Electronische muizeval

IoT project

Bachelor in toegepaste informatica
Afstudeerrichting: 3WT Application Development

Naam : Joris Van Puyenbroeck

Academiejaar 2021-2022

Campus : Geel

INHOUDSOPGAVE

INLEIDING

Een muizeval die electronisch gestuurd sluit afhankelijk van sensor input, dat was de opdracht. Het was het begin van een leuke uitdaging waaraan ik met plezier veel tijd heb gespendeerd.

Je leert doorheen deze opleiding je sterke en minder sterke kanten kennen en IoT, op de grens tussen soft- en hardware, is echt wel mijn ding.

We overlopen eerst de gevraagde functionaliteiten met foto's en bespreken telkens de code die voor dat stukje relevant is.

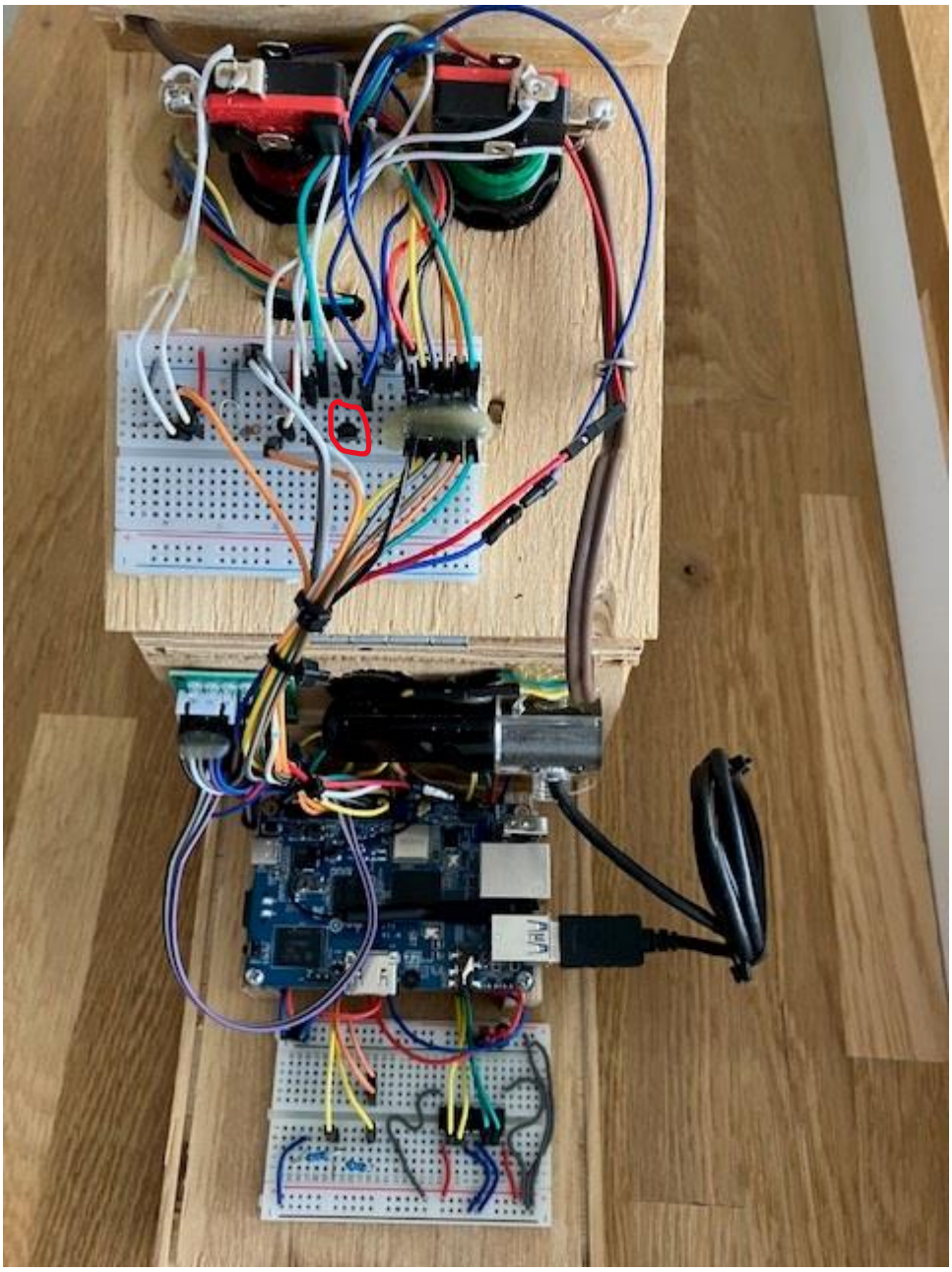
Daarna volgt een zelfevaluatie en de link naar het youtube-filmpje.

In bijlage is de volledige code van het main script te vinden. De code van de functies per onderdeel staan uitgelegd bij de basis- en optionele functies.

1 BASISFUNCTIONALITEIT

1.1 A really nice setup (2)

Het moeilijkste was te bepalen hoe de deur mechanisch snel genoeg zou sluiten zonder dat de muis terug zou kunnen ontsnappen. Het ontwerp waarop dit gebaseerd is (https://www.youtube.com/watch?v=2MAMy1cTbcA), gebruikt een stepper motor om de klep gewoon te laten vallen. Echter dan is de val nog niet volledig automatisch; dan moet de gebruiker ze ook nog steeds openen. Vandaar de keuze voor een DC motor. Maar het nadeel van een DC motor is dat je niet precies de positie kunt bepalen. De oplossing zou zijn dat de DC motor actief blijft en de poort actief openhoudt. Dit is in productie verre van ideaal omdat het energie kost. Maar op die manier had ik een oplossing voor het slot (de stepper motor) die heel precies kan sluiten, en de DC motor, die heel snel kan openen en dichtlaten vallen. Een magneetje om de deur open te houden was de kers op de taart geweest, maar ondertussen zat daar al de 12v lamp die ik wou schakelen als de muis gevangen is. De val is uitgevoerd in grenen multiplex, met plexiglas ramen.



Alle electronica zit netjes in een apart kastje. Er waren geen pins meer vrij op de orange pi. Ik gebruikte een oude webcam van Creative labs en arcade buttons van Chinese makelij. Hot glue werd veelvuldig aangebracht, en naar goede Vlaamse gewoonte ook wat Tec7. De orange pi is demonteerbaar. Cable management kostte me een uurtje ongeveer.

1.2 Stepper locks the door (2)

Twee matrices bepalen de voorwaarts -of achterwaarts beweging van de stepper motor. Het is een wave-patroon, dat afgegaan wordt in een dubbele loop. Wanneer in de matrix een 1 staat, zal de Orange Pi de respectievelijke stepper output pins (3,4,6 en 9) hoog zetten. Voorwaarts bewegen (matrix of sequentie 'fw') wordt gebruikt in de 'lock' functie en Achterwaarts ('rv') in de 'unlock' functie.

```
import time
import wiringpi as wp

wait_time = 0.002
step_pins = [3, 4, 6, 9]

# forward

fw = [0, 1, 2, 3]
fw[0] = [1, 0, 0, 0]
fw[1] = [0, 1, 0, 0]
fw[2] = [0, 0, 1, 0]
fw[3] = [0, 0, 0, 1]

# reverse
```

```
rv = [0, 1, 2, 3]
rv[0] = [0, 0, 0, 1]
rv[1] = [0, 0, 1, 0]
rv[2] = [0, 1, 0, 0]
rv[3] = [1, 0, 0, 0]

def move(seq):
    for i in range(0, 100):                  # 100 is 1/3 rotation to
open/close lock
        for step in range(0, 4):
            for pin in range(0, 4):
                xpin = step_pins[pin]       # get GPIO number
                if seq[step][pin] != 0:     # check if pin in sequence is 1
or 0
                    wp.digitalWrite(xpin, True)  # set pin high
                else:
                    wp.digitalWrite(xpin, False)
            # Wait before moving on
            time.sleep(wait_time)

def lock():
    print("locking ")
    move(fw)
    return True

def unlock():
    print("unlocking ")
    move(rv)
    return False
```

Wanneer deze functies uitgevoerd zijn geven ze als return een Boolean mee, zodat we weten of de val 'locked' (true) is of 'unlocked' (false). Bv. in het main script sluiten we de val alsvolgt:

```
            locked = lock()
```

In een verkorte versie van een python if-statement bepalen we ook dat de val eerst moet unlocken als de val wil opengaan

```
        elif must_open and not open: # release
            locked = unlock() if locked else locked
```

1.3 Ultrasonic measurement (2)

Doorheen het gekraste glas zien we bovenaan de camera en daaronder de acoustische sensor. De sensor werkte eerst niet heel goed in het kastje. Dit had vermoedelijk te maken met de houten poort 'steunen' (op de foto niet te zien) die steeds een meting van 20-21 opleverden, als de poort dicht was maar ook als de poort open was :

```
Status : Observing, Distance:  21.149158477783203
No mouse in cage. Door must be opened
```

Wanneer een zacht object dicht genoeg bij de sensor komt schiet de waarde omhoog ipv omlaag. De code klopt volgens het boekje, en na tests bleek dat enkel de afstand tot harde objecten (houten blokje bijvoorbeeld) goed worden geschat.

De functie is overgenomen uit het labo:

```
import time
import wiringpi as wp

echo_pin = 0
```

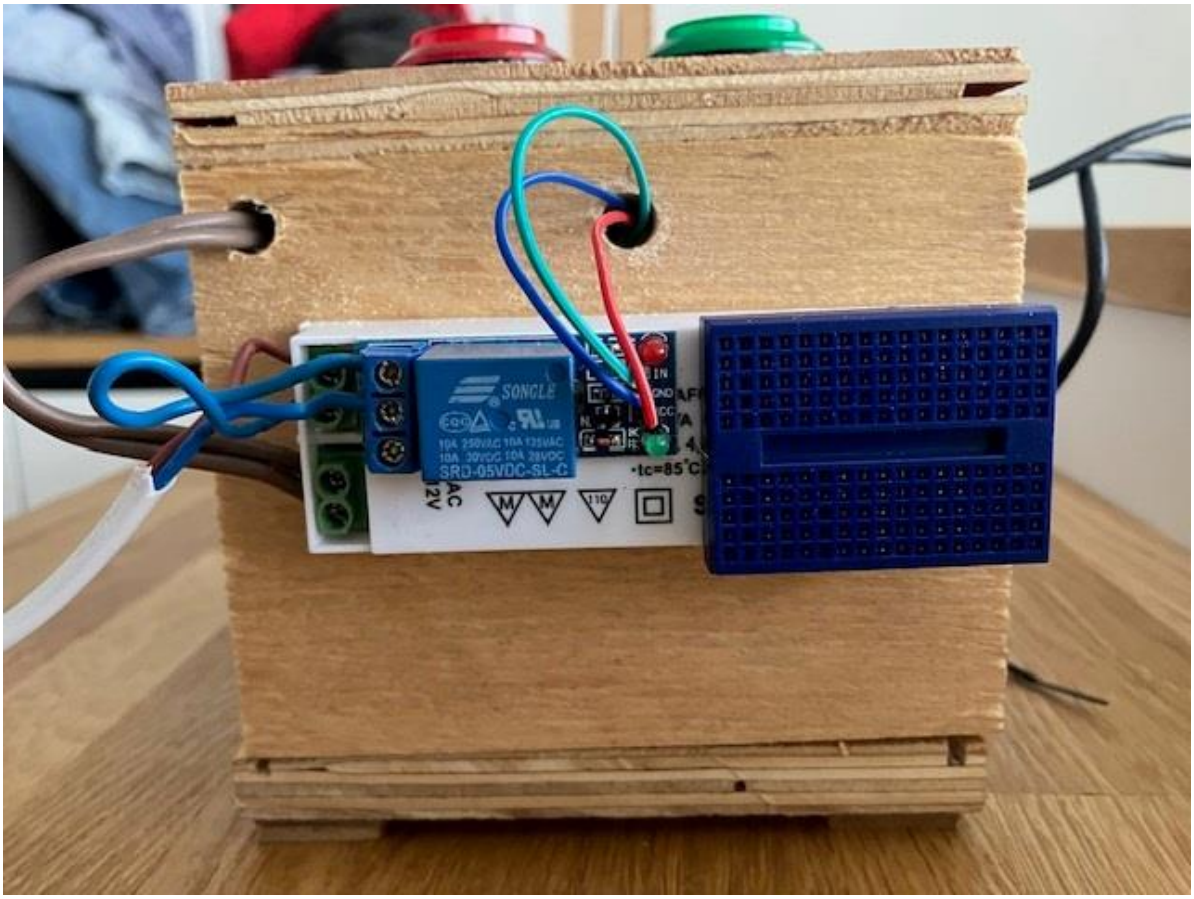
```
trig_pin = 1

def measure():
    wp.digitalWrite(trig_pin, 1)
    time.sleep(0.00001)
    wp.digitalWrite(trig_pin, 0)

    while wp.digitalRead(echo_pin) == 0:
        pass
    signal_start = time.time()
    while wp.digitalRead(echo_pin) == 1:
        pass
    signal_end = time.time()

    signal_time = signal_end - signal_start
    distance = signal_time * 17000
    time.sleep(0.5)
    return distance
```

De trigger pin wordt op output gezet gedurende een micro seconde. Vervolgens wordt gemeten hoe lang de echo pin als input signaal ontvangt. D.m.v. een eenvoudige formule kan dan de afstand afgeleid worden.

In de main code is de logica alsvolgt:

```
        measurement = measure()

        if measurement <= 15 :
            print("Mouse in cage. Door must be closed")
            must_open = False
```

1.4 Light turn on/off (2)



Om het voor mezelf wat moeilijker te maken, koos ik voor een Chinese relay die ik nog had liggen omdat die kleiner was en makkelijker te bevestigen op de 12v transfo (voeding van het licht). Echter deze moét gestuurd worden door 5v terwijl de OPI op haar outputs slechts 3,3v levert. Dit heb ik opgevangen met een extra transistor die op de base 3,3v van de OPI output krijgt, maar ook 5v van de OPI op collector om dan te schakelen naar het relay.

De code is eenvoudig: op pin 15 stuurt een output de 12v lamp.

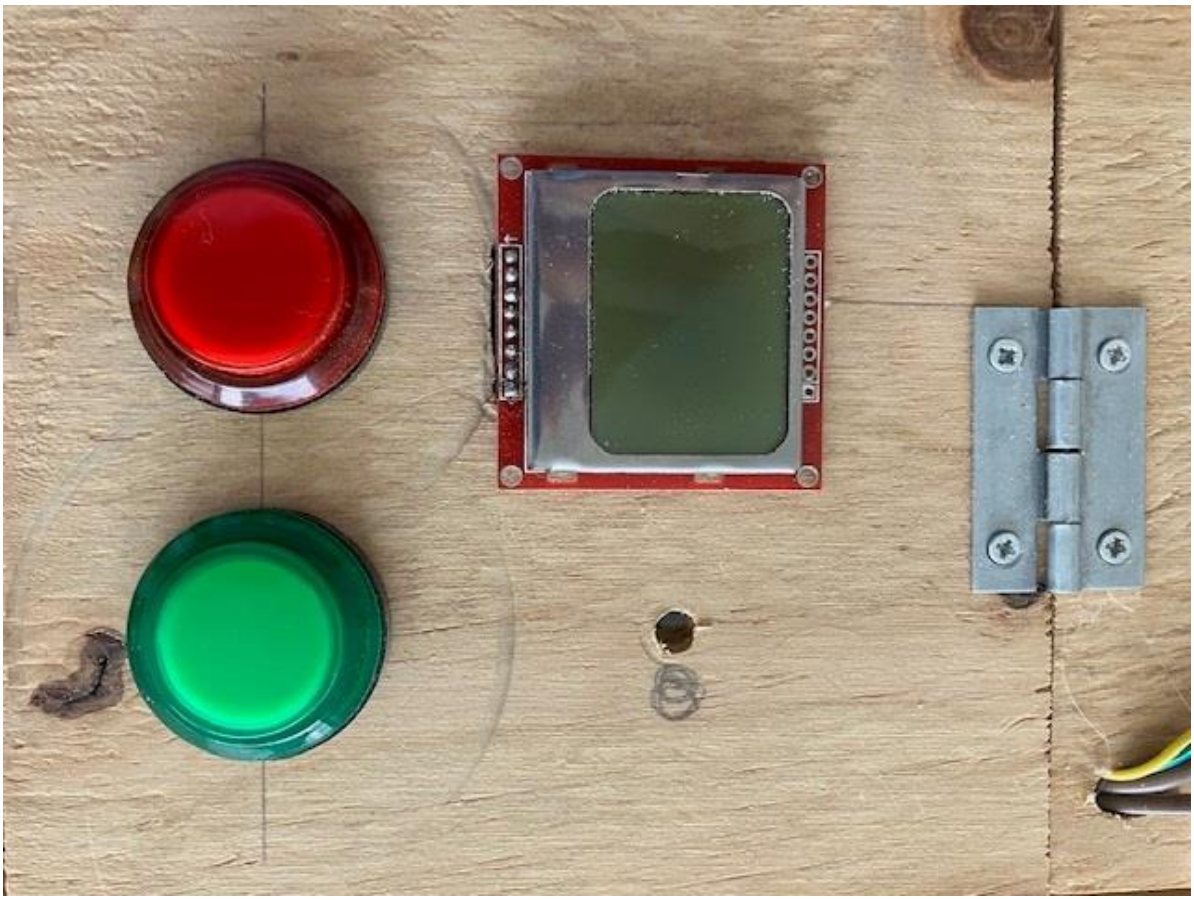
```
light_pin = 15

def light_on():
    print("Light on")
    wp.digitalWrite(light_pin, True)
    return True

def light_off():
    print("Light off")
```

```
    wp.digitalWrite(light_pin, False)
    return False
```

Wanneer de deur gesloten is, gaat het licht aan, wanneer de deur open is, gaat het licht uit.

```
        if not must_open and open:    # catch
            (…)
            light = light_on()
            (…)
        elif must_open and not open: # release
            (…)
            light = light_off()
```

1.5 Buttons (2)



Twee input buttons zijn voorzien om te openen (groen) en te sluiten (rood).

```
# Set button pins as input
wp.pinMode(green_pin, 0)         # Set pin to mode 0 ( INPUT )
wp.pinMode(red_pin, 0)
```

Het programma is zo opgevat dat de OPI in een aparte thread luistert naar button-events.

```
t1 = threading.Thread(target=buttons, args=(green_pressed, red_pressed,
exit_event))
```

```
green_pressed = threading.Event()
red_pressed = threading.Event()
```

De buttons kunnen de automatische werking onderbreken, maar MouseTrap 5.0 (versie bij redactie van deze paper) zal steeds verder evalueren of er geen object / muis in de val zit en na openen de val toch sluiten als er effectief een muis in zit.

```
def buttons(green_pressed, red_pressed, exit_event):
    while True:
        if(wp.digitalRead(green_pin) == 1): #input is active up (pull down)
            green_pressed.set()
            time.sleep(0.3) #anti bouncing
        if(wp.digitalRead(red_pin) == 1): #input is active up (pull down)
            red_pressed.set()
            time.sleep(0.3) #anti bouncing
        if exit_event.is_set():
            break
```

In de hoofdthread wordt het event opgevangen en meteen bewaard in een state (‘green’ of ‘red’).

```
        if green_pressed.is_set():
            print("Green button pressed")
            green = True
            green_pressed.clear()
        if red_pressed.is_set():
            print("Red button pressed")
            red = True
            red_pressed.clear()
```

Vervolgens wordt afhankelijk van de status van de val en de ingedrukte knop de opdracht state (must_open) gezet:

```
            print("No mouse in cage. Door must be opened")
            must_open = True
            if red:
```

```
                print("However, red button was pressed. Door must be closed")
                must_open = False
                red = False
```

1.6 LCD (2)

De LCD die in het aangekochte pakket zat, werkte niet. Het was de bedoeling de nieuw bestelde (rode) LCD aan te sturen met een aparte thread (zodat seconden correct kunnen worden weergegeven), maar om één of andere reden bleef de main thread vastlopen telkens na 5 tal seconden wanneer ik dit probeerde uit te voeren. Vandaar dat nu de LCD code is opgenomen in het main programma. Soms skipt het een seconde, maar dat is geen drama. De LCD wordt geactiveerd door de chip select pin (13) op laag te zetten.

```
def ActivateLCD():
    wp.digitalWrite(pin_CS_lcd, 0)       # Actived LCD using CS
    time.sleep(0.000005)
def DeactivateLCD():
    wp.digitalWrite(pin_CS_lcd, 1)       # Deactived LCD using CS
    time.sleep(0.000005)
```

De huidige tijd wordt geformat met ‘strftime’ (gesuggereerd door GitHub CoPilot)

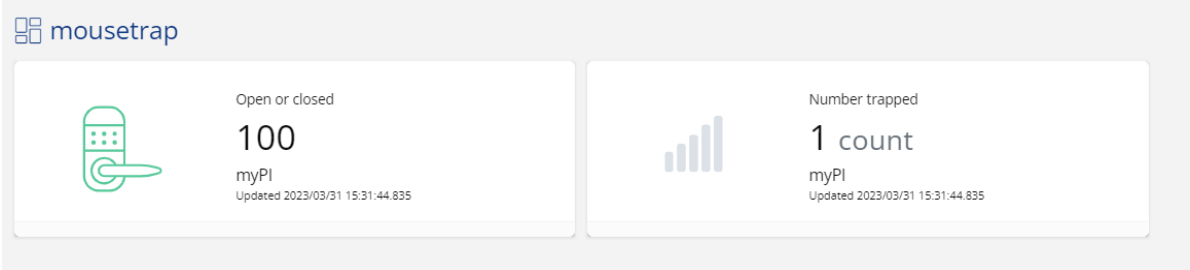
```
        current_time = time.time()
        date_string = time.strftime("%d/%m/%Y", time.localtime(current_time))
        time_string = time.strftime("%H:%M:%S", time.localtime(current_time))
        trapped_string = str(number_trapped)

        status_string = 'Armed' if open else 'Trapped'

        ActivateLCD()
        lcd.clear()
        lcd.go_to_xy(0, 0)
        lcd.put_string(date_string + '\n' + time_string + '\n' + 'Stat: ' +
status_string + '\n' + 'Trapped: ' + trapped_string + '\n')
        lcd.refresh()
        DeactivateLCD()
```

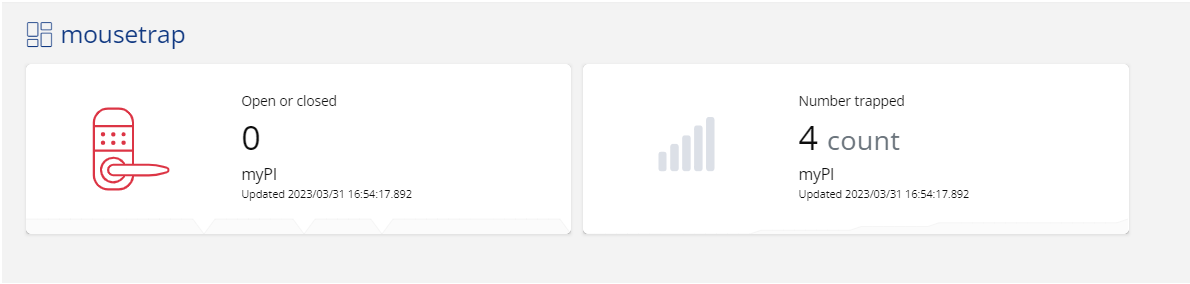
1.7 uBeac upload (2)

Configuratie volgens les:

```
url = "http://itfactory012345678.hub.ubeac.io/iotjorisvp"
uid = "iotjorisvp"
```

```
        status = 100 if open else 0
        data= {"id": uid,"sensors":[{'id': 'status','data': status},{'id':
               'number trapped','data': number_trapped}]}
        r = requests.post(url, verify=False, json=data)
```





Live Data

Type	Sensor	Gateway	Date	Value
Counter (Count)	number trapped	My gateway	2023/03/31 16:42:49.306	0
N/A (N/A)	status	My gateway	2023/03/31 16:42:51.691	100
Counter (Count)	number trapped	My gateway	2023/03/31 16:42:51.691	0
N/A (N/A)	status	My gateway	2023/03/31 16:42:54.073	100

2 OPTIONAL FEATURES

2.1 Live stream (2)

Ik koos voor Motion-software (https://motion-project.github.io/index.html). Ik heb het op de Orange Pi uit de source moeten compilen. Na make en make install kan je de service gewoon starten met 'systemctl start motion'.

```
def open_stream():
    os.system('sudo systemctl start motion')
    return True

def close_stream():
    os.system('sudo systemctl stop motion')
    return False
```

De OPI krijgt via Wifi ook een IP address op mijn thuisnetwerk 192.168.0.0/24 . Wanneer ik naar dat adres surf via de browser (poort 8081):

2.2 DC-motor (2)

Via twee software matige PWM pins kunnen twee richtingen voor de motor gezet worden:

```
def pullUp(speed):
    wp.softPwmWrite(dcmotor_pin1, 0)
    wp.softPwmWrite(dcmotor_pin2, speed)

def pullDown(speed):
    wp.softPwmWrite(dcmotor_pin1, speed)
    wp.softPwmWrite(dcmotor_pin2, 0)

def fullStop():
    wp.softPwmWrite(dcmotor_pin1, 0)
    wp.softPwmWrite(dcmotor_pin2, 0
```

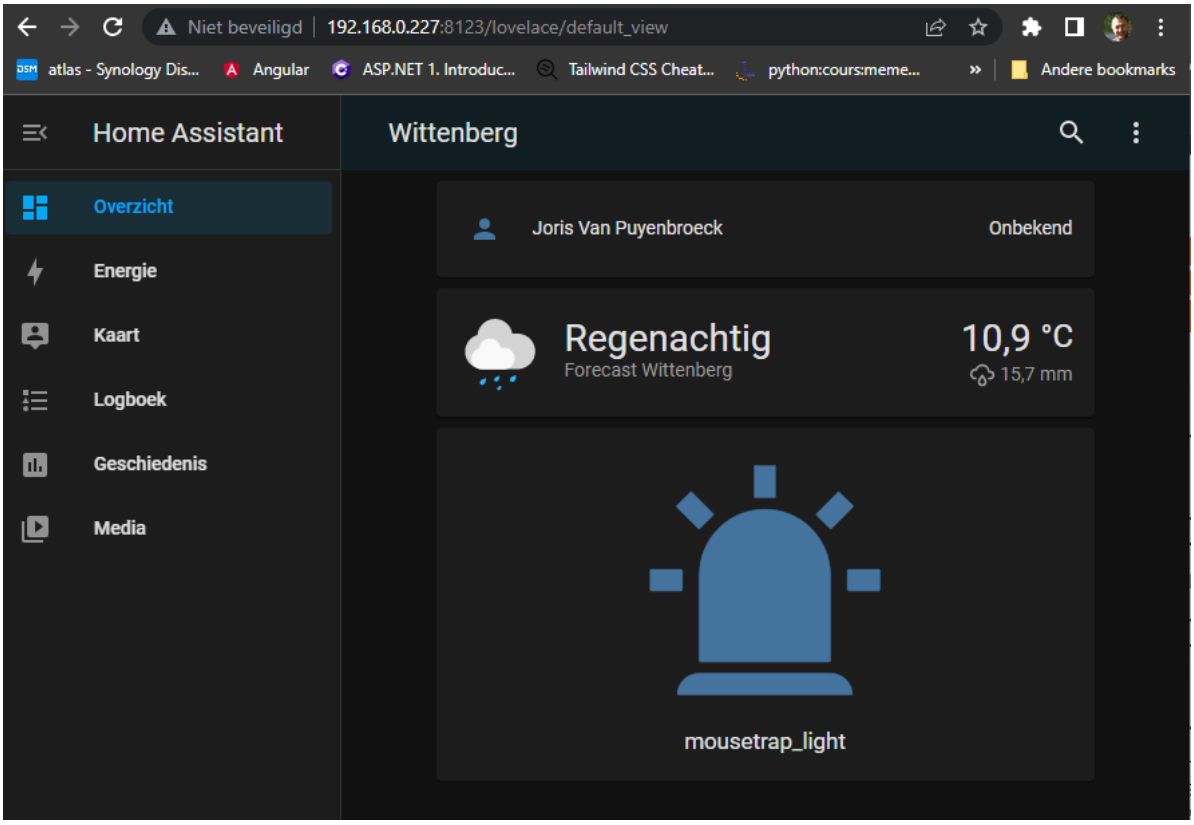
Wanneer de muizeval poort sluit, moet de H-brug op de L293D eerst de motor in full stop zetten, en daarna de poort neerlaten (pulldown) met een ingestelde snelheid ‘speed’. Het sleep interval is gekozen ifv de lengte van de draad.

```
        if not must_open and open:   # catch
            locked = unlock() if locked else locked
            fullStop()
            pullDown(speed)
            time.sleep(0.5)
            fullStop()
```

2.3 Home-assistant (2)

Op mijn thuis server (Synology) http://192.168.0.227:8123/lovelace/default_view heb ik een Home Assistant service draaien en heb daar een dashboard ingesteld.

Het is een 'proof of concept' : een schakelaar die het licht aan- en uitzet. Ik kan en zal uiteraard nog uitbreiden.



De configuratie hierachter :

```
title: Wittenberg
views:
  - path: default_view
    title: Home
    cards:
      - type: entities
        entities:
          - person.joris_van_puyenbroeck
      - type: weather-forecast
        entity: weather.wittenberg
        show_forecast: false
      - show_name: true
        show_icon: true
        type: button
```

```
        entity: input_boolean.mousetrap_light
        name: mousetrap_light
        hold_action:
          action: url
          url_path: http://192.168.0.222/light_turnoff.php
        tap_action:
          action: url
          url_path: http://192.168.0.222/light_turnon.php
```

Wat HA hier doet is de virtuele button een boolean 'entiteit' laten zetten 'mousetrap_light' die kan geswitched worden. Om aan te zetten voert HA dan een url uit. Heel ingewikkeld allemaal, maar het voordeel hier is dat het ook remote kan geactiveerd worden. Indien ik een console switch bv. 'gpio write 15 1' had gebruikt, had dit niet kunnen gebruikt worden op mijn thuisserver die HA runt, omdat die natuurlijk geen lokale GPIO heeft. Nu runt de OPI een webserver met daarop enkele php scripts die op afstand uitgevoerd kunnen worden. Dit is verre van ideaal, maar ik heb nog een te beperkte kennis van Home Assistant om het beter te laten werken.

2.4 Multi- Threading (2)

Zie onder 1.5 Buttons. Tussen de twee threads (main and button) worden events gezet en ge-cleared. De threads worden bij afsluiten allemaal netjes gestopt met een exit event. Een derde thread voor LCD is niet gelukt. Mogelijks overbelasting van de CPU ?

```
except KeyboardInterrupt:
    print("KeyboardInterrupt")
    exit_event.set()
```

```
        if exit_event.is_set():
            break
```

2.5 Artificial intelligence (2)

We nemen een foto van wat in de val zit, maar we sluiten eerst de live stream, want die gebruikt hetzelfde /dev/ . We schrijven de foto naar mousetrap.bmp.

```
import cv2
from live.stream import open_stream, close_stream

def take_picture():
  # close stream, otherwise it will interfere with the picture
  stream = close_stream()
  # take picture
  webcam = cv2.VideoCapture(1)
  ret, frame = webcam.read()
  rotate = cv2.rotate(frame, cv2.ROTATE_180)
  cv2.imwrite('/home/orangepi/pi_scripts/mouse_trap/ai/mousetrap.bmp', frame)
  del (webcam)
  # open stream
  stream = open_stream()
```

Daarna gebruiken we een aangepaste versie van het label_image.py script. Aangezien we het script niet meer zoals in les aanroepen vanop de command line, geven we de argumenten lokaal mee.

```
import time
import numpy as np
from PIL import Image
import tflite_runtime.interpreter as tflite


def is_mouse():
  image = './ai/mousetrap.bmp'
  model_file = './ai/model_unquant.tflite'
  input_mean = 127.5
  input_std = 127.5
  num_threads = 3
  labels = ['0 cat', '1 mouse', '2 background']

  interpreter = tflite.Interpreter(
      model_path= model_file,
      num_threads= num_threads)
  interpreter.allocate_tensors()

  input_details = interpreter.get_input_details()
  output_details = interpreter.get_output_details()
```

```
# check the type of the input tensor
floating_model = input_details[0]['dtype'] == np.float32

# NxHxWxC, H:1, W:2
height = input_details[0]['shape'][1]
width = input_details[0]['shape'][2]
img = Image.open(image).resize((width, height))

# add N dim
input_data = np.expand_dims(img, axis=0)

if floating_model:
  input_data = (np.float32(input_data) - input_mean) / input_std

interpreter.set_tensor(input_details[0]['index'], input_data)

start_time = time.time()
interpreter.invoke()
stop_time = time.time()

output_data = interpreter.get_tensor(output_details[0]['index'])
results = np.squeeze(output_data)

top_k = results.argsort()[-5:][::-1]

for i in top_k:
  if floating_model:
    print('{:08.6f}: {}'.format(float(results[i]), labels[i]))
  else:
    print('{:08.6f}: {}'.format(float(results[i] / 255.0), labels[i]))

print('time: {:.3f}ms'.format((stop_time - start_time) * 1000))

if results[1] > results[0]:
    return True
else:
    return False
```

De functie is_mouse() berekent op basis van het model of deze afbeelding een muis bevat (True) of niet (False). Daartoe vergelijkt het de score van de muise klasse (die zich in de labels array op de tweede positie met index 1 bevindt, met de score van de sock klasse met index 0.

Update 5/4/’23 . De switch van zachte naar harde objecten om te scannen betekende ook het hertrainen van het AI model.

3 SELF-GRADING

Ik geef mezelf 20/20 . Als ik alles optel kom ik aan 24/20.

4 LINK NAAR YOUTUBE

https://www.youtube.com/watch?v=qPPL8Nrp0o0

5 BIJLAGE : CODE

```
import time
import wiringpi as wp
import threading
import requests
from door.dcmotor import pullDown, pullUp, fullStop
from lock.stepper import lock, unlock
from distance.distance_sensor import measure
from button.button import buttons
from light.relay import light_on, light_off
from screen.lcd import ActivateLCD, DeactivateLCD
from screen.ClassLCD import LCD
from live.stream import open_stream, close_stream
from ai.camera import take_picture
from ai.label_image import is_mouse

# SETUP
print("Start")
echo_pin = 0
trig_pin = 1
green_pin = 8
red_pin = 7
dcmotor_pin1 = 2
dcmotor_pin2 = 5
light_pin = 15
pin_CS_lcd = 13
speed = 100
locked = False
open = True
must_open = True
stay_closed = False
green = False
red = False
light = False
stream = False
mouse = False
scanned = False
number_trapped = 0
step_pins = [3, 4, 6, 9]
url = "http://itfactory012345678.hub.ubeac.io/iotjorisvp"
uid = "iotjorisvp"
status = 0
exit_event = threading.Event()
green_pressed = threading.Event()
red_pressed = threading.Event()
```

```
PIN_OUT     =   {
                'SCLK'  :   14,
                'DIN'   :   11,
                'DC'    :   16,
                'CS'    :   13,
                'RST'   :   10,
                'LED'   :   12,
}

wp.wiringPiSetup()

# Set dist_sensors pins as output and input
wp.pinMode(trig_pin, 1)     # output
wp.pinMode(echo_pin, 0)     # input

# Set all stepper pins as output
for pin in step_pins:
    wp.pinMode(pin, 1) # 1 is output
    wp.digitalWrite(pin, False)

# Set pins as a softPWM output
wp.softPwmCreate(dcmotor_pin1, 0, 100)
wp.softPwmCreate(dcmotor_pin2, 0, 100)

# Set light pin as output
wp.pinMode(light_pin, 1)

# Set button pins as input
wp.pinMode(green_pin, 0)         # Set pin to mode 0 ( INPUT )
wp.pinMode(red_pin, 0)

# Set SPI pins as output and activate lcd
wp.wiringPiSPISetupMode(1, 0, 400000, 0)  #(channel, port, speed, mode)
wp.pinMode(pin_CS_lcd , 1)            # Set pin to mode 1 ( OUTPUT )
ActivateLCD()
lcd = LCD(PIN_OUT)

#create two new threads
t1 = threading.Thread(target=buttons, args=(green_pressed, red_pressed,
exit_event))
# t2 = threading.Thread(target=beac)

#start the threads
t1.start()
# t2.start()

# main loop

# start streaming
```

```
stream = open_stream()

try:

    lcd.clear()
    lcd.set_backlight(0)

    while True:

        # lcd output
        current_time = time.time()
        date_string = time.strftime("%d/%m/%Y", time.localtime(current_time))
        time_string = time.strftime("%H:%M:%S", time.localtime(current_time))
        trapped_string = str(number_trapped)

        status_string = 'Armed' if open else 'Trapped'

        ActivateLCD()
        lcd.clear()
        lcd.go_to_xy(0, 0)
        lcd.put_string(date_string + '\n' + time_string + '\n' + 'Stat: ' +
status_string + '\n' + 'Trapped: ' + trapped_string + '\n')
        lcd.refresh()
        DeactivateLCD()

        # ubeac output

        status = 100 if open else 0
        data= {"id": uid,"sensors":[{'id': 'status','data': status},{'id':
'number trapped','data': number_trapped}]}
        r = requests.post(url, verify=False, json=data)

        if green_pressed.is_set():
            print("Green button pressed")
            green = True
            green_pressed.clear()
        if red_pressed.is_set():
            print("Red button pressed")
            red = True
            red_pressed.clear()

        measurement = measure()

        if measurement <= 15 :
            print("Object in cage. Door must be closed")
            must_open = False

            if mouse and scanned:
```

```
                print("Object was recognised as mouse. Must trap.")
                must_open = False
                stay_closed = True
                number_trapped += 1

            elif not mouse and not scanned:
                print("Object was not yet recognised as mouse. Must close and
scan first.")
                must_open = False
                stay_closed = False
                scanned = False

            else:
                print("Object was not recognised as mouse. Must release")
                must_open = True
                stay_closed = False

            if green:
                print("Green button was pressed. Door must be opened.")
                must_open = True
                scanned = False
                mouse = False
                stay_closed = False
            if red:
                print("Red button was pressed. Door must be closed.")
                must_open = False
                red = False

        else :
            print("No object in cage. Door must be opened")
            must_open = True
            scanned = False
            stay_closed = False
            if green:
                print("Green button was pressed. Door must be opened")
                must_open = True
                green = False
            if red:
                print("Red button was pressed. Door must be closed")
                must_open = False
                red = False


        if not must_open and open:   # catch
            locked = unlock() if locked else locked
            fullStop()
            pullDown(speed)
            time.sleep(0.5)
            fullStop()
```

```
            stay_closed = True
            locked = lock()
            measurement = measure()
            open = False
            light = light_on()
            if not scanned:
                take_picture()
                mouse = is_mouse()
                scanned = True
            print("Status : Closed, Distance: ", measurement)

        elif not must_open and not open: # stay closed
            measurement = measure()
            open = False
            print("Status : Closed, Distance: ", measurement)
            time.sleep(1)

        elif must_open and not open: # release
            locked = unlock() if locked else locked
            if not stay_closed:
                pullUp(speed)
            open = True
            if green:
                print("Granting 10 secs to remove object")
                time.sleep(10)
                green = False
            measurement = measure()
            light = light_off()
            print("Status : Open, Distance: ", measurement)
            time.sleep(1)

        else:
            measurement = measure()
            open = True
            print("Status : Observing, Distance: ", measurement)
            time.sleep(1)

except KeyboardInterrupt:
    print("KeyboardInterrupt")
    exit_event.set()
    lcd.clear()
    lcd.refresh()
    lcd.set_backlight(1)
    DeactivateLCD()
    locked = unlock() if locked else locked
    light = light_off() if light else light
    pullUp(speed)
    stream = close_stream()
    print("\nDone")
```