Group project

IOT advanced 2CCS1

TristanBerkmans
R0784105
Wim Hembrechts
R0751084
Rutger Stessens
R0842944
Timo Van Litsenborg
R0851521

CAMPUS

Geel

Technology
Elektronics-ICT / Applied informatics

IOT

Course unit: IOT advanced

Educational activity: IOT advanced

2CCS1

Academiejaar 2021-2022

Table of contents

Table of Contents

Video link

https://youtu.be/spLrY8o-zmY

1 Raspberry pi

1.1 Inleiding

Voor het vak IOT kregen we de opdracht om een parkeergarage te bouwen. Deze parkeergarage dient te beschikken over nummerplaatherkenning en moet aangeven hoeveel plaatsen bezet zijn. We gebruiken hiervoor het programma Visual Studio Code. Alvorens van start te gaan, hebben we een schema gemaakt met de stappen die we gaan ondernemen om dit project tot een goed einde te brengen. We vinden het belangrijk om stapgsgewijs te werken en hebben een evenredige taakverdeling gemaakt.

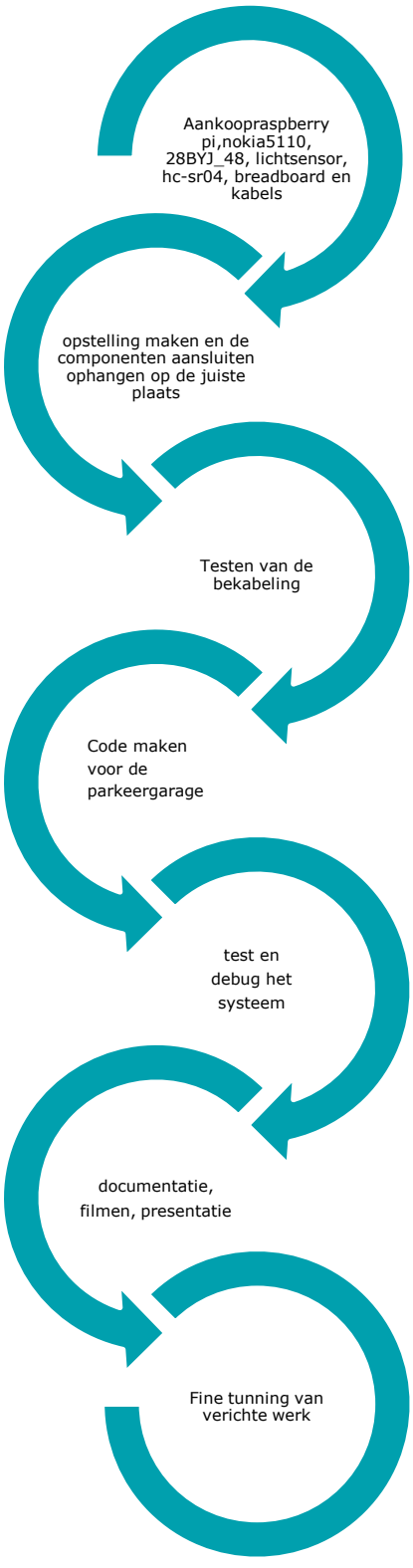
1.2 Planning

Als team hadden we besloten om een degelijke planning te maken van wat wie moet doen. Natuurlijk hebben we elkaar, waar nodig, ondersteund en geholpen. We hebben allemaal wel een moment gehad waar alles vast liep. Gelukkig konden we dan rekenen op elkaars hulp. De ondersteuning die we elkaar geboden hebben, werd niet toegevoegd in onderstaande tabel. We deden ons best alles zo snel en nauwkeurig mogelijk uit te voeren binnen de geplande tijdsspanne. Dit was niet altijd evident. In onderstaande tabel staat de uiteindelijke verdeling van de taken.

	Tristan	Wim	Rutger	Timo
29/10/2021	Bespreking	Bespreking	Bespreking	Bespreking
12/11/2021	LCD + Aansluiting	Lichtsensor + push melding	Motor(poort)	Webinterface
19/11/2021	Opstelling + Aansluitingen	Samenvoegen Code	Ultrasone sensor	Database
26/11/2021	Debuggen + Document opstellen	Nummerplaat Herkenning + Editing	Inspreken + Powerpontpresentait maken	Filmen + Editing

1.3 Schematische weergave van de opstelling

Voordat we begonnen aan dit project, hebben we de stappen uitgewerkt die we gaan volgen. We hebben deze tijdens het hele project aangehouden.

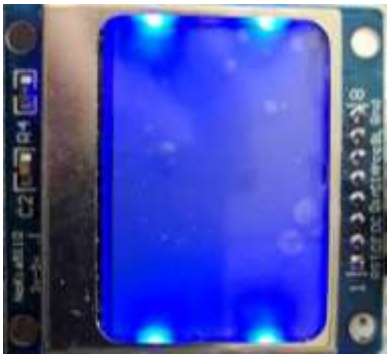


1.4 Nodige materialen

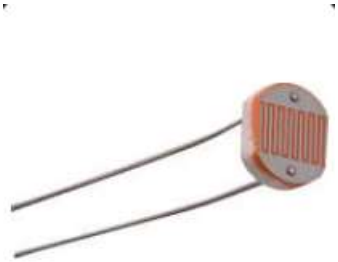
We hebben een Raspberry Pi aangeschaft om dit project tot een goed einde te brengen. De Raspberry Pi is een compact printplaatje dat zonder problemen in een behuizing ter grootte van een pakje sigaretten past en de functionaliteit van een complete computer bevat.

We hebben ook een 28BYJ_48 gekocht. Deze motor heeft een groot kracht en kan gebruikt worden om beweging te genereren.

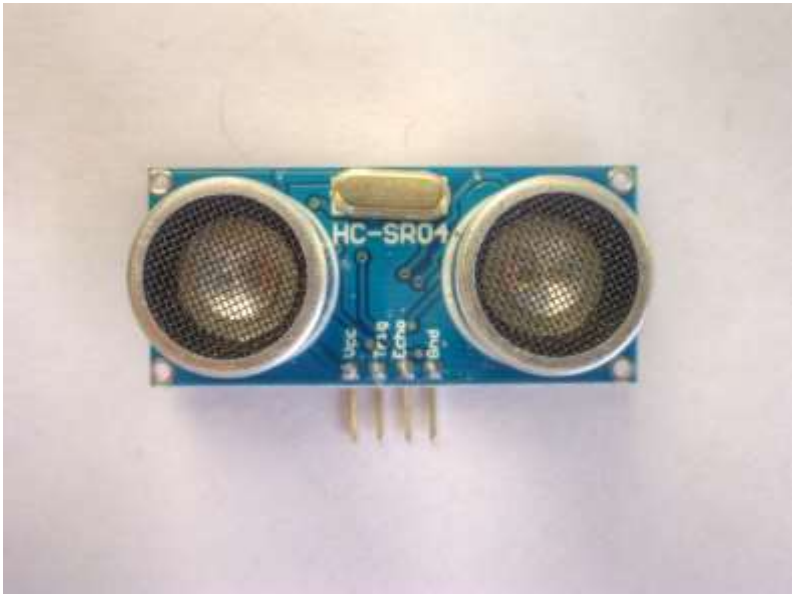
Daarnaast hebben we een nokia5110 aangeschaft. Dit is een lcd scherm dat gaat tonen welke parkeerplaatsen vrij zijn.



Ook hebben we vier lichtsensoren aangekocht. Dit is om te verifieren of er auto’s op de parking staan.

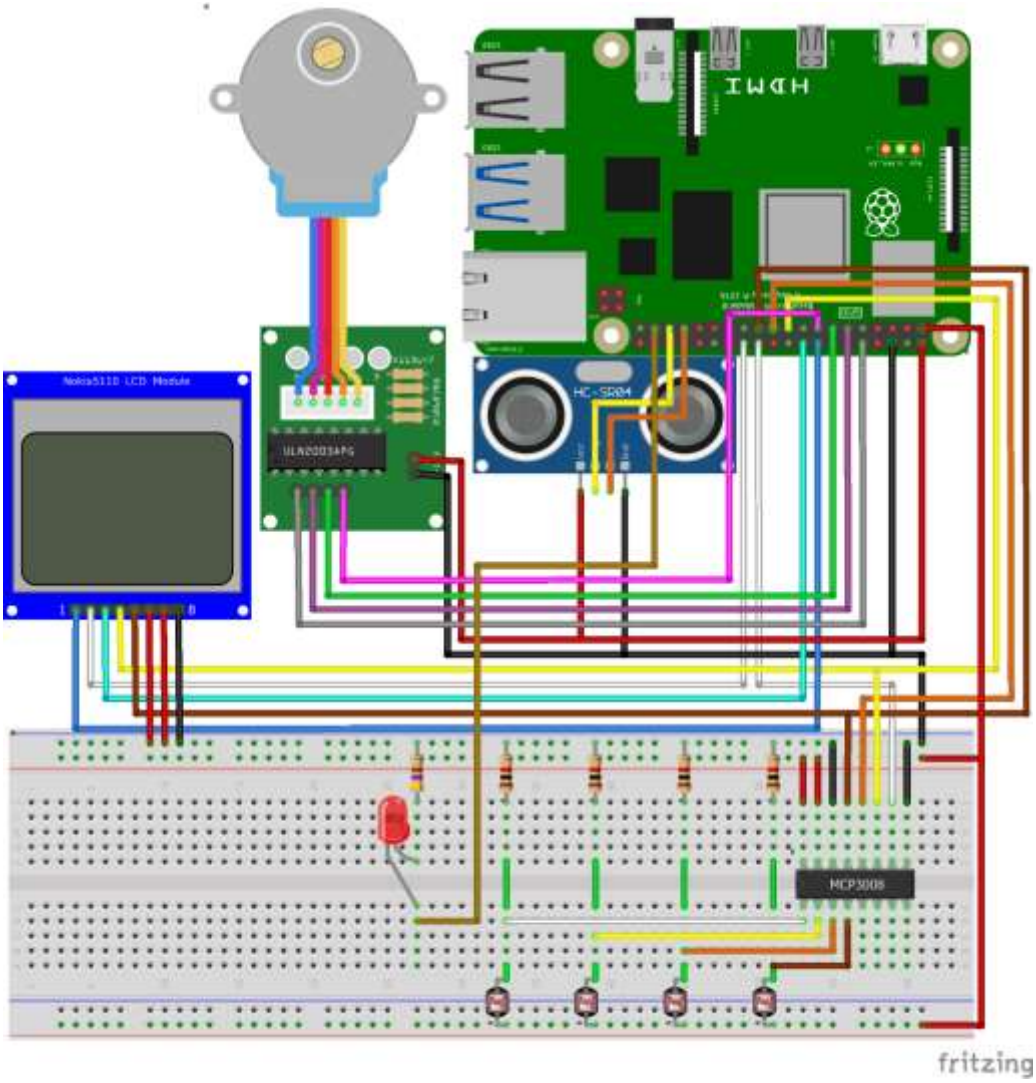


Om afstanden te meten hebben we een de HC-SR04 gekocht. Door gebruik te maken van geluidsgolven meet deze de afstand tot een voorwerp.



1.5 Blokschema en elektrisch schema

1.6 Montage



De ondernomen stappen zijn de volgenden:

1. Aansluiting scherm en testen
2. Mcp3008 aansluiten op raspberry
3. Lichtsensors connectrerren met stroom en ground en dan de waarden laten vertalen door de mcp3008
4. Het aansluiten van de motor
5. Het aansluiten van de ultrasonic sensor

1.7 Code

1.7.1 pi

```
#!/usr/bin/env python
# IMPORTING USED LIBRARIES
# Library enables the use of time functions
import time
import datetime

# Library to execute commands in CLI
import os

# Library to be able to use json files
import json

# Libraries to control MCP3008
import busio
import digitalio
import board
from adafruit_bus_device.spi_device import SPIDevice
import cgitb; cgitb.enable()
import spidev

# Libraries to control LCD display
import adafruit_pcd8544
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

# Library needed to be able to use raspberry pi's GPIO-pins
import RPi.GPIO as GPIO

# Library needed to send message via simplepush
from simplepush import send

# Library needed to connect to database
import MySQLdb

# Library to send requests
import requests

# SETUP NEEDED TO BE ABLE TO RUN THE PROGRAM CORRECTLY
# Database setup
database=MySQLdb.connect(host="localhost",user="pi",passwd="raspberry",db="Project_IOT")
cursor=database.cursor()
now = datetime.datetime.now()
cursor.execute("INSERT INTO Parking_1(available, plate, dtime) VALUES(%s, %s, %s)",
(1,"",now))
cursor.execute("INSERT INTO Parking_2(available, plate, dtime) VALUES(%s, %s, %s)",
(1,"",now))
cursor.execute("INSERT INTO Parking_3(available, plate, dtime) VALUES(%s, %s, %s)",
(1,"",now))
cursor.execute("INSERT INTO Parking_4(available, plate, dtime) VALUES(%s, %s, %s)",
(1,"",now))
database.commit()

# Ultrasone setup
trigger=19
echo=13

# Motor setup
motor1=22
motor2=27
motor3=15
```

```
motor4=17
GPIO.setup(trigger,GPIO.OUT)
GPIO.setup(echo,GPIO.IN)
GPIO.setup(motor1,GPIO.OUT)
GPIO.setup(motor2,GPIO.OUT)
GPIO.setup(motor3,GPIO.OUT)
GPIO.setup(motor4,GPIO.OUT)
delay=0.002
duration=3

# Initialize SPI bus
SPI=busio.SPI(board.SCK,MOSI=board.MOSI,MISO=board.MISO)

# MCP3008 setup
CS0=digitalio.DigitalInOut(board.CE0)
MCP3008=SPIDevice(SPI,CS0,baudrate=1000000)

# LCD setup
DC=digitalio.DigitalInOut(board.D23)
CS1=digitalio.DigitalInOut(board.CE1)
RESET=digitalio.DigitalInOut(board.D24)
Display=adafruit_pcd8544.PCD8544(SPI,DC,CS1,RESET,baudrate=1000000)
Display.bias=4
Display.contrast=60
Display.invert=True
Display.fill(0)
Display.show()
font=ImageFont.load_default()

# Numberplate setup
plate=""

# Led setup
led=26
GPIO.setup(led,GPIO.OUT)

# Simplepush setup
message=0

# FUNCTIONS CREATED TO MAKE THE PROGRAM EASIER
# Reads value of chosen channel of the Analog-Digital converter
def readMCP3008(channel):
    if ((channel>7)or(channel<0)):
        return -1
    with MCP3008:
        r=bytearray(3)
        SPI.write_readinto([1,(8+channel)<<4,0],r)
        time.sleep(0.000005)
        output=((r[1]&3)<<8)+r[2]
        return output
# Read distance between ultrasone and nearest object
def ultrasone():
    # Sent pulse to trigger pin
    GPIO.output(trigger,True)
    time.sleep(0.00001)
    GPIO.output(trigger,False)
    # Wait till echo pin is high
    while GPIO.input(echo)==0:
        time.sleep(0.000001)
    starttime=time.time()
    # Wait till echo pin is low
    while GPIO.input(echo)==1:
        time.sleep(0.000001)
    stoptime=time.time()
```

```
    # Calcutate time between echo high and low
    TimeElapsed=stoptime-starttime
    # Convert to cm
    distance=TimeElapsed*17000
    return distance
# Motor does 4 steps backwards (opens barrier)
def motoropen():
    setStep(0, 0, 0, 1)
    time.sleep(delay)
    setStep(0, 0, 1, 0)
    time.sleep(delay)
    setStep(0, 1, 0, 0)
    time.sleep(delay)
    setStep(1, 0, 0, 0)
    time.sleep(delay)
# Motor does 4 steps forwards (closes barrier)
def motorclose():
    setStep(1, 0, 0, 0)
    time.sleep(delay)
    setStep(0, 1, 0, 0)
    time.sleep(delay)
    setStep(0, 0, 1, 0)
    time.sleep(delay)
    setStep(0, 0, 0, 1)
    time.sleep(delay)
# Motor does 1 steps
def setStep(w1, w2, w3, w4):
  GPIO.output(17, w1)
  GPIO.output(15, w2)
  GPIO.output(27, w3)
  GPIO.output(22, w4)
# Opens barrier and reads numberplate if necessary
def barrier(plate,available):
    # If all spots are taken don't open barrier
    if
((available[0]==False)and(available[1]==False)and(available[2]==False)and(available[3]==
=False)):
        return plate
    # If at least one spot is free read if car is in front of barrier
    else:
        if (ultrasone()<6):
            # Read plate
            plate=readplate()
            # Open barrier
            for i in range(120):
                motoropen()
            # Wait until car is gone
            while (ultrasone()<6):
                time.sleep(0.1)
            # Close barrier
            for i in range(120):
                motorclose()
        return plate

# Reads data from database
def readdata():
    # Read data from database about parking spot 1
    cursor.execute("SELECT available FROM Parking_1 ORDER BY dtime DESC LIMIT 1")
    available1=cursor.fetchall()[0][0]
    cursor.execute("SELECT plate FROM Parking_1 ORDER BY dtime DESC LIMIT 1")
    plate1=cursor.fetchall()[0][0]
    # Read data from database about parking spot 2
    cursor.execute("SELECT available FROM Parking_2 ORDER BY dtime DESC LIMIT 1")
    available2=cursor.fetchall()[0][0]
    cursor.execute("SELECT plate FROM Parking_2 ORDER BY dtime DESC LIMIT 1")
    plate2=cursor.fetchall()[0][0]
```

```
    # Read data from database about parking spot 3
    cursor.execute("SELECT available FROM Parking_3 ORDER BY dtime DESC LIMIT 1")
    available3=cursor.fetchall()[0][0]
    cursor.execute("SELECT plate FROM Parking_3 ORDER BY dtime DESC LIMIT 1")
    plate3=cursor.fetchall()[0][0]
    # Read data from database about parking spot 4
    cursor.execute("SELECT available FROM Parking_4 ORDER BY dtime DESC LIMIT 1")
    available4=cursor.fetchall()[0][0]
    cursor.execute("SELECT plate FROM Parking_4 ORDER BY dtime DESC LIMIT 1")
    plate4=cursor.fetchall()[0][0]
    available=[available1,available2,available3,available4]
    plates=[plate1,plate2,plate3,plate4]
    return available,plates

# Writes data to database
def inputdata(available,plates,now):
    cursor.execute("SELECT available FROM Parking_1 ORDER BY dtime DESC LIMIT 1")
    # Write data from parking spot 1 to database
    if (str(cursor.fetchall())[2]!=str(available[0])):
        cursor.execute("INSERT INTO Parking_1(available,plate,dtime)
VALUES(%s,%s,%s)",(available[0],plates[0],now))
        database.commit()
    cursor.execute("SELECT available FROM Parking_2 ORDER BY dtime DESC LIMIT 1")
    # Write data from parking spot 2 to database
    if (str(cursor.fetchall())[2]!=str(available[1])):
        cursor.execute("INSERT INTO Parking_2(available,plate,dtime)
VALUES(%s,%s,%s)",(available[1],plates[1],now))
        database.commit()
    cursor.execute("SELECT available FROM Parking_3 ORDER BY dtime DESC LIMIT 1")
    # Write data from parking spot 3 to database
    if (str(cursor.fetchall())[2]!=str(available[2])):
        cursor.execute("INSERT INTO Parking_3(available,plate,dtime)
VALUES(%s,%s,%s)",(available[2],plates[2],now))
        database.commit()
    cursor.execute("SELECT available FROM Parking_4 ORDER BY dtime DESC LIMIT 1")
    # Write data from parking spot 4 to database
    if (str(cursor.fetchall())[2]!=str(available[3])):
        cursor.execute("INSERT INTO Parking_4(available,plate,dtime)
VALUES(%s,%s,%s)",(available[3],plates[3],now))
        database.commit()

# Takes pictures and analyses the license plate and returns this value
def readplate():
    plate=""
    while (plate==""):
        # Take picture
        os.popen('fswebcam -r 1280x720 --no-banner /home/pi/Pictures/plate.jpg')
        time.sleep(1)
        # Analyse the picture
        with open('/home/pi/Pictures/plate.jpg','rb')as fp:
            response=requests.post(
            'https://api.platerecognizer.com/v1/plate-reader/',
            data=dict(regions=['be']),
            files=dict(upload=fp),
            headers={'Authorization':'Token e591c6fc970cf088b555daa6adb78d35179feba8'})
        if (response.json()['results']):
            plate=response.json()['results'][0]['plate']
        else:
            time.sleep(1)
    return plate

# Shows the right information on the LCD-display
def lcd(available,plates):
    image=Image.new('1',(Display.width,Display.height))
```

```
    draw=ImageDraw.Draw(image)
    draw.rectangle((0,0,Display.width,Display.height),outline=255,fill=255)
    if available[0]==0:
        draw.text((1,0),('P1: '+plates[0]),font=font)
    else:
        draw.text((1,0),'P1: Vrij',font=font)
    if available[1]==0:
        draw.text((1,8),('P2: '+plates[1]),font=font)
    else:
        draw.text((1,8),'P2: Vrij',font=font)
    if available[2]==0:
        draw.text((1,16),('P3: '+plates[2]),font=font)
    else:
        draw.text((1,16),'P3: Vrij',font=font)
    if available[3] == 0:
        draw.text((1,24),('P4: '+plates[3]),font=font)
    else:
        draw.text((1,24),'P4: Vrij',font=font)
    availability=available[0]+available[1]+available[2]+available[3]
    if availability==0:
        draw.text((1,32),'Vol',font=font)
    elif availability==1:
        draw.text((1,32),'1P Vrij',font=font)
    elif availability==2:
        draw.text((1,32),'2P Vrij',font=font)
    elif availability==3:
        draw.text((1,32),'3P Vrij',font=font)
    elif availability==4:
        draw.text((1,32),'4P Vrij',font=font)
    else:
        draw.text((1,32),'ERROR',font=font)
    Display.image(image)
    Display.show()

# Determines howmany parking spots are still available and returns this value
def spotsavailable(available,plates,plate):
    # Read which spots are available
    for i in range(4):
        if (round(readMCP3008(i)*100/1023)<25):
            if (available[i]==True):
                plates[i]=plate
                available[i]=0
        else:
            if (available[i]==False):
                plates[i]=""
                available[i]=1
    # If all spots are taken send message to owner and turn on the LED
    if
((available[0]==False)and(available[1]==False)and(available[2]==False)and(available[3]=
=False)):
        message=+1
        if (message==1):
            send('QTgtTt','Parking Lot','The parking lot is full','Vibrate')
        GPIO.output(led,1)
    elif
((available[0]==True)or(available[1]==True)or(available[2]==True)or(available[3]==True)
):
        message=0
        GPIO.output(led,0)
    return available,plates

# MAIN PROGRAM
try:
    while True:
        # Read data from database
        data=readdata()
```

```
            # Detect car, read and analyses licenceplate
            plate=barrier(plate,data[0])
            # Determinate which spots are available
            data=spotsavailable(data[0],data[1],plate)
            # Show data on LCD-display
            lcd(data[0],data[1])
            # Write data to database
            inputdata(data[0],data[1],now)
    except KeyboardInterrupt:
        # Cleaning LCD
        Display.fill(0)
        Display.show()

        # Cleaning GPIO-pins
        GPIO.cleanup()

        # ENDING THE PROGRAM

        print("Program ended")
```

1.7.2 web interface

```
#!/usr/bin/env python
# library import
import cgitb ; cgitb.enable()
import MySQLdb
import datetime

# Connect with the database
database = MySQLdb.connect(host="localhost", user="pi", passwd="raspberry",
db="Project_IOT")

#Select the database
cursor= database.cursor()

# web page
print("Content-Type: text/html")
print("""
<html>
<head>
    <meta charset="UTF-8">
    <title>Parkingplace</title>
    <meta http-equiv="refresh" content="1">
</head>
<body>
<h1>Parkingplace Information</h1>
<svg width="375" height="175">
    <rect x='0' y='0' width='375' height='175' style='fill: gray' />"
""")

# check vaulues from parking 3
cursor.execute("SELECT available, plate, dtime FROM Parking_3 ORDER BY dtime
DESC LIMIT 1")
values = cursor.fetchall()
for value in values:
    p3 = value[0]
    if value[0] == 1:
        print("<rect x='0' y='0' width='175' height='75' style='fill:
limegreen' />")
    else:
        print("<rect x='0' y='0' width='175' height='75' style='fill: red'
/>")
        print("<text x='10' y='55' font-family='Verdana' font-size='40'
fill='black'> %s </text>" % (value[1]))
        st_p3 = value[2]


# check vaulues from parking 2
cursor.execute("SELECT available, plate, dtime FROM Parking_2 ORDER BY dtime
DESC LIMIT 1")
values = cursor.fetchall()
for value in values:
    p2 = value[0]
    if value[0] == 1:
        print("<rect x='200' y='0' width='175' height='75' style='fill:
limegreen' />")
    else:
        print("<rect x='200' y='0' width='175' height='75' style='fill: red'
/>")
```

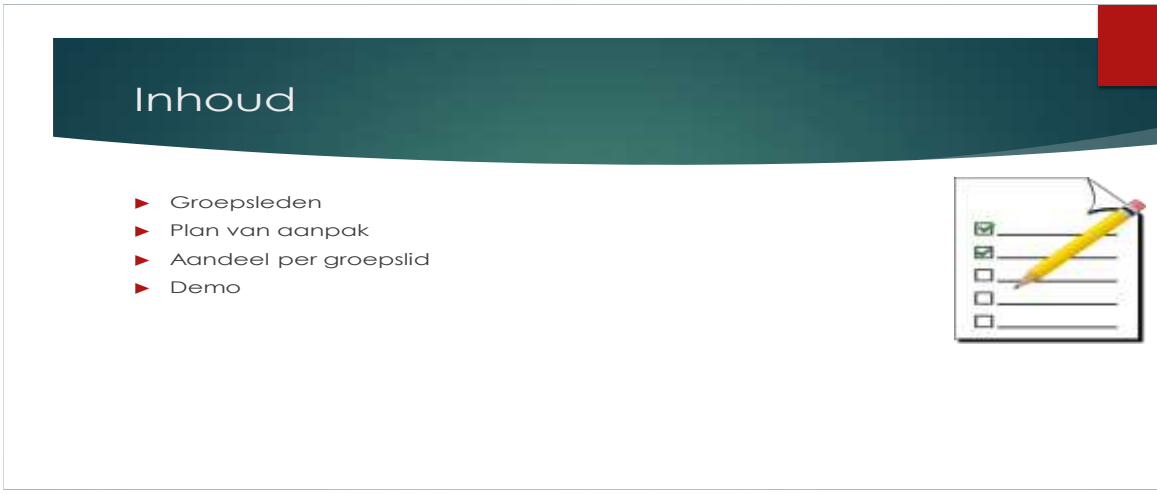
```
        print("<text x='210' y='55' font-family='Verdana' font-size='40'
fill='black'> %s </text>" % (value[1]))
        st_p2 = value[2]

# check vaulues from parking 4
cursor.execute("SELECT available, plate, dtime FROM Parking_4 ORDER BY dtime
DESC LIMIT 1")
values = cursor.fetchall()
for value in values:
    p4 = value[0]
    if value[0] == 1:
        print("<rect x='0' y='100' width='175' height='75' style='fill:
limegreen' />")
    else:
        print("<rect x='0' y='100' width='175' height='75' style='fill: red'
/>")
        print("<text x='10' y='150' font-family='Verdana' font-size='40'
fill='black'> %s </text>" % (value[1]))
        st_p4 = value[2]

# check vaulues from parking 1
cursor.execute("SELECT available, plate, dtime FROM Parking_1 ORDER BY dtime
DESC LIMIT 1")
values = cursor.fetchall()
for value in values:
    p1 = value[0]
    if value[0] == 1:
        print("<rect x='200' y='100' width='175' height='75' style='fill:
limegreen' />")
    else:
        print("<rect x='200' y='100' width='175' height='75' style='fill:
red' />")
        print("<text x='210' y='150' font-family='Verdana' font-size='40'
fill='black'> %s </text>" % (value[1]))
        st_p1 = value[2]

print("""
</svg>
<br>
<br>
<br>
<h2>Parking Fee</h2>
<p>The first minute is free, after that you pay 5 euros per minute. </p>
""")

# Calculate and display parink fee for parking 1
if p1 == 0:
    time_p1 = str(datetime.datetime.now() - st_p1)[:-7]
    s1 = time_p1[-2:]
    m1 = time_p1[-5: -3]
    h1 = time_p1[0]
    if m1 == "00" and h1 == "0":
        fee1 = "FREE"
    else:
        fee1 = str((int(h1) * 300) + int(m1) * 5) + "euro"
    print("<p>Parkingplace 1 is taken for %s hours, %s minutes and %s
seconds. The parking fee is %s!</p>" % (h1, m1, s1, fee1))
```

```
# Calculate and display parink fee for parking 2
if p2 == 0:
    time_p2 = str(datetime.datetime.now() - st_p2)[:-7]
    s2 = time_p2[-2:]
    m2 = time_p2[-5: -3]
    h2 = time_p2[0]
    if m2 == "00" and h2 == "0":
        fee2 = "FREE"
    else:
        fee2 = str((int(h2) * 300) + int(m2) * 5) + "euro"
    print("<p>Parkingplace 2 is taken for %s hours, %s minutes and %s
seconds. The parking fee is %s!</p>" % (h2, m2, s2, fee2))

# Calculate and display parink fee for parking 3
if p3 == 0:
    time_p3 = str(datetime.datetime.now() - st_p3)[:-7]
    s3 = time_p3[-2:]
    m3 = time_p3[-5: -3]
    h3 = time_p3[0]
    if m3 == "00" and h3 == "0":
        fee3 = "FREE"
    else:
        fee3 = str((int(h3) * 300) + int(m3) * 5) + "euro"
    print("<p>Parkingplace 3 is taken for %s hours, %s minutes and %s
seconds. The parking fee is %s!</p>" % (h3, m3, s3, fee3))

# Calculate and display parink fee for parking 4
if p4 == 0:
    time_p4 = str(datetime.datetime.now() - st_p4)[:-7]
    s4 = time_p4[-2:]
    m4 = time_p4[-5: -3]
    h4 = time_p4[0]
    if m4 == "00" and h4 == "0":
        fee4 = "FREE"
    else:
        fee4 = str((int(h4) * 300) + int(m4) * 5) + "euro"
    print("<p>Parkingplace 4 is taken for %s hours, %s minutes and %s
seconds. The parking fee is %s!</p>" % (h4, m4, s4, fee4))

# display information when parking is full/free
if p1 == p2 == p3 == p4 :
    if p1 == 0:
        print("<p>All parkingplace's are currently taken!</p>")
    else:
        print("<p>All parkingplace's are currently free!</p>")

print("""
</body>
</html>
""")
```

2 Presentatie

Slide 1

Slide 2



Slide 3

Slide 4

Plan van aanpak

- 29/10: Taken verdeling
- 12/11: Afspreken deadlines
- 19/11: Deadline individueel deel
- 26/11: Code samenvoegen + troubleshooting
- 30/11: Filmen + documentatie

Slide 5

Aandeel per groepslid

Rutger Stessens:
→Slagboom + Ultrasoon
Wim Hembrechts:
→Analyseren van nummerplaat
→Push berichten
Tristan Berkmans:
→Visualisatie op Lcd
→Maquette bouwen
Timo Van Litsenborg:
→Webinterface + Database

Slide 6

Demo

3 Besluit

3.1 Project beschrijving

We vonden het een heel leerrijk project. We hebben allemaal ons steentje bij gedragen en allemaal veel bijgeleerd. We zijn er in geslaagd om een optimale taakverdeling te maken en iedereen van onze groep heeft zich aan de afspraken gehouden. We hebben elkaar ook verdergeholpen indien er zich een probleem voordeed.

We zijn bijzonder fier op het resultaat namelijk een parking die weet hoeveel parkeerplaatsen bezet zijn en hoeveel er nog vrij zijn. Dit doen we door gebruik te maken van vier lichtsensors. Als de parkeerplaats vol staat, krijgt de eigenaar een melding op zijn gsm en aan de poort gaat er een lampje branden. Bij elke parkeerplaats waarop een auto staat, verschijnt de nummerplaat op de display. Dit zowel op de webinterface als op het lcd-scherm in de garage.

3.2 Evaluatie

Opdracht	Werkt	Status
Werkende slagboom met utrasone sensor	10%	Actief
Werkende analyse van de nummer plaat	20%	Actief
Correct visualiseren op de display met data uit de display	10%	Actief
Correcte werking webinterface	10%	Actief
Verstuur bericht wanneer de 4 parkeerplaatsen bezet zijn	10%	Actief
Professionele presentatie en demonstratie - Voorstelling van het team - Plan van aanpak - Aandeel per groepslid en takenverdeling - Mooi werkend schaal model met bespreking van de bovenstaande criteria	20%	In progress
Creatieve extra's (betaaltimer)	10%	Actief
Upload project bundel (15 pages) met youtube link naar video (2 min.)	10%	Done