Software-ontwerp
Van
Babyscanner

Opgesteld door	:	Groep Babyscanner
Projectleider	:	Timothy Singowikromo
Projectleden	:	Kai Harten
		Jure Vidmar
		Talha Kocak
		Timothy Singowikromo
Begeleider	:	J.Z.M Broeders
Datum van uitgifte	:	24 – 06 -2018

Inhoudsopgave

Versiehistorie

Versie	Datum	Wijzingen	Auteur
1.0	24-06-2018	Opzet software design	Kai Harten

1 Inleiding

In dit document wordt het ontwerp beschreven van het systeem babyscanner. Dit wordt gedaan met UML-diagrammen waar het systeem op architectuurniveau wordt laten zien. Vervolgens wordt het systeem uitgebreider beschreven in het gedetailleerde ontwerp.

1.1 Doel

Een systeem ontwikkelen wat bij/rondom een couveuse geplaatst kan worden en een hoogte en lengtemeting kan uitvoeren. Een 3D-model kan vervolgens gegenereerd worden.

1.2 Documentconventies

Dit document zal worden uitgewerkt in Word met lettertype Calibri en lettergrootte 11.

1.3 Doelgroepen en suggesties voor het lezen

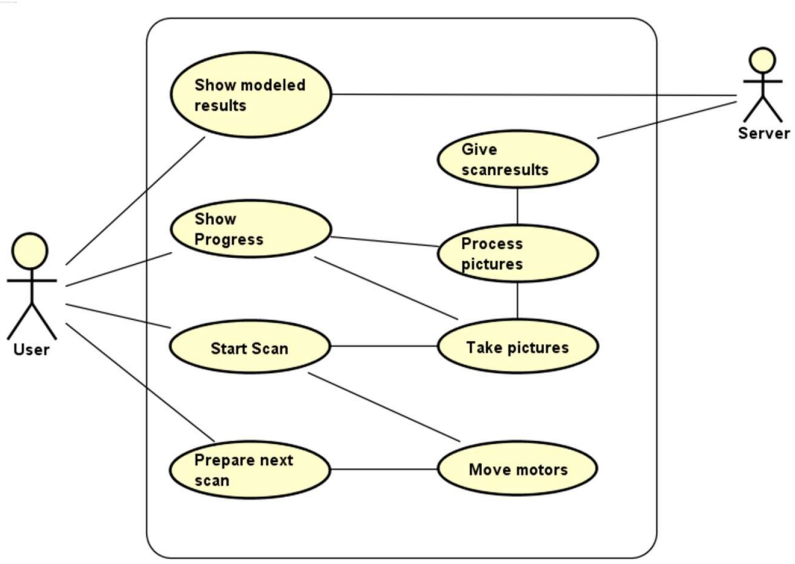
De opdrachtgever van het project, de begeleider van het project en de projectleden van vervolgfases van het project.

2 Architectuurontwerp

In dit hoofdstuk wordt het architectuurontwerp beschreven van het babyscansysteem op softwaregebied. Dit architectuurontwerp bestaat uit classes met interfaces welke beide uitgelicht zullen worden. Er is gekozen voor python, vanwege het hardware platform en bestaande software module die beschikbaar zijn.

2.1 Use Case Diagram

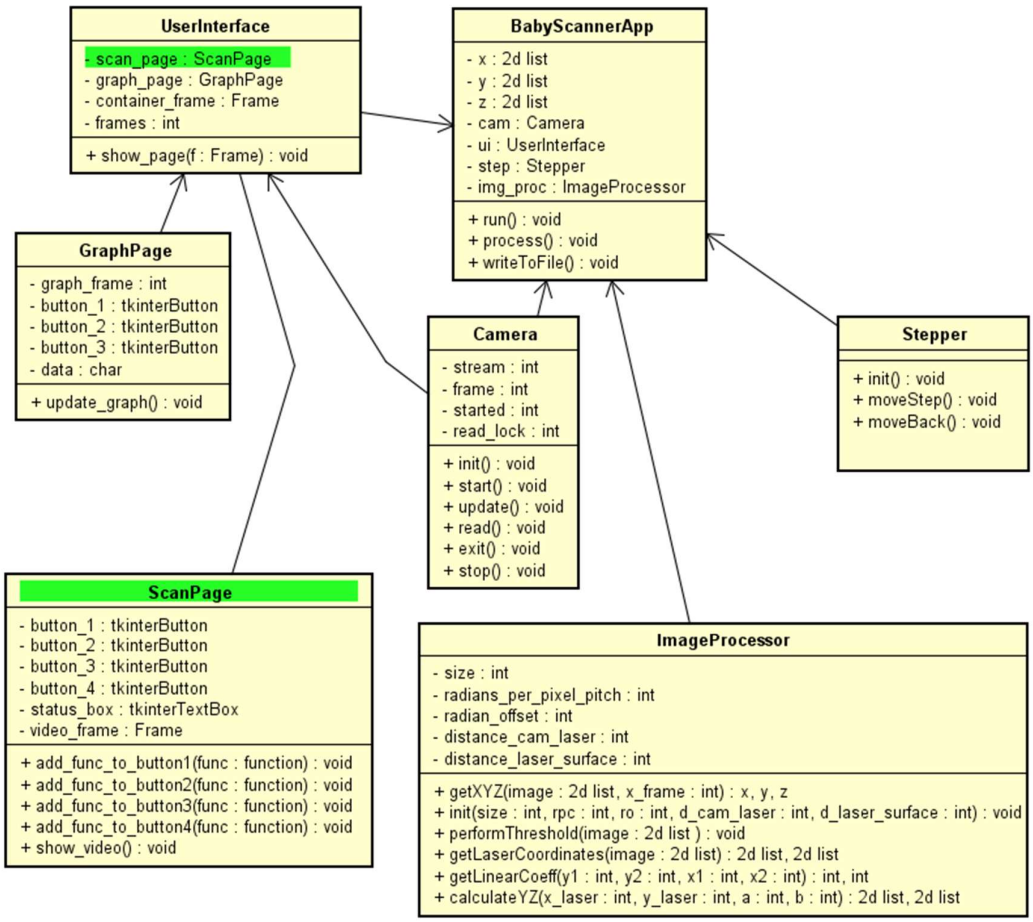
In figuur 1 valt te zien dat de gebruiker kan een scan starten. Wanneer dit gebeurt gaat het systeem de motoren bewegen en foto's nemen. Tijdens het nemen van foto's kan worden laten zien hoe ver de scan al compleet is aan de gebruiker. Wanneer alle foto's zijn genomen worden de foto's verwerkt en wordt er ruwe x-coördinaten geschreven en doorgegeven aan de server. Vervolgens komen er gemodelleerde resultaten terug van de server. Het systeem is in staat om dit weer te geven aan de gebruiker. In dit document wordt de server ook beschreven om de modellen te modelleren.



Figuur 1 Use Case diagram van het systeem

2.2 Class Diagram

Om figuur 1 te realiseren, wordt er gekeken naar wat voor classes of entiteiten deze functionaliteit voor elkaar kunnen krijgen. In figuur 2 staat een class diagram van het Babyscansysteem. Er is hiervoor gekozen dat de class UserInterface voor de interactie zorgt met de gebruiker. Hierdoor kan de gebruiker een scan starten, resultaten en voortgang bekijken en het systeem klaarzetten voor een volgende scan. Daarnaast is er een Stepper class die de motoren kan laten bewegen wanneer een scan is gestart of wanneer een volgende scan moet worden klaargezet. Wanneer een scan is gestart zorgt de Camera class dat er foto's kunnen worden genomen. Wanneer alle foto's zijn genomen dan gaat de ImageProcessor class de foto's analyzeren om xyz-data te genereren. De hoofdapplicatie class BabyScannerApp zorgt ervoor dat de interfaces met elkaar samenwerken, daarnaast geeft het de scanresultaten aan de server en geeft het de voortgang en de gemodeleerde resultaten door aan de UserInterface. In figuur 2 staat een overzicht van de classes in een Class Diagram.



Figuur 2 Class diagram van het systeem

2.3 Beschrijving van de classes

In deze paragraaf worden de verschillende deelsystemen een voor een beschreven.

2.3.1 ImageProcessor

Deze class zorgt ervoor dat een afbeelding kan worden verwerkt met een rode laserlijn en een zwarte achtergrond, zodat y en z-coördinaten van het object worden gegenereerd.

2.3.2 Camera

Deze class zorgt ervoor dat de camera wordt uitgelezen en afbeeldingen vanuit meerdere plekken in de code kan worden opgevraagd.

2.3.3 Stepper

Deze class zorgt ervoor dat de motorcontroller kan worden aangestuurd met commando's die via een seriële verbinding worden verstuurd.

2.3.4 UserInterface

Deze class zorgt ervoor dat de gebruiker de applicatie kan bedienen en systeeminformatie krijgt te zien.

2.3.5 BabyScannerApp

Deze class zorgt ervoor dat alle andere classes met elkaar samenwerken en daarnaast zorgt het voor het algemene verloop van het algoritme.

3 Detailontwerp

In dit hoofdstuk wordt het ontwerp verder uitgewerkt. De python applicatie wordt per class toegelicht en daarna het MATLAB script van de plotter op de server.

3.1 Image processor

De image processor is in staat om een afbeelding met de afmeting van 1280x720 waarop een rode lijn met een donkere ondergrond te zien is, om te zetten naar een lijst met X, Y en Z-coördinaten.

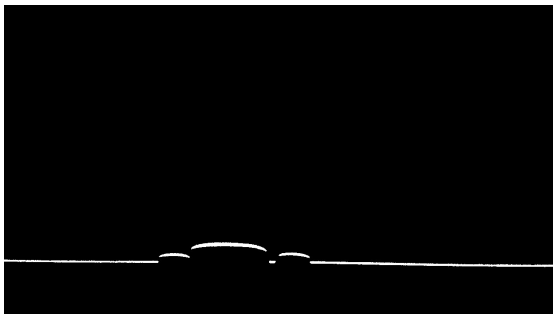
Hierbij gebruikt het de opencv module om een afbeelding te filteren. Daarnaast maakt het gebruik van de numpy module, zodat er matrix berekeningen gedaan kunnen worden over de afbeelding. Daarnaast gebruikt het ook de math module, omdat er goniometrische berekeningen worden gedaan. In figuur 3 staat een overzicht van alle methodes die bij deze class horen.

```python
import cv2
import numpy as np
import math
import matplotlib.pyplot as plt

# This class takes an image with a red laser line and
# calculates the height
class ImageProcessor:

    # Initializes the attributes
    def __init__(self, size = 256, ···

    # Returns the lists X, Y and Z after proceessing an image
    def getXYZ(self, image, x_frame): ···

    # Private function to filter the image, returns a filtered image
    def __performThreshold(self, image): ···

    # Private function Takes an filtered_image and gets the coordinates of the laser line in the mage
    def __getLaserCoordinates(self, image): ···

    # Private function gets coefficients from y = ax + b  to calculate baseline
    def __getLinearCoeff(self, y_1, y_2, x_1, x_2): ···

    # Private function calculates every Z value per Y value based on the laser lines and coefficients
    def __calculateYZ(self, x_laser, y_laser, a, b): ···
```

Figuur 3 Image processor class

3.1.1 Filteren

Wanneer een afbeelding binnen komt, wordt deze gefilterd met BGR2GRAY. Dit is een functie van opencv en converteert de afbeelding naar grayscale. Vervolgens wordt er gebruik gemaakt van het Otsu filter. Dit filter zoekt met een drempelwaarde, zodat de laserlijn in het wit kan worden gescheiden van de zwarte achtergrond. In figuur 4 staat de uitvoer na deze twee bewerkingen die in de methode ‘ __performTreshold’ worden uitgevoerd.



Figuur 4 Afbeelding na BGR2GRAY en Otsu

3.1.2 Laserlijn coördinaten bepalen

Na het filteren worden de x en y-coördinaten bepaald van de laserlijn. Dit wordt gedaan door de afbeelding op te splitsen is 256 segmenten van links naar rechts. Hierbij wordt in elk segment gezocht naar de witte pixels en de bijbehorende coördinaten. Van deze positionering wordt het middelpunt genomen. Het y-coördinaat wordt vervolgens met -1 vermenigvuldigd, omdat de matrix-vorm van 0 tot -720 gaat. Dit wordt uitgevoerd in de classmethode __getLaserCoordinates.

3.1.3 Basislijn bepalen

Nadat de x en y-coördinaten zijn bepaald, wordt de basislijn geformuleerd. De lijnlaser staat niet altijd op de afbeelding. Dit komt, omdat de behuizing van de scanner niet altijd stabiel staat. Een basislijn wordt opgesteld, zodat de instabiliteit geen factor zal zijn op het verwerken van de laserlijn. Dit kan worden gedaan met vergelijking 1.

$$y = ax + b \tag{1}$$

Vergelijking 1 geeft de standaard formule van een rechte lijn. Hier zijn 2 coördinaten voor nodig. Een hellingshoek a die de bepaald kan worden door een begin en eindpunt te nemen van de laserlijn, mede de offset b. Door dit te berekenen kan dit worden ingevuld om de basislijn te bepalen. Deze operatie wordt uitgevoerd in classmethode __getLinearCoeff.

3.1.4 Y en Z-coördinaten van het object bepalen

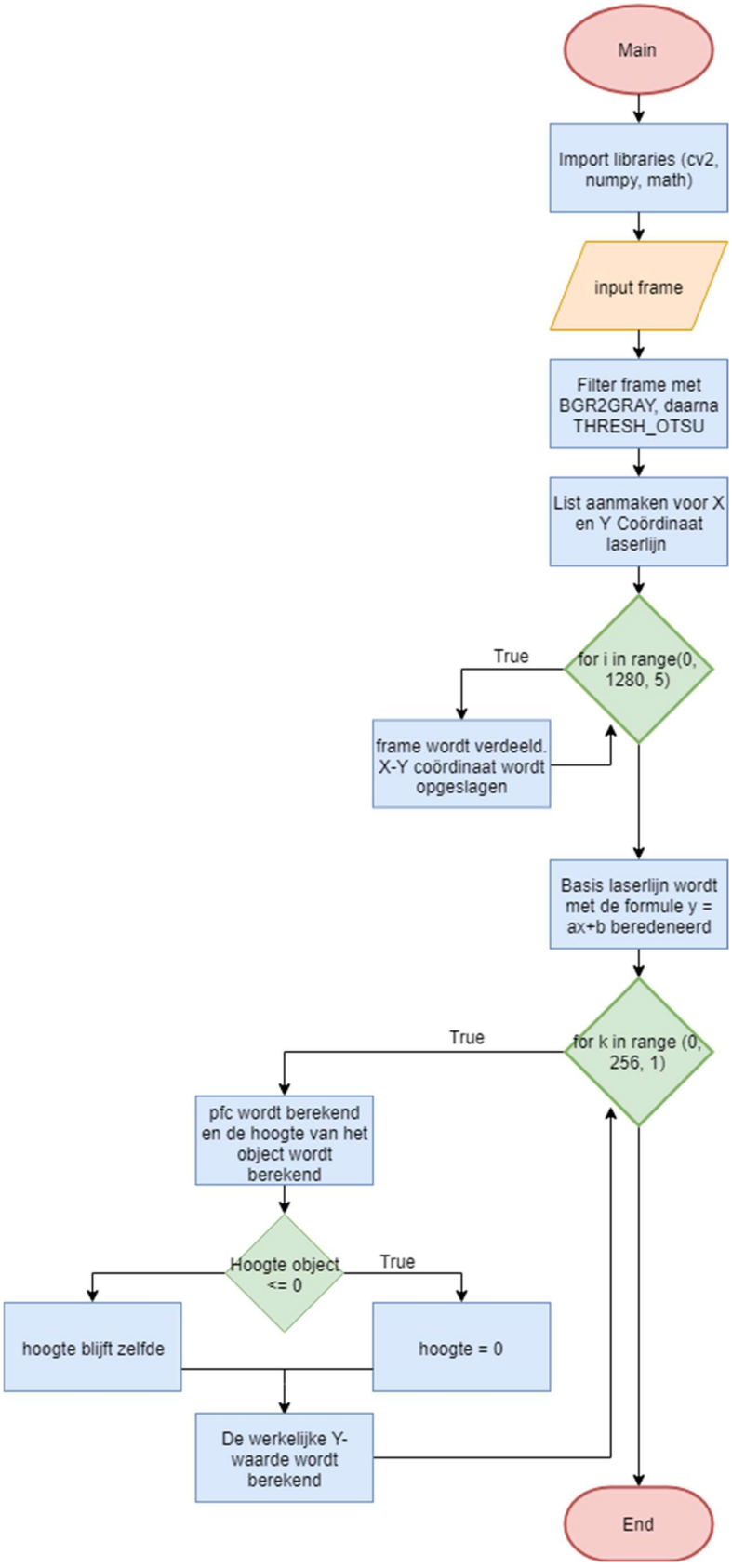
Met de coëfficiënten a en b kan de basislijn worden bepaald. Door de coördinaten van de basislijn af te trekken van de laserlijn resteert er een afstand van alles wat afwijkt van de basislijn. Dit staat gelijk aan de pfc (pixels from centre). Hiermee kan de hoogte worden bepaald door onderstaande formule 2.

$$D = 46{,}5 - \left(\frac{h}{\tan(pfc * 0{,}0014 + 0{,}2686)}\right) \tag{2}$$

Hierbij is h de afstand tussen de laser en de camera. D is de hoogte van het object dat gescand wordt. De D-waarde is het Z coördinaat wat wordt opgeslagen per y segment om op elke y waarde een Z-hoogte te krijgen. Dit is een waarde in millimeter. Om de y-waarde ook in millimeters te krijgen wordt dit berekend met de onderstaande vergelijking 3. Deze operatie wordt uitgevoerd in de classmethode __calculateYZ.

$$Y_{werkelijk} = Y_{frame} \cdot \left(\frac{56}{1280}\right) \cdot 10 \tag{3}$$

In figuur 5 staat bovenstaand algoritme schematisch weergegeven in een flowchart.



Figuur 5 Flowchart van Image Processor algoritme

3.2 Camera

Dit deelsysteem zorgt ervoor dat de applicatie gebruik kan maken van images. Dit deelsysteem gebruikt de opencv library om een camerastream op te zetten die continu blijft streamen en geeft de mogelijkheid om een image uit de buffer te kopiëren wanneer de applicatie dat opvraagt. Om dit parallel te laten gebeuren, wordt er gebruik gemaakt van een threading module met lock. Op deze manier kan de applicatie niet zomaar naar een ander stuk code gaan, wanneer de CameraStream class bezig is om een image op te halen of te kopiëren.

```
from threading import Thread, Lock
import cv2

class CameraStream:

    def __init__(self, src = 0, width = 1280, height = 720): ···

    def start(self): ···

    def update(self): ···

    def read(self): ···

    def stop(self): ···

    def __exit__(self, exc_type, exc_value, traceback): ···
```

Figuur 6 Camera class

3.2.1 Het opzetten van een camerastream

Bij het opzetten van de camerastream wordt de VideoCapture mode van opencv gebruikt. Dit wordt in een class attribuut gezet en de breedte en hoogte worden meegegeven naar specificaties van de camera. In dit geval 1280x720. Vervolgens wordt een flag die aangeeft of de stream is begonnen op false gezet. Zodat de camera pas begint met streamen zodra de methode start wordt aangeroepen.

3.2.2 Het updaten van de camerastream

Om te zorgen dat de stream niet direct stopt na 1 keer, wordt er een updatemethode gebruikt. In deze loop wordt er gekeken of de started flag staat gezet. Mits deze voorwaarde wordt voldaan, dan wordt een frame gelezen met de read methode van opencv. Vervolgens wordt een slot van de threading module gevraagd, zodat het gegenereerde frame veilig gekopieerd kan worden gekopieerd naar de class attribuut frame. Daarna wordt het slot weer vrijgegeven.

3.2.3 Het uitlezen van de camerastream

Andere modules maken niet direct gebruik van de stream, maar kunnen een frame uitlezen doordat een frame gekopieerd kan worden van de class attribuut frame. Dit wordt gedaan, nadat het slot is verkregen en daarna wordt het slot weer vrijgegeven van de thread.

3.3 Stepper

Dit deelsysteem zorgt ervoor dat commando's naar de stappenmotorcontroller kan worden gestuurd. Dit wordt gerealiseerd met de serial module. Hiermee kan een seriële verbinding worden opgezet en commando's worden verstuurd. Het programma stuurt een 'S' naar de motorcontroller om de stappenmotor te bewegen naar links van begin tot eind. Daarnaast kan het programma een 'R' sturen om de stappenmotor naar rechts te bewegen van eind tot begin.

```
import serial
import time
import threading

# This class controlls the stepper motor via a serial connection.
# Use moveStep to move the scanner forward
# Use moveBack to move the scanner back to its start position

class Stepper:

    # Creates serial instance and opens port
    def __init__(self, port, baudrate=115200, timeout=2): ···

    # Moves the stepper from one side to the other side
    def moveStep(self): ···

    # Moves the stepper back home
    def moveBack(self): ···
```

Figuur 7 Stepper class

3.3.1 Het opzetten van een seriële verbinding

Er wordt gebruik gemaakt van de module 'serial'. Een eigen class attribuut krijgt de eigenschappen van een serial class. Dit wordt aangeroepen door een serial object te maken met een seriële poort, baudrate en timeout. Dit wordt in dit geval geconfigureerd door de BabyScannerApp. Vervolgens wordt de er gekeken of de poort al open is. Is de poort open dan wordt deze gesloten en opnieuw geopend.

3.3.2 Het versturen van commando's

Bij het versturen van commando's wordt de seriële stream eerst leeggespoeld. Dit wil zeggen dat de characters die nog mogelijk in de buffer zijn blijf hangen worden weggehaald. Vervolgens wordt er gekeken naar hoeveel bytes er binnenkomen nadat het bericht 'S\n' wordt verstuurd. Wanneer dit ongelijk is aan nul dan geeft het de binnengekomen bericht terug als functie uitvoer. Eerst wordt het nog gedecodeerd, omdat het een byte array is en een UTF-8 encoded string als uitvoer is gewenst. Vervolgens controleert de hoofdapplicatie BabyScannerApp of dit bericht voldoet aan de voorwaarden.

3.4 User Interface

Dit deelsysteem is verantwoordelijk voor de interactie met de gebruiker. Een multi-page applicatie wordt opgezet met frames en knoppen zodat de gebruiker feedback krijgt van de camera en daarnaast zelf een scan kan starten en naar de scanresultaten kijken. Dit wordt gerealiseerd met de modules tkinter en PIL. Tkinter is een framework waar al volledige frames, knoppen, statusbars en andere widgets inzitten die zelf kunnen worden geplaatst en geconfigureerd.

```
import tkinter as tk
from tkinter import ttk
from tkinter import StringVar

import PIL
from PIL import Image
from PIL import ImageTk

import os
import cv2

LARGE_FONT = ("Verdana", 12)

# This class makes a container for multiple pages for the UI
class UserInterface(tk.Tk):
    def __init__(self, master): ···
    def show_frame(self, cont): ···

    class StartPage(tk.Frame):
        def __init__(self, parent, controller): ···
        def show_video(self): ···
        def add_function_to_button1(self,function): ···
        def add_function_to_button2(self, function): ···
        def add_function_to_button3(self,function): ···
        def add_function_to_button4(self,function): ···

    class PageOne(tk.Frame):
        def __init__(self, parent, controller): ···
        def update_graph(self):            ···

    class GraphTwo(tk.Frame):
        def __init__(self, parent, controller): ···
        def update_graph(self):            ···
```

Figuur 8 User Interface class

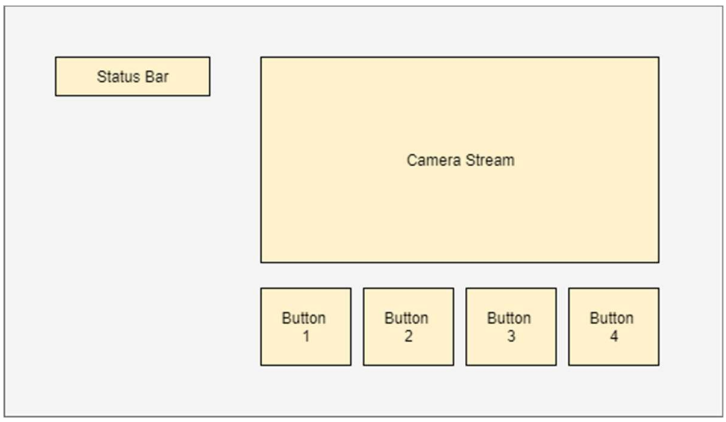
3.4.1 Het opzetten van een grafische userinterface

De mainclass UserInterface is van het type class Tkinter. Dit wordt geconfigureerd door een container aan te maken waarin de verschillende pagina's kunnen worden opgeslagen en met een functie kan een pagina 'naar boven worden gehaald'. Dit wilt zeggen dat het zichtbaar wordt en er interactie kan plaatsvinden met de widgets in een pagina.

Om de container te plaatsen wordt er gebruik gemaakt van de functie 'grid' binnen het Tkinter framework. Hiermee kan een raster worden gemaakt van cellen en kolommen om hierin vervolgens widgets te plaatsen. De container is een widget van het type frame.

3.4.2 De multi-page layout

Het ontwerp van de verschillende pagina's bestaat uit verscheidene widgets per pagina. In figuur 8, 9 en 10 staan ontwerpen van de pagina's. Hierin is de grijze buitenkant de container zoals beschreven in paragraaf 3.4.1

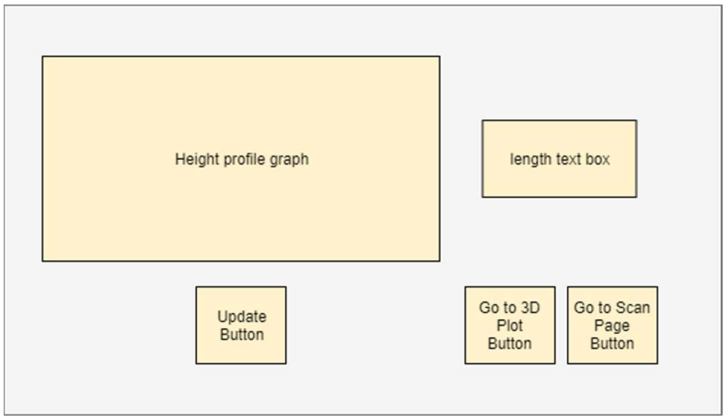


Figuur 9 Ontwerp van de scanpagina

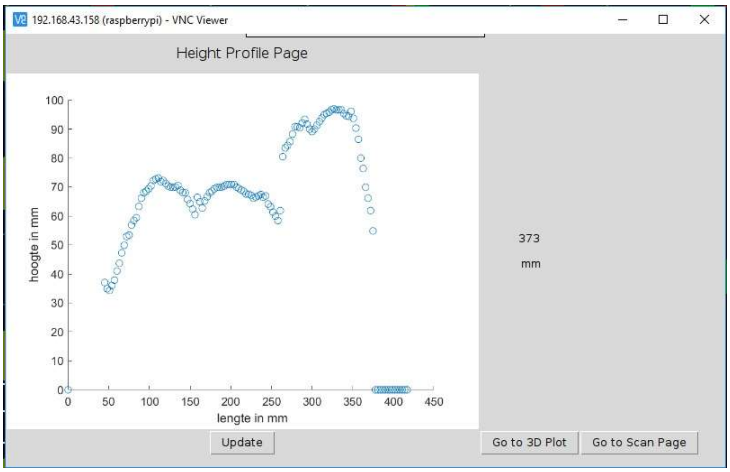
Elke widget wordt aangemaakt en met de functie grid op een plek in het raster gezet. Hier worden de dimensies meegegeven in pixels en de kleur van de widget. In figuur 8 bestaat het ontwerp uit 4 keer een button-widget, 1 keer een camera stream label en een status-bar label. De button-widgets krijgen een kleur mee, afmetingen en een functie die uitgevoerd wordt wanneer de button wordt ingedrukt.

Figuur 10 Gerealiseerde ontwerp van de scanpagina

In figuur 11 en 12 wordt een afbeelding doorgegeven vanaf de server om een grafiek weer te geven. Dit wordt gedaan binnen een frame. Met de updateknop kan er worden gekeken of er een nieuw bestand binnen is gekomen vanaf de server. Daarnaast kan er genavigeerd worden naar de andere resultaatpagina en naar de scanpagina.

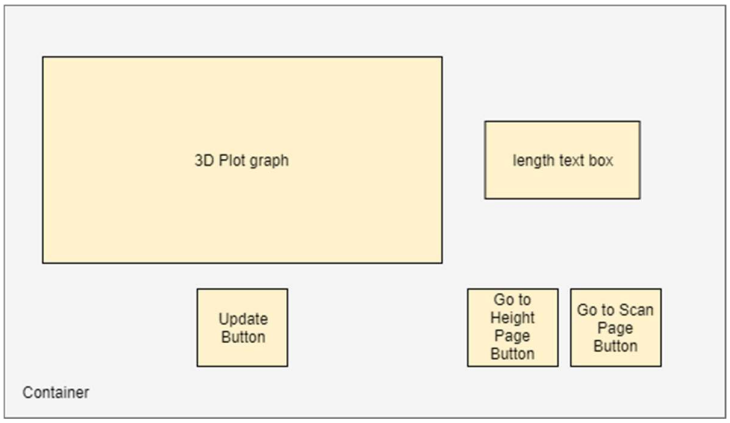


Figuur 11 Ontwerp van grafiekpagina met hoogteresultaten

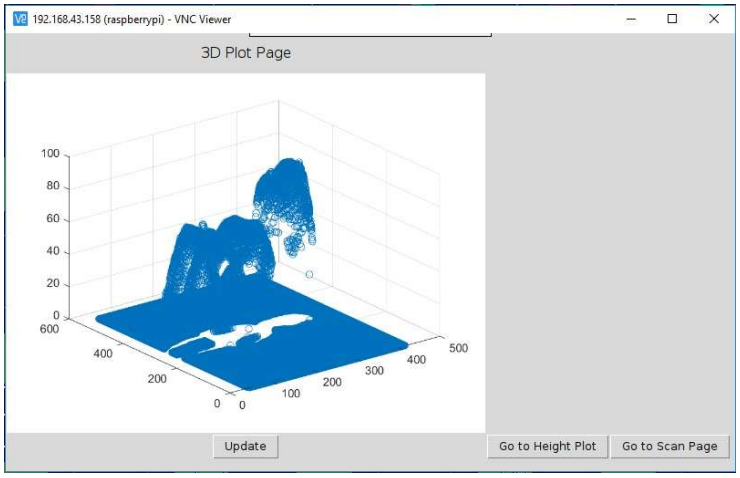


Figuur 12 Gerealiseerde ontwerp van de 2d grafiekpagina

In figuur 13 en 14 staat het ontwerp van de grafiekpagina met 3d-plotresultaten. Deze pagina, verschilt niet veel van de andere grafiekpagina. Het is slechts een andere afbeelding.



Figuur 13 Ontwerp van de grafiekpagina met 3d-plotresutaten



Figuur 14 Gerealiseerde resultaat van de 3d grafiekpagina

3.4.3 Een videostream verwerken in de userinterface

Een functie bij de paginaclass ScanPage heet show_video. Deze functie wordt gebruikt om periodiek met een interval het beeld te updaten met een afbeelding die klaar staat. Hiervoor wordt de camera class doorgegeven vanuit de hoofdapplicatie. Elke 10ms wordt de video geüpdatet. Er wordt gebruik gemaakt van opencv en pil, om een afbeelding te streamen in tkinter. Eerst wordt de afbeelding geschaald om binnen de frame te passen. Vervolgens wordt het geconverteerd met BGR2RGBA. Hier komt een RGB-array uit. Dit kan vervolgens weer met de pil module omgezet worden naar een tkinter image die in een label past. Hiervoor wordt de functie PhotoImage gebruikt.

3.4.4 Functies van buitenaf koppelen aan knoppen

Met de lambda kan er binnen python een functie worden meegegeven als parameter aan een functie mét argumenten. Dit wilt zeggen dat in de applicatie een functie van een andere class of module kan worden meegegeven aan een button. In figuur 7 valt te zien dat er vier functies zijn gemaakt die een functie als parameter meenemen. In deze functies wordt de knop opnieuw geconfigureerd met de nieuwe functie. Dit wordt gedaan zodat de overkoepelende applicatie zelf kan beslissen welke functie wordt meegegeven en dat dit niet wordt gedaan vanuit de UserInterface class.

3.5 BabyScannerApp

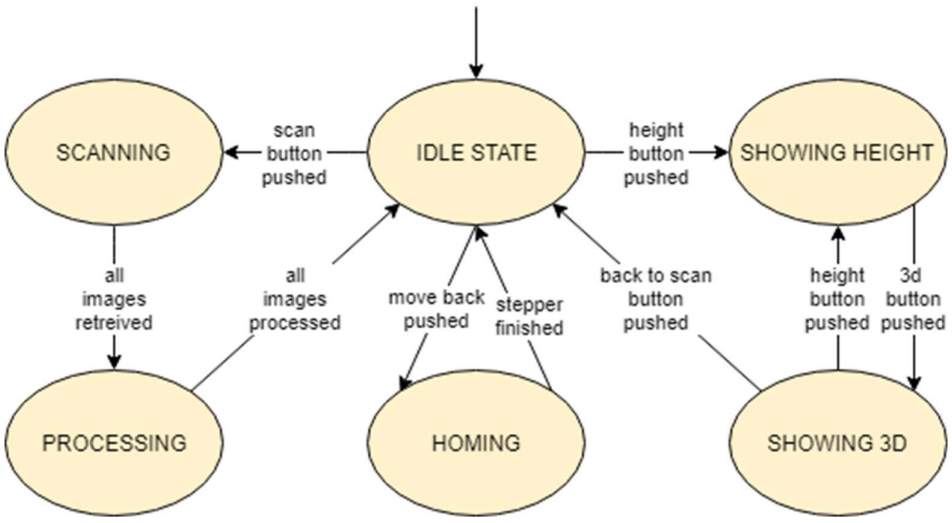
Dit deelsysteem zorgt ervoor dat frames kunnen worden geschoten met de camera. Daarnaast kan het de laser aan en uit zetten. Tot slot worden de laserlijnen in de frames verwerkt en door de kromming met triangulatie bepaald wat de diepte is.

```python
import os
import userinterface as gui
import stepper as step
import camera as cam
from imageprocessor import ImageProcessor

# The flag for when the interval function needs to stop
global stopFlag
stopFlag = True

global count
count = 0

class BabyScannerApp():

    def __init__(self, *args, **kwargs):...

    def restart(self):
        global stopFlag
        stopFlag = True

    def run(self):...

    def process(self):...

def main():...

if __name__ == "__main__":
    main()
```

Figuur 15 BabyScannerApp Class

3.5.1 State Machine

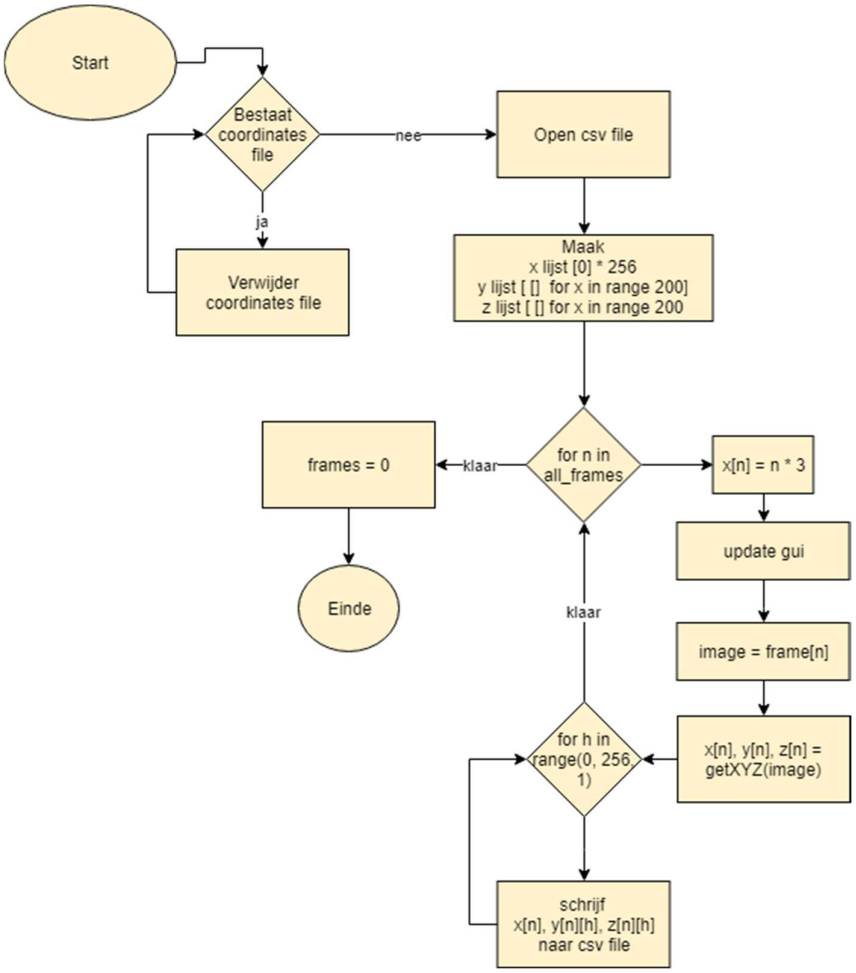
De BabyScannerApp werkt volgens een state machine. De werking hiervan staat schematisch weergegeven in figuur 10. De methode __init__ wordt uitgevoerd in 'IDLE STATE' en blijft in een loop de camera stream weergeven en wachten op input van de gebruiker. Wanneer de scanknop wordt ingedrukt, dan triggered dit een event en gaat het systeem over op de state 'SCANNING'. In deze staat wordt de motorcontroller aangestuurd en met een interval van 90ms een foto genomen en gekopieerd naar een lijst. Daarnaast wordt er naar de UserInterface gecommuniceerd bij welke frame het systeem nu momenteel is. Wanneer alle foto's/frames zijn genomen en gekopieerd genereert dit een



Figuur 16 Statemachine diagram van BabyScannerApp

Een event. Hierbij wordt de periodieke interval om een foto te nemen geannuleerd en een transitie gestart naar de staat ‘PROCESSING’.

In de staat ‘PROCESSING’ wordt er gekeken of een bestand met xyz-coördinaten al bestaat. Is dit het geval, dan wordt dit verwijderd en een nieuw bestand gemaakt, zodat er niet met oude data wordt gewerkt. Vervolgens wordt er geïtereerd door elke afbeelding. Per afbeelding wordt de image processor aangeroepen om van de afbeelding een lijst met xyz-coördinaten te maken. Deze lijst wordt naar het csv bestand geschreven. In figuur 11 staat een flowchart van dit principe schematisch weergeven. Wanneer alle afbeeldingen zijn verwerkt zal het systeem weer terug naar de staat ‘IDLE’ gaan.



Figuur 17 Flowchart van Processing State

In de staat ‘IDLE’ kan nu ook ‘move back’ worden gedrukt. Hierdoor gaat het systeem naar ‘HOMING’ staat. In deze staat wordt de motorcontroller aangestuurd om terug te bewegen naar het begin. Wanneer dit commando is verstuurd en succesvol is ontvangen, gaat het systeem weer naar de staat ‘IDLE’.

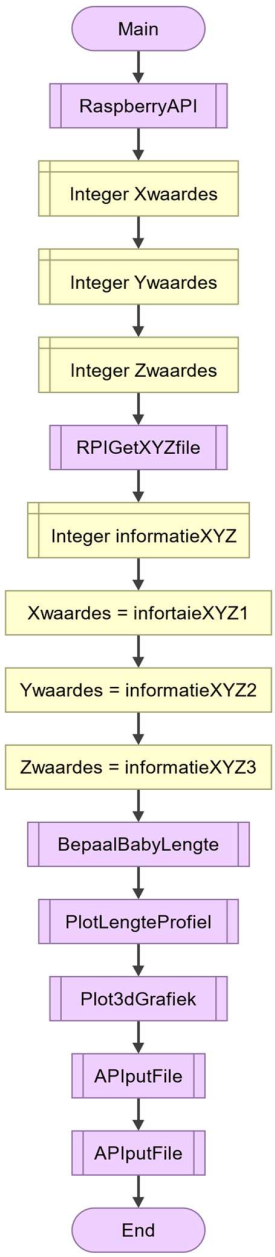
In de staat ‘IDLE’ kan ook naar de resultatenpagina worden genavigeerd. Hierbij wordt de andere user-interface aangeroepen waarop de grafiek staat weergeven met het hoogteprofiel. Vanuit deze pagina kan er ook weer worden genavigeerd naar de 3d-plotpagina en terug naar de scanpagina.

3.6 Plotter

In figuur 15 staat de flowchart van de plotter software. Er wordt gebruik gemaakt van MATLAB. In MATLAB kan er gebruik gemaakt worden van de RaspberryAPI, hiermee kan er een verbinding tussen MATLAB en Raspberry Pi worden gesteld.

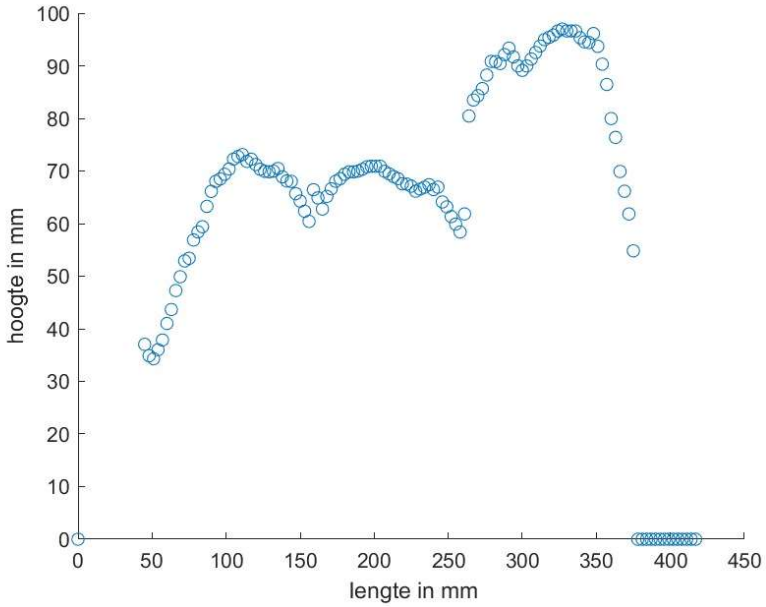
Vervolgens worden drie variabelen aangemaakt om de xyz-waarden uit het opgevraagde csv bestand op te slaan. Daarna wordt de functie RPI-GetXYZfile aangeroepen. Met deze functie kan er data vanaf de Raspberry Pi gehaald en worden en in MATLAB worden geladen. Met de functie 'bepaalbabylengte' kan de lengte van de baby worden bepaald. Dit wordt gedaan met behulp van een for-loop. In deze loop wordt er gekeken naar elke Z-waarde per X-waarde. Op één x-coördinaat liggen meerdere y en z-waardes. Om een duidelijke 2d-grafiek te plotten moet eerst de maximale waardes worden bepaald. Er wordt daarna gekeken bij welke x-waarde, de z-waarde voor het eerst steeg boven de drempel-waarde. Dit kan ook gedaan worden voor de laatste x-waarde waarbij de z-waarde boven de drempelwaarde was. Door deze twee waardes van elkaar af te trekken kan de lengte van de baby bepaald worden. In de functie 'plotlengteprofiel' wordt er een 2d-grafiek geplot. In deze grafiek kan het lengteprofiel worden bekeken van het object. Vervolgens wordt er in de functie 'plot3dgrafiek' een 3d-plot gemaakt van de data.

Tot slot worden deze twee afbeeldingen teruggestuurd naar de Raspberry Pi met de functie APIputFile. Zodat deze door de UserInterface kan worden weergegeven.

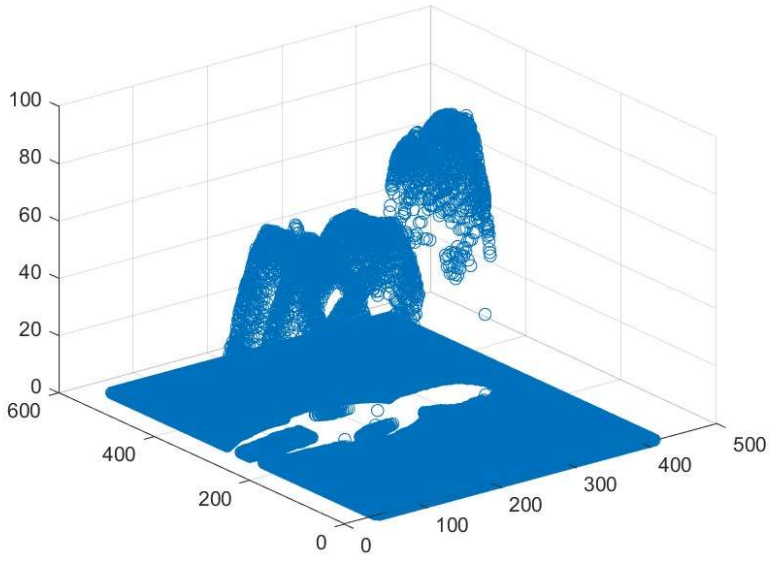


Figuur 18 Flowchart Plotter Script

In figuur 19 staat het gegenereerde hoogteprofiel van een babypop. Dit is het resultaat van het matlab script, wanneer een csv bestand met xyz data wordt verwerkt. In figuur 20 staat het 3d beeld.



Figuur 19 Hoogteprofiel



Figuur 20 3d plot van gescande baby