Namen: Stijn Boutsen & Leendert Wilms
Richting: MA – EA-ICT

Schooljaar: 2014 – 2015
Datum: 1 juni 2015

"Niveaudetectie van gevulde wijnflessen"

1 Algemene inleiding

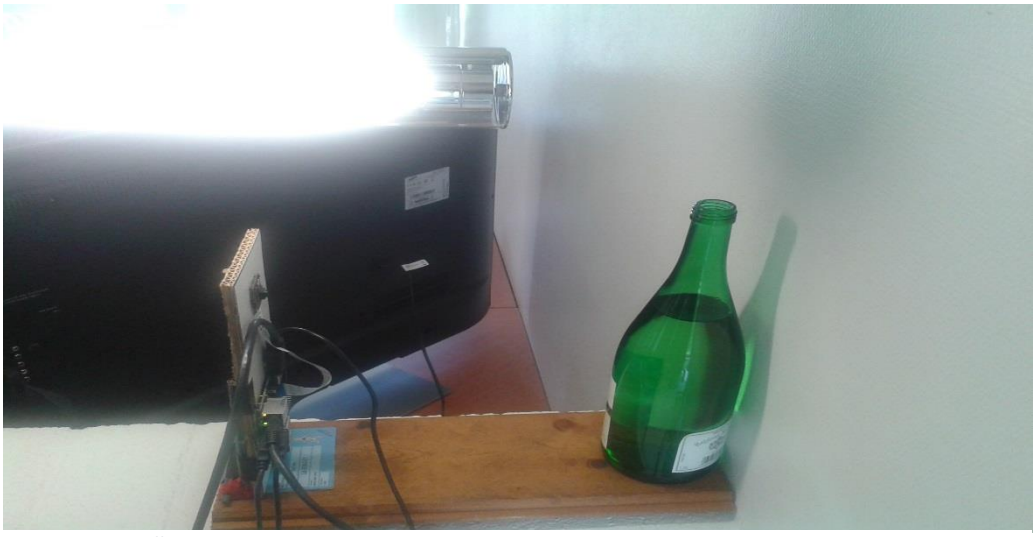
Voor het vak innovatie in de sensortechnologie werd een project gebouwd omtrent het gebruik van een camera. Er werd gekozen om gebruik te maken van een Raspberry Pi en de Pi NOIR camera in combinatie met het Raspbian OS.

Het project bepaalt aan de hand van de camera of een wijnfles, die voorbij een sensor komt, al dan niet genoeg gevuld is. Indien de fles genoeg gevuld is mag deze verder gaan en is deze in principe klaar voor verkoop. Is dit niet het geval, dan is oftewel de fles niet genoeg gevuld door het bottlingproces, of is er een lek in de wijnfles en is ze dus kapot. In beide gevallen moet er sowieso worden ingegrepen.

De benodigde materialen zijn dus als volgt:

- De Raspberry Pi
- Pi NOIR camera
- Externe verlichting
- LDR
- Weerstand
- Condensator

Het programma wordt gestart met het commando "./runDetectie".

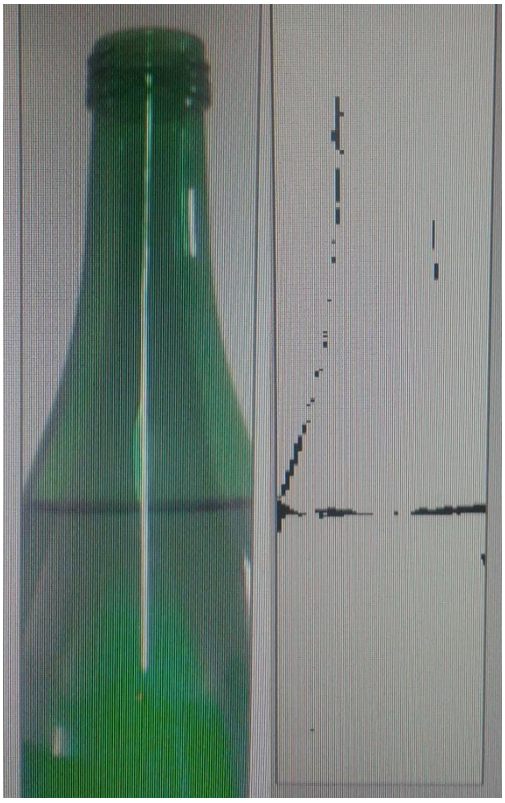


Figuur 1: Opstelling

2 Software – camera module

In de bijlage van dit verslag vind u de software geschreven voor de camera module (Bottlemachine.py). Deze software gaat het vloeistofniveau van de fles bepalen en kijken of deze overeenstemt met een vooropgestelde hoogte. Onderstaande figuur geeft de opbouw in testfase van de module.

Er wordt een foto van 100x400px getrokken door de Pi NOIR camera. Door een lage resolutie te nemen, zal de processor veel sneller zijn in de benodigde berekeningen. Omdat flessen wijn voornamelijk groen zijn, gaan we ervoor zorgen dat de groene pixels boven een bepaalde treshold allemaal wit worden. Dit geldt ook voor de witte achtergrond. Uit figuur 2a valt op te merken dat de vloeistoflijn op de foto een donkerdere lijn veroorzaakt. Met andere woorden, alle donkere pixels worden zwart gemaakt en de lichtere pixels worden wit gemaakt. Dit geeft dat de meeste zwarte pixels zich gaan bevinden rond de vloeistoflijn. Op figuur 2b kan je zien wat dit geeft in een zwart-wit foto.

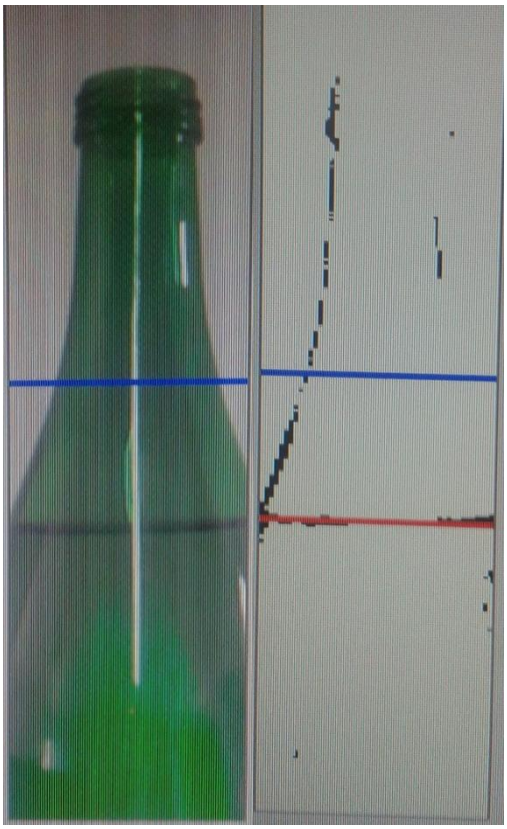


Figuur 2: a) Voor processing. b) Na processing

Merk op dat de flessen zonder dop zijn getrokken. Dit zorgde voor een extra schaduw aka zwarte pixels. Omdat dit normaal gezien op het einde van een proces zou staan, als de doppen nog niet op de fles zitten, hebben wij verkozen ook deze weg te laten. De lange witte streep in het midden op de fles komt door een TL-lamp die voor extra belichting zorgde.

Omdat in een echt proces de flessen noch de camera ooit van hoogte veranderen, is er besloten om een vaste referentielijn te nemen voor elke fles. Deze referentielijn is te zien op figuur 3 in het blauw. Indien de vloeistoflijn boven deze waarde komt te liggen, zal er een groene lijn door het vloeistofoppervlak worden getrokken. Indien dit niet zo is, een rode lijn.

Omdat het vloeistofoppervlak meer donkere pixels heeft, kan men bepalen op welke hoogte in de foto de meeste zwarte pixels horizontaal voorkomen oftewel waar de vloeistoflijn zich bevind. Eenmaal we dit hebben, is het nog een simpele vergelijking om vervolgens tot een conclusie te komen of een fles wijn al dat niet genoeg gevuld is. Figuur 3 geeft een voorbeeld van een niet goed gevulde fles met bepaling van de minimum hoogte van het vloeistofoppervlak.



Figuur 3: a) Voor processing. b) Na processing

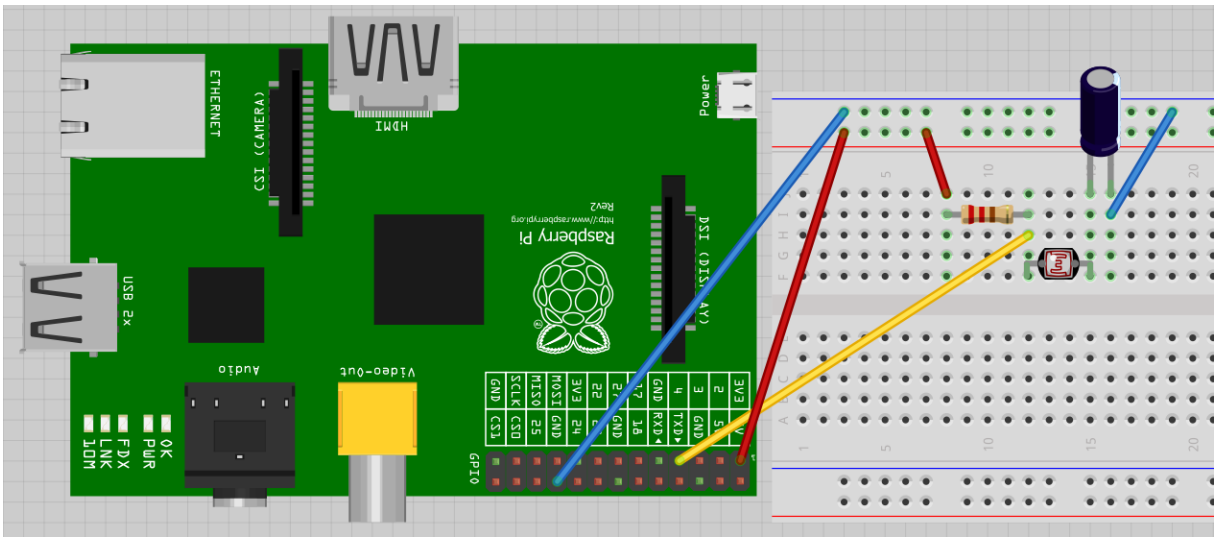
3 Hardware – detectie module

Met behulp van een LDR is het mogelijk om de lichtintensiteit te detecteren. Zo kan er bepaald worden of er een fles aanwezig is. Wanneer er een fles gedetecteerd wordt er een foto genomen waarna de beeldanalyse gestart wordt.

Om de weerstand van de LDR uit te lezen werd er gebruik gemaakt van een charge/discharge circuit. Door het plaatsen van een condensator ontstaat er een RC kring met een tijdsconstante $\tau = R*C$. Deze tijdsconstante geeft de tijd waarna de condensator 66.6% opgeladen is. De volledige capaciteit wordt bereikt na ongeveer 5τ. Hiermee hoeft er geen ADC te worden gebruikt.

Wanneer de GPIO pin van de Raspberry Pi hoog is wordt er een lading op de condensator geplaatst, afhankelijk van de tijdsconstante. Daarna wordt de GPIO pin laag getrokken waarna de condensator ontlaadt. Hierbij loopt er een counter die bijhoudt hoe lang het duurt voor de condensator leeg is.

Indien de lichtintensiteit die op de LDR valt verandert, verandert de weerstand van de LDR zelf. Hierdoor wordt er een andere tijdsconstante τ bekomen waardoor er meer of minder lading op de condensator geplaatst wordt.



Figuur 4- Charge/discharge circuit

Bijlage

(De laatste versies van de bestanden zijn terug te vinden op https://github.com/Boutsman/NiveauDetectie)

Python-script voor image processing

```
import imgproc
from imgproc import *
# Import komt van
https://www.cl.cam.ac.uk/projects/raspberrypi/tutorials/image-processing/intro.html

# ALTIJD EERST VOLGENDE SCRIPT TYPEN
# sudo modprobe bcm2835-v4l2


# 200x800px waren de begininstellingen voor te testen, nadien aangepast voor sneller systeem
width =  100
height = 400

# Instelbare hoogte, wanneer is de fles wijn genoeg gevuld
height_wine =  185


# Camera en viewer openen
camera = Camera(width, height)
viewer = Viewer(width, height, "Bottlemachine")

# Foto nemen en tonen via viewer
img = camera.grabImage()
preview = camera.grabImage()

# Een blauwe lijn trekken waar het vloeistofoppervlak boven moet liggen
for x in range (0, preview.width):
        preview[x, height_wine -1] = 0, 0, 255
        preview[x, height_wine] = 0, 0, 255
        preview[x, height_wine +1] = 0, 0, 255

viewer.displayImage(preview)


# Deze functie kan de vorm bepalen van de fles
# If green > 150 and red > 150 and blue > 150


# iterate over each pixel in the image
for x in range (0, img.width):
        for y in range (0, img.height):
                red, green, blue = img[x, y]

                if blue > green and blue > red and blue < 190:
                        # Change colour to black
                        img[x, y] = 0, 0, 0
                else:
                        # Change colour to white
                        img[x, y] = 255, 255, 255


# Instellingen voor de lijn met meeste zwarte pixels te bepalen
countblack = 0
maxIndex = 0
max = 0

# Deze functie bepaalt welke lijn de meeste zwarte pixels heeft en hoeveel deze zijn
for y in range (0, img.height):
        for x in range (0, img.width):
                # Aangezien we zwart wit gebruiken, moeten we maar naar 1 ding kijken vb of groen 255 of 0 is
                red, green, blue = img[x, y]

                # Tel het aantal zwarte pixels per rij
                if red < 1 and green < 1 and blue < 1:
                        countblack += 1
                        if countblack > max:
                                max = countblack
                                maxIndex = y
                        else:
                                pass
                else:
                        pass
        # Lijst terug 0 maken, anders wordt alles opgeteld
        countblack = 0

print "Op lijn", maxIndex, "zijn de meeste zwarte pixels aanwezig."
print "Op deze lijn bevinden zich", max, "zwarte pixels."

if maxIndex < height_wine:
        print "De fles is VOLDOENDE gevuld."

        # Een groene lijn trekken van 3px dik, op het vloeistofoppverlak
        for x in range (0, img.width):
                img[x, maxIndex -1] = 0, 255, 0
                img[x, maxIndex] = 0, 255, 0
                img[x, maxIndex +1] = 0, 255, 0
else:
        print "De fles is ONVOLDOENDE gevuld."

        # Een rode lijn trekken van 3px dik, op het vloeistofoppverlak
        for x in range (0, img.width):
                img[x, maxIndex -1] = 255, 0, 0
                img[x, maxIndex] = 255, 0, 0
                img[x, maxIndex +1] = 255, 0, 0

# Een blauwe lijn trekken waar het vloeistofoppervlak boven moet liggen
for x in range (0, img.width):
        img[x, height_wine -1] = 0, 0, 255
        img[x, height_wine] = 0, 0, 255
        img[x, height_wine +1] = 0, 0, 255

viewer.displayImage(img)

# Lang genoeg om te zien wat er is gebeurd (10sec.)
waitTime(10000)
```

Python-script voor meting van de tijdsconstante.

```
import RPi.GPIO as GPIO
import time

# setup function
def setup():
  global oldVal
  global pinNr
  oldVal = 0
  pinNr = 4
  GPIO.setmode(GPIO.BCM)


def RCtime():
  global pinNr
  global oldVal

  measurement = 0

  GPIO.setup(pinNr, GPIO.OUT)
  GPIO.output(pinNr, GPIO.LOW)
  time.sleep(0.1)

  GPIO.setup(pinNr, GPIO.IN)

  while(GPIO.input(pinNr)==GPIO.LOW):
    measurement += 1

  #return measurement
  if measurement>5000:
    if oldVal==0:
              print '1'
      execfile("Bottlemachine.py")
    oldVal = 1
  else:
    if oldVal==1:
      print '0'
    oldVal = 0

#main function
setup()
while True:
  RCtime()
```