol | Min | Typ | Max | Unit | Note |
|---|---|---|---|---|---|
| Idd | | 0.2 | 0.5 | mA | supply current (1) |
| VSSA | - 3.6 | - 4 | - 4.4 | V | referenced to VDD |
| IVSSA | | | 2.0 | mA | sink capability |
| VREF | 3.6 | 4 | 4.4 | V | VREF = VDD - VSSA |