Project Parkingplace

IoT Advanced

Boons Jordy CCS01 r0853681
Latifine Ilias CCS01 r0847060
Liekens Jentel CCS01 r0835618
Van den Bergh Sam CCS02 r0751478
Volz Cody CCS01 r0766008

CAMPUS
Geel

Cloud & Cyber Security

Academic year 2021-2022

Inhoudstabel

Content

1 Het project

1.1 Projectbeschrijving

Het is de bedoeling om een functionele parkeerplaats te maken. Deze parkeerplaats moet werken met automatische nummerplaat-herkenning en visualisatie voor de gebruikers en de eigenaars. Er moet ook een werkende slagboom aanwezig zijn. Voor de eigenaars moet er een web interface zijn en er moet een bericht gestuurd worden via sms als de 4 plaatsen zijn ingenomen. Voor de gebruikers van de parkeerplaats wordt er gewerkt met een LCD- scherm dat weergeeft hoeveel plaatsen er nog beschikbaar zijn.

1.2 Werking van de parking

De werking van de parkeerplaats verloopt als volgt: Als eerste stopt de auto voor een gesloten bareel die beschikt over een ultrasone sensor. De bestuurder kan hier op een LCD-scherm kijken hoeveel plaatsen er nog beschikbaar zijn. Als er geen plaatsen meer beschikbaar zijn zal de bareel gesloten blijven en zal er een led beginnen te branden. Als er nog plaatsen vrij zijn, zal de bareel open gaan en wordt er een foto van de nummerplaat genomen. Van deze foto zal de nummerplaat worden ingelezen d.m.v. AI-technologie, waarna deze wordt toegevoegd aan een database. De auto kan na het openen van de slagboom verder rijden en parkeren op een vrije plaats. Het is ook de bedoeling dat iedere parkeerplaats beschikt over een LDR-sensor. Deze zal meten of de parkeerplaats in kwestie bezet is, of niet. Na alle vorige stappen te doorlopen, wordt het LCD-scherm aan de ingang bijgewerkt net zoals de remote web interface. Wanneer dat de vier plaatsen in de parkeerplaats zijn ingenomen zal er een push message in de vorm van een sms naar de eigenaar verstuurd worden.

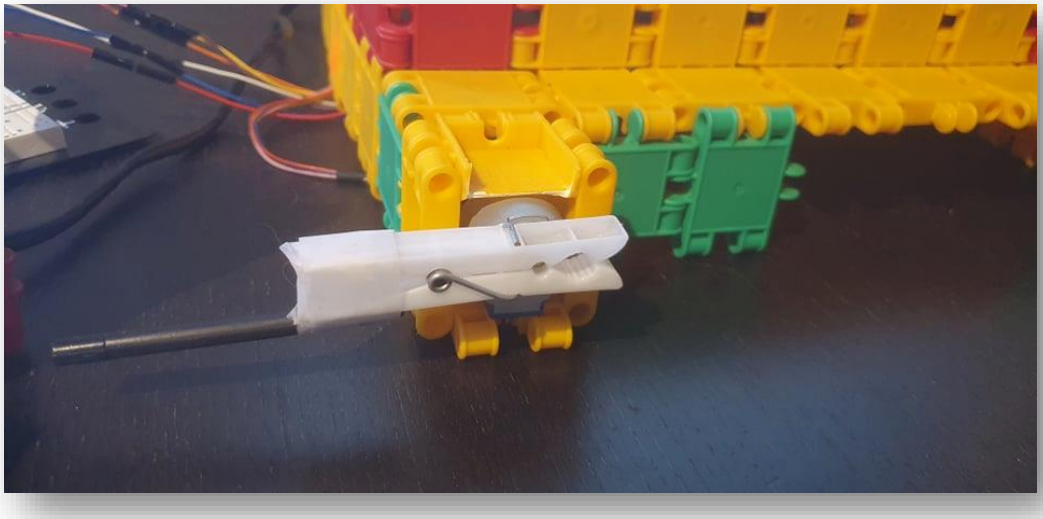
2 Componenten

In onderstaande punten worden alle componenten die er in dit project gebruikt worden, overlopen.

2.1 Stappenmotor

In dit project werken wij met de stappen motor “28BYJ-48 5V”. Deze stappen motor dient om de slagboom aan te sturen. Er is een wasknijper met een klein stokje aan bevestigt zodat dit effectief als slagboom kan dienen. Op de afbeelding hiernaast kan je zien hoe deze eruitziet. Op de afbeelding op de volgende pagina zie je hoe dit is geïntegreerd.

Figuur 1: Stappenmotor

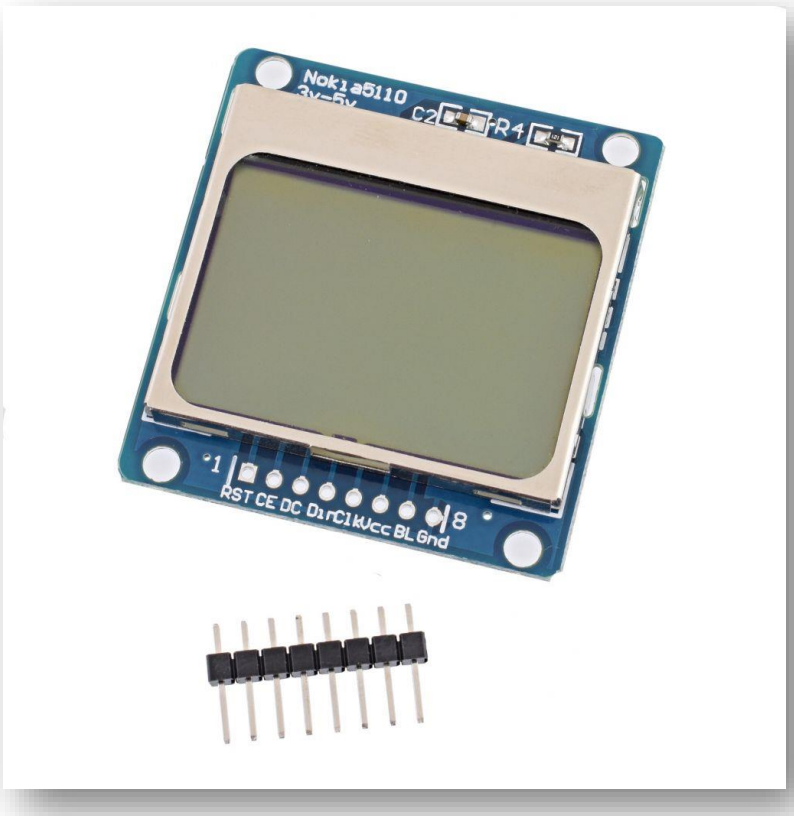


Figuur 2: Slagboom

```
motor_pins = [19, 13, 6, 26]

halfstep_seq = [
  [1,0,0,0],
  [1,1,0,0],
  [0,1,0,0],
  [0,1,1,0],
  [0,0,1,0],
  [0,0,1,1],
  [0,0,0,1],
  [1,0,0,1]
]

def slagboom():
  #Open
  for i in range(128):
    for halfstep in range(8):
      for pin in range(4):
        GPIO.output(motor_pins[pin],
halfstep_seq[halfstep][pin])
      time.sleep(0.001)
  time.sleep(3)
  #Close
  for i in range(128):
    for halfstep in range(8):
      for pin in range(4):
        GPIO.output(motor_pins[pin], halfstep_seq[7-
halfstep][pin])
      time.sleep(0.001)
```

2.2 LCD

De LCD wordt gebruikt om aan de bestuurder van de wagen te tonen welke plaatsen er nog vrij zijn op de parking. Het component dat we hiervoor gebruikt hebben is de Nokia LCD 5110.



Figuur 3: Nokia LCD 5110

```
def lcd():
  disp.begin(contrast=60)
  disp.clear()
  disp.display()
  image = Image.new('1', (LCD.LCDWIDTH, LCD.LCDHEIGHT))
  draw = ImageDraw.Draw(image)
  draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT),
outline=255, fill=255)
  font = ImageFont.load_default()
  return draw, image, font

def write_lcd():
  disp.clear()
  disp.display()
  image = Image.new('1', (LCD.LCDWIDTH, LCD.LCDHEIGHT))
  font = ImageFont.load_default()

  draw = ImageDraw.Draw(image)
```

```
  draw.rectangle((0,0,LCD.LCDWIDTH,LCD.LCDHEIGHT),
outline=255, fill=255)

  draw.text((1,8), 'P.1: bezet' if parkingSpots[0] else 'P.1:
vrij', font=font)
  draw.text((1,16), 'P.2: bezet' if parkingSpots[1] else
'P.2: vrij', font=font)
  draw.text((1,24), 'P.3: bezet' if parkingSpots[2] else
'P.3: vrij', font=font)
  draw.text((1,32), 'P.4: bezet' if parkingSpots[3] else
'P.4: vrij', font=font)

  disp.image(image)
  disp.display()
```

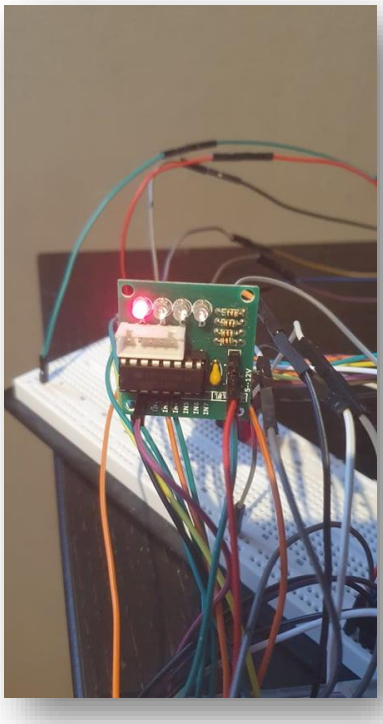
Telkens wanneer de LDR's gecontroleerd worden, zal de LCD zich aanpassen om de juiste parkeerplaatsen te tonen die nog vrij zijn.

Figuur 4: Display

2.3 Led

Wanneer er geen plaats meer vrij is op de parking, gaat het eerste lampje van de stappenmotor branden. We gebruiken hiervoor de LED module ULN2003. De module kan je zien op onderstaande afbeeldingen. Op de eerste afbeelding zie je de ULN2003 module en op de volgende pagina zie hoe deze in praktijk gebruikt wordt.

Figuur 5: ULN2003



Figuur 6: Led module in praktijk

```
if np.all(parkingSpots):
      GPIO.output(LED, 1)
    else:
      GPIO.output(LED, 0)
```

Wegens gebrek aan een ledje, laten we het eerste lampje van de ULN2003 branden.

2.4 LDR

De LDR oftewel Light-Dependent Resistor is een lichtgevoelige weerstand die is ingebouwd op elke parkeerplaats. Deze lichtgevoelige weerstand maakt het mogelijk om te kijken welke plaatsen er nog beschikbaar zijn. Wanneer er een wagen zich parkeert op één van de 4 plaatsen zal de LDR een andere waarde meten dan als er geen wagen staat. Op de afbeelding hiernaast zie je hoe deze lichtgevoelige weerstand er uitziet en op de afbeelding hieronder hoe deze in het project is geïntegreerd.

Figuur 7: LDR5116

Figuur 8: Parkeerplaatsen

```
def ldr():
  for i, parking in enumerate(parkingSpots):
    waarde = read_spi(i + 1)
    print(waarde)
    if waarde < ldr_value:
      parkingSpots[i] = True
    else:
```

```
      parkingSpots[i] = False
    if parkingSpots[i] != oldParkingSpots[i]:
      write_lcd()
      pubnub.publish().channel('parkingSpots').message(str(i)
+ '|' + str(parkingSpots[i]).lower()).sync()
    if np.all(parkingSpots):
      GPIO.output(LED, 1)
    else:
      GPIO.output(LED, 0)
    oldParkingSpots[i] = parkingSpots[i]
```

We bekijken de waarde van elke LDR, op basis hiervan kunnen we besluiten of er een wagen geparkeerd staat of niet. De parkingSpots array zal aangepast worden naar de actuele waarden.

2.5 Ultrasone Sensor

Om te controleren of er een voertuig voor de slagboom staat hebben wij een ultrasone sensor gebruikt. Door de afstand te meten kunnen we afleiden of er een voertuig staat of niet. Wanneer dit het geval is zal zijn nummerplaat gescand worden, slagboom opengaan…

Figuur 9: HC-SR04

Figuur 10: Ultrasone sensor in praktijk

```
def distance():
    GPIO.output(GPIO_TRIGGER, True)

    time.sleep(0.00001)
    GPIO.output(GPIO_TRIGGER, False)

    StartTime = time.time()
    StopTime = time.time()

    while GPIO.input(GPIO_ECHO) == 0:
        StartTime = time.time()

    while GPIO.input(GPIO_ECHO) == 1:
        StopTime = time.time()

    TimeElapsed = StopTime - StartTime
    distance = (TimeElapsed * 34300) / 2
    return distance

def push_message(message):
```

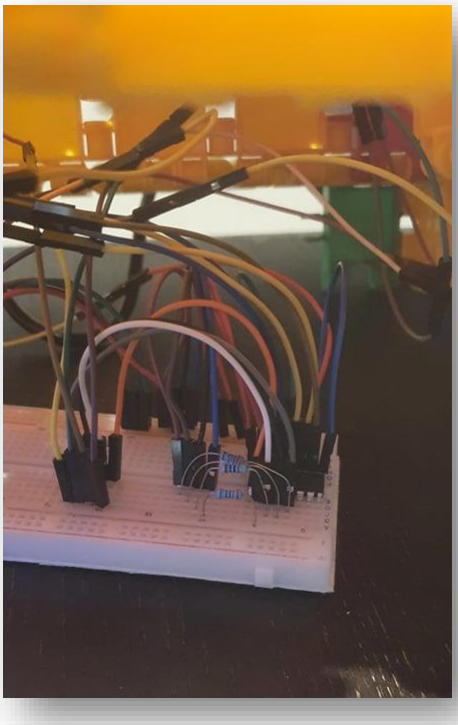
```
  print('send message')
  responseData = sms.send_message({
    "from": "Parking checker",
    "to": "NUMMER",
    "text": message,
  })
  if responseData["messages"][0]["status"] == "0":
    print("Message sent successfully.")
  else:
    print("Message failed with error: " +
responseData['messages'][0]['error-text'])

def parking_check(nummerplaat):
  cursor.execute("SELECT * FROM on_premise ORDER BY id")
  data = cursor.fetchall()

  license_plate = nummerplaat
  action = 'Entering'
  time = datetime.now()
  vehicle = ''

  for vehicle in data:
      if license_plate == vehicle[2]:
          action = 'Leaving'
          vehicle = vehicle
  if action == 'Entering':
    if np.all(parkingSpots):
      print('All parking spots are occupied')
      push_message('All parking spots are occupied')
    else:
      cursor.execute("INSERT INTO on_premise(timestamp,
license_plate) VALUES(%s, %s)", (time, license_plate))
      cursor.execute("INSERT INTO vehicles(timestamp,
license_plate, action, time) VALUES(%s, %s, %s, %s)", (time,
license_plate, action, 0))
      slagboom()
  if action == 'Leaving':
      cursor.execute("DELETE FROM on_premise WHERE id = %s",
(str(vehicle[0])))
      cursor.execute("INSERT INTO vehicles(timestamp,
license_plate, action, time) VALUES(%s, %s, %s, %s)", (time,
license_plate, action, (time - vehicle[1]).total_seconds()))
      slagboom()
  database.commit()
  pubnub.publish().channel('entrance').message('reload').sync
()
```

We controleren of er een wagen voor de slagboom staat d.m.v. de tijd tussen start en aankomst signaal van de ultrasoon te berekenen. Hierna zal de parking check plaatsvinden. Deze checkt of de wagen binnen of buiten rijdt en of er nog een plaats vrij is. Indien er plaats is gaat de slagboom open en wordt de wagen geregistreerd in de database. Wanneer er geen plaats meer vrij is zal er een sms bericht gestuurd worden naar de nummer van de verantwoordelijke en zal de slagboom niet open gaan.

2.6 Weerstandjes

Deze weerstanden zorgen ervoor dat de stroom die door de LDR loopt niet te hoog is waardoor deze niet stuk zullen gaan. Op de eerste onderstaande afbeelding zie je hoe deze weerstandjes eruit zien en op de eerstvolgende hoe deze in praktijk zijn gebruikt.

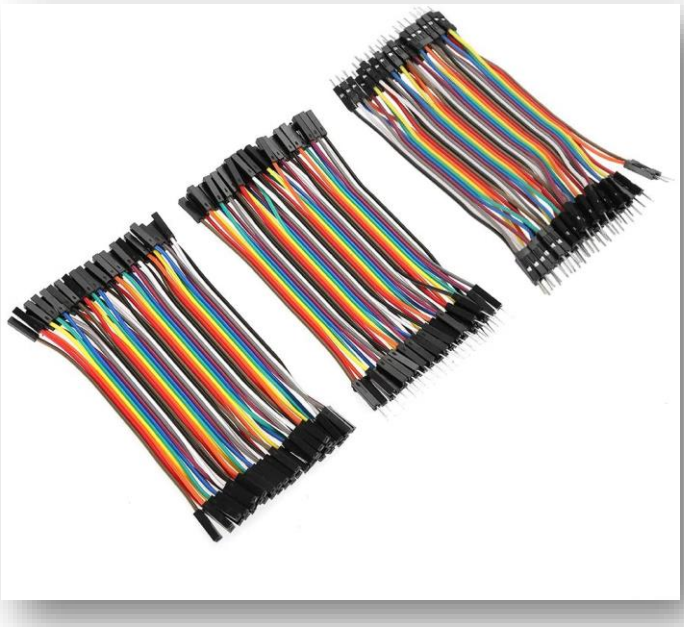
Figuur 11: Weerstandjes



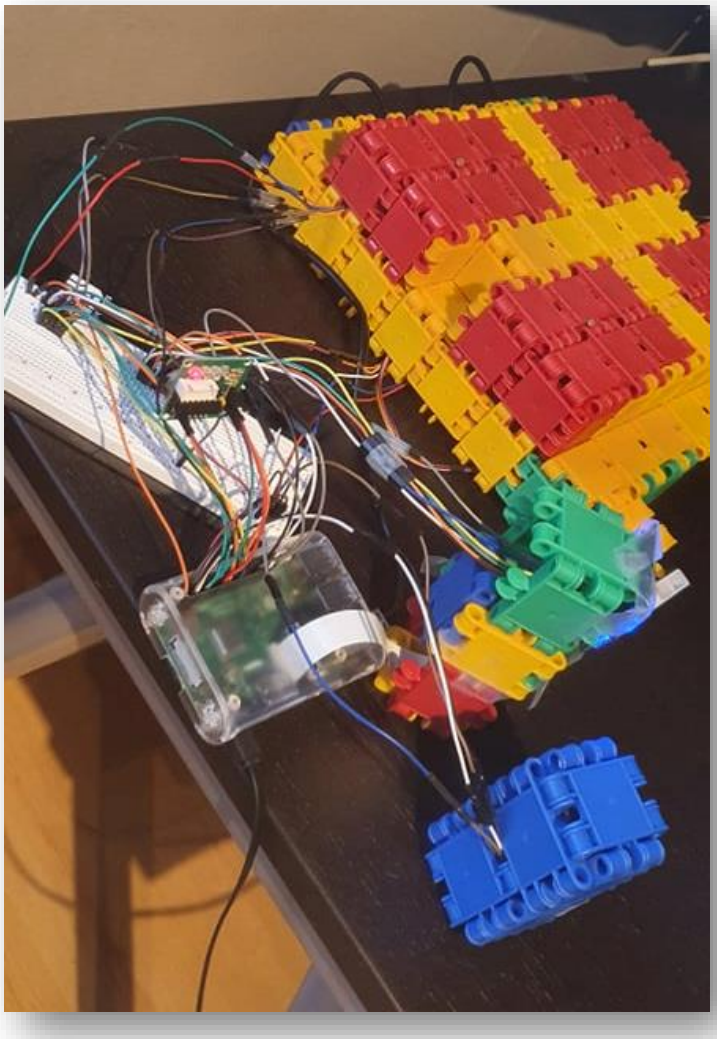
Figuur 12: Weerstandjes in de praktijk

2.7 Bekabeling

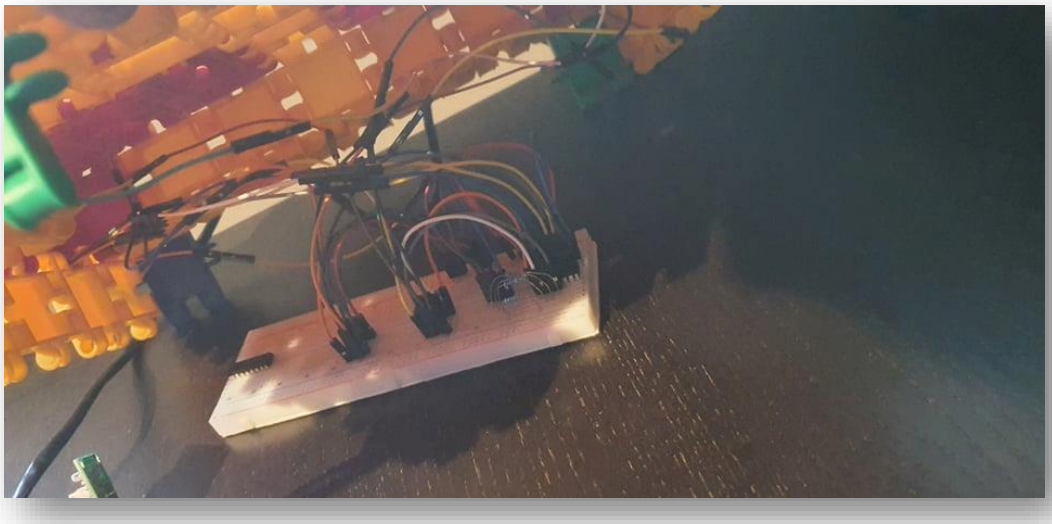
We hebben F(emale)-M(ale), M-M, F-F kabels gebruikt om alles met elkaar te verbinden. Je kan een afbeelding met de gebruikte kabels op onderstaande afbeelding zien. Op de twee afbeeldingen daaronder kan je kijken hoe dit er in de praktijk uitziet.



Figuur 13: Bekabeling



Figuur 14: Zichtbare bekabeling

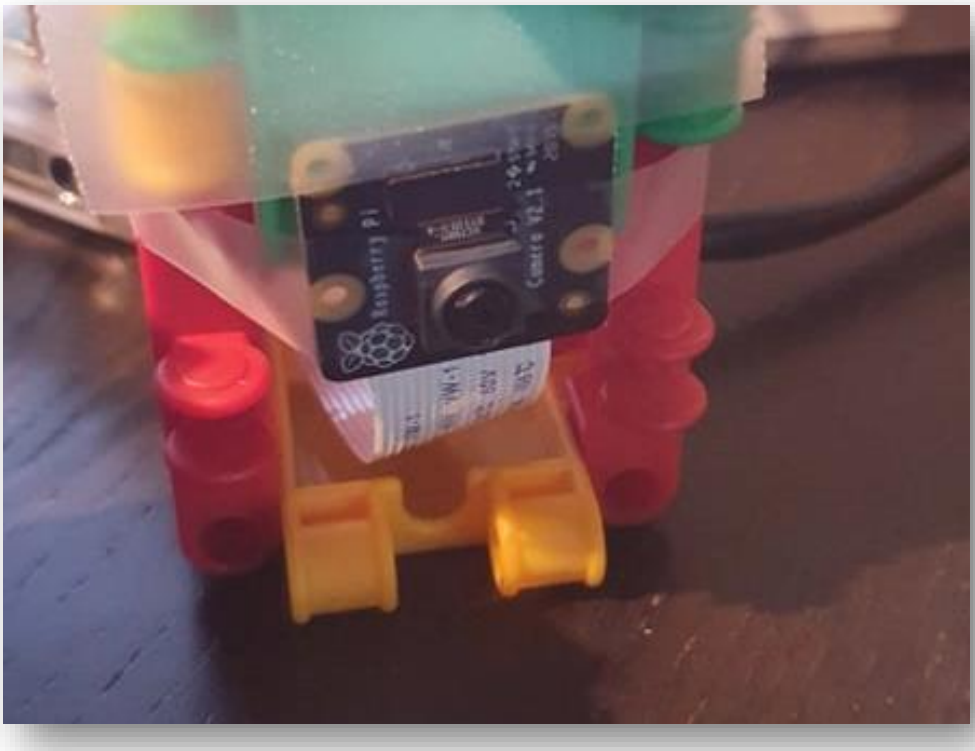


Figuur 15: Verborgen bekabeling

2.8 Pi-Camera module V2(Black)

Om de beelden van de nummerplaten vast te leggen hebben we een PI-camera module V2 gebruikt. Deze beschikt over een 8MP lens met de mogelijkheid om infrarood foto's te maken. De camera kan verbonden worden op de PI via de CSi interface. Op onderstaande afbeelding kan je zien hoe deze camera eruit ziet en op de afbeelding eronder kan je zien hoe we deze in de praktijk hebben geïntrigeerd.

Figuur 16: Pi-Camera



Figuur 17: Camera in de praktijk

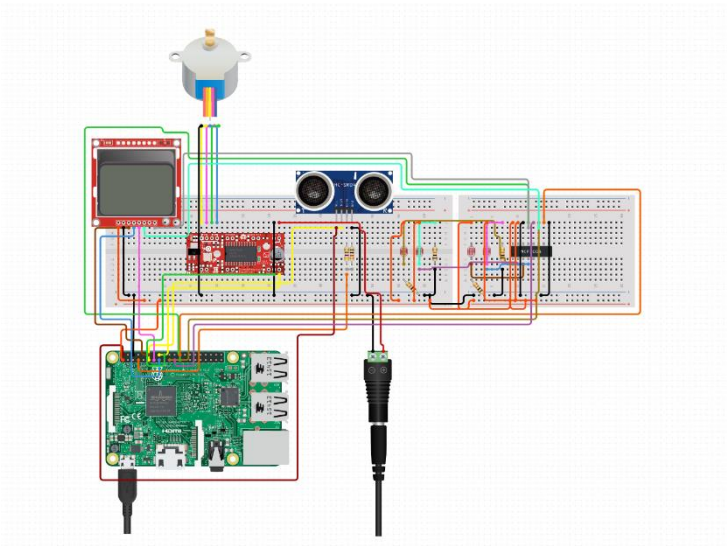
```
def nummerplaat():
    time.sleep(2)

    file_name = "/home/pi/Pictures/img_" + str(time.time()) +
".jpg"
    camera.capture(file_name)


    regions = ['be', 'nl']
    with open(file_name, 'rb') as fp:
        response = requests.post(
            'https://api.platerecognizer.com/v1/plate-
reader/',
            data=dict(regions=regions),
            files=dict(upload=fp),
            headers={'Authorization': 'Token X'})
    if response.json()['results'] != []:
      nummerplaat = response.json()['results'][0]['plate']
      print(nummerplaat)
      parking_check(nummerplaat)
    else:
      print('License Plate not found')
```

De camera neemt een foto van de wagen die voor de slagboom staat. Deze foto sturen we door naar een API, deze geeft ons de nodige informatie in JSON-formaat terug. Hieruit halen we dan de nummerplaat. Indien de nummerplaat niet herkend wordt, zal hier een melding van gegeven worden.

3 Opstelling

3.1 Elektrisch Schema

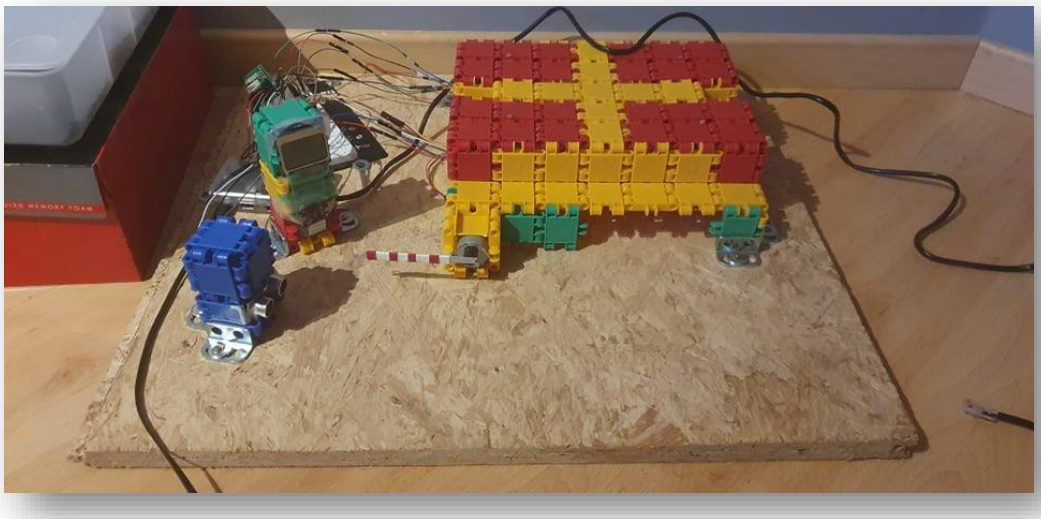
Op de afbeelding hiernaast kan je zien hoe al de componenten uit de vorige stappen worden gecombineerd in de praktijk. Dit wordt weergegeven aan de hand van een elektrisch schema en is gemaakt met behulp van circuit.io. De camera staat hier niet op vermeld, maar deze wordt verbonden met de CSi interface op de PI.



Figuur 18: Elektrisch schema

3.2 Fysieke Opstelling

Op onderstaande afbeelding kan je zien hoe dat het elektrisch schema er in de praktijk uit ziet.



Figuur 19: Opstelling

4 Code

Hieronder wordt de verschillende aspecten van de code opgesomd die voor het functioneren van het programma zorgen.

Smart Parking Web

Voor de web versie hebben we gebruikt gemaakt van het javascript framework Vue Js. Dit wordt momenteel ook gehost door middel van Firebase. Aangezien we met firebase aan het werken waren, hebben we ook gebruik gemaakt van de authenticatie module die Firebase zelf bevat.

Github repository: https://github.com/r0766008/smart-parking-web

Smart Parking API

Om een link te leggen tussen de database en de webversie hebben we een aparte API gemaakt die hier voor zorgt. Dit is gemaakt met behulp van node js, samen met het express framework. Dit wordt momenteel ook gehost door middel van Heroku.

Github repository: https://github.com/r0766008/smart-parking-api

Smart Parking Python code

Een overzicht van alle code die gebruikt wordt om het project te laten draaien is terug te vinden in deze Github repository. Alle keys en tokens zijn weggelaten.

Github repository: https://github.com/r0751478/smart-parking-python

5 Demonstratie

Wij hebben dit project uiteraard ook werkende gekregen. Deze werking wordt gedemonstreerd aan de hand van een kort filmpje. Dit filmpje kan je op onderstaande URL bekijken.

URL: https://www.youtube.com/watch?v=1B4nbw4Ofgc

6 Besluit

De conclusie die we kunnen vormen is dat we er in geslaagd zijn om alle eisen van de klant in ons project te kunnen verwerken. Zo zijn we er in geslaagd om een parking te creëren met een werkende bareel met ultrasone sensor. Ook is er een LCD aanwezig die dat weergeeft welke plaatsen er beschikbaar en welke er ingenomen zijn. Indien dat deze allemaal zijn ingenomen zal er een lampje gaan branden aan de ingang. Voor de beheerder is er ook een web interface om alle veranderingen van nabij op te volgen. Hij zal ook een sms ontvangen indien dat alle plaatsen zijn ingenomen. Of een plaats al dan niet is ingenomen kan men zien aan de hand van een LDR die op de parkeerplaats is bevestigd. Al deze handelingen worden ook bewezen in bovenstaande demonstratie.

7 Lijst van afbeeldingen

8 Bronnen

https://opencircuit.be/product/28BYJ-48-5V-stepper-motor-4-phase-5-wire

https://www.amazon.in/SunRobotics-Photoresistor-Sensitive-Resistor-Dependent/dp/B081JD7G8H

https://www.gmelectronic.com/graphical-lcd-display-84x84-nokia-5110-spi-blue-backlight

https://www.antratek.be/hc-sr04-ultrasonic-sonar-distance-sensor

https://www.elektor.nl/geekcreit-5pcs-5v-stepper-motor-with-uln2003-driver-board-dupont-cable

https://nl.wikipedia.org/wiki/Weerstand_(component)

https://www.amazon.nl/Akozon-kleurrijke-Male-Male-stekkerbruggen-Breadboard/dp/B07GJLCGG8/ref=asc_df_B07GJLCGG8/?tag=nlshogostdde-21&linkCode=df0&hvadid=494673549046&hvpos=&hvnetw=g&hvrand=26747366632535804 82&hvpone=&hvptwo=&hvqmt=&hvdev=c&hvdvcmdl=&hvlocint=&hvlocphy=9040074&hvtargid=pla-557319319615&psc=1