UNIVERSIDADE
LUSÓFONA

GIMP3D

Trabalho Final de curso

Relatório Final

Filipe Coutinho

João Lima

Rui Ribeiro

João Carvalho

Trabalho Final de Curso | LEI | 01/07/2022

www.ulusofona.pt

Direitos de cópia

Resumo

A unidade curricular do trabalho final de curso tem como objetivo criar uma plataforma de monitorização de impressoras 3D de construção de casas (GIMP3D), proposta feita pela empresa Yucoders.

Durante este período existem várias etapas a cumprir, desde a programação em arduinos, até ao relatório final, o que torna este trabalho num fluxo contínuo e organizado por etapas que têm de ser planeadas e geridas de acordo com os requisitos a cumprir.

As impressoras 3D de casas vêm inovar o futuro da construção civil permitindo aumentar a velocidade de construção e a diminuição de custos, a solução do nosso TFC tem como finalidade monitorizar através de sensores o consumo de materiais utilizados ao longo dessas construções.

Abstract

The curricular unit of the final course work aims to create a monitoring platform for house building 3D printers (GIMP3D), proposal made by the company Yucoders.

During this period there are several steps to be taken, from arduinos programming to the final report, which makes this work a continuous and organized flow in stages that must be planned and managed according to the requirements to be met.

The house 3D printers come to innovate the future of civil construction allowing to increase the speed of construction and the reduction of costs, the solution of our TFC aims to monitor through sensors the consumption of materials used throughout these constructions.

Índice

Resumo iii

Abstract iv

Índice v

Lista de Figuras viii

Lista de Tabelas ix

1 Identificação do Problema 1

1.1.1 Porque queremos calcular o volume de betão? 1

1.1.2 Porque queremos calcular a consistência do betão? 1

1.1.3 Porque queremos saber o stock de agregados disponíveis? 1

1.2 Análise Comparativa 2

Levantamento e análise de Requisitos 3

1.3 – Requisitos Funcionais 3

1.3.1 Requisito 1 (RF1) – Obter dados para arduino através de sensores 3

1.3.2 Requisito 2 (RF2) – Enviar dados do arduino para uma REST API 3

1.3.3 Requisito 3 (RF3) – Obter JWT 3

1.3.4 Requisito 4 (RF4) – Calcular o volume de agregados 3

1.3.5 Requisito 5 (RF5) – Calcular o volume do betão 3

1.3.6 Requisito 6 (RF6) – Calcular a consistência do betão 3

1.3.7 Requisito 7 (RF7) – Adicionar dados recolhidos pelos microcontroladores na base de dados 4

1.3.8 Requisito 8 (RF8) – Obter imagens da camara localizada na impressora 4

1.3.9 Requisito 9 (RF9) – Calcular área de impressão através de processamentos de imagem 4

1.3.10 Requisito 10 (RF10) – Criar REST API Python para correr o algoritmo de cálculo de área de impressão 4

1.4 – Requisitos Não-Funcionais 4

1.4.1 Requisito 1 (RNF1) – A REST API será escrita em .NET Core 4

1.4.2 Requisito 2 (RNF2) – Hospedar a API na AWS 4

1.4.3 Requisito 3 (RNF3) – O envio de dados para a API deverá utilizar JWT 4

1.4.4 Requisito 4 (RNF4) – O envio de dados para a API deverá ser feito no formato JSON 4

1.4.5 Requisito 5 (RNF5) – Para processamento de imagem deve ser utilizado a biblioteca OpenCV 4
1.5 Requisitos Implementados 5
2 Viabilidade e Pertinência 6
2.1 Resultado Final 6
2.1.1 Cálculo do volume dos tanques 6
2.1.2 Cálculo do volume de betão 6
3 Solução Desenvolvida 7
3.1 Cálculo do volume do betão – Objetivo Inicial 7
3.2 Cálculo do volume do betão – Solução Desenvolvida 8
3.2.1 Passos do algoritmo 9
3.3 Cálculo da consistência do betão 13
3.4 Cálculo do volume de agregados 14
3.4.1 Volume de água 14
3.4.2 Volume de sólidos 15
3.4.3 Fluxo de execução 16
3.5 Microprocessor e sensors 18
3.5.1 Arduino NODEMCU (ESP32) 18
3.5.2 Sensor ultrassónico 18
3.5.3 Sensor fotoelétrico 18
3.6 Arquitetura 19
4 Benchmarking 20
4.1 Sensores 20
4.2 Impressão 3D 20
4.3 Cálculo de área de impressão 21
5 Método e planeamento 22
6 Resultados 23
6.1 Cálculo da área de impressão 23
6.2 Cálculo do volume de materiais em tanques 23
Bibliografia 25
Glossário 26
Anexo 1 – Modelo de Dados 27
Anexo 2 – Plano de Trabalho Gantt 28

Anexo 3 – Resultados do cálculo da área 29

Lista de Figuras

Figura 1 - Impressora 3d COBOD 1
Figura 2 – Montagem da impressora 6
Figura 3 - Primeira impressão concluída 6
Figura 4 - Polígono Irregular 7
Figura 5 - Exemplo marcador aruco 8
Figura 6 - Caixa aberta Figura 7 - Caixa Fechada 8
Figura 8 - Obtenção de *Frames* 9
Figura 9 - Função de recorte 9
Figura 10 - Caixa Aberta Recortada 9
Figura 11 – Contornos Figura 12 - Canny Edges 10
Figura 13 - Diminuição de ruido 11
Figura 14 - Bounding box exterior 11
Figura 15 - Função para ajuste de cantos 12
Figura 16 - Conversão de coordenadas 12
Figura 17 - Procura de altura 13
Figura 18- Cálculo de área através dos cantos de um retângulo 13
Figura 19 - Diagrama UML Tanques 14
Figura 20 - Diagrama de tanque de agregados 15
Figura 21 - Ficheiro config.h 16
Figura 22 - Volume Controller 17
Figura 23 - Cálculo de volume para VerticalOvalTank 17
Figura 24 – NodeMCU 18
Figura 25 - Jsn-sr04t 18
Figura 26 - GY-VL53L0XV2 L53L0X TOF 18
Figura 27 - Arquitetura 19
Figura 28 - 3DMultiVision 20
Figura 29 - 3DLevelScanner S 20
Figura 30 – Comunidade conceptual criada por ICON 20
Figura 31 - Dimensões reais da caixa 23
Figura 32 - Resultados de teste 23
Figura 33 - Modelo de dados 27
Figura 34 - Plano de trabalho Gantt 28
Figura 35 - Caixa Fechada 29
Figura 36 - Caixa aberta 29
Figura 37- Caixa Aberta 29
Figura 38- Caixa Aberta 30
Figura 39 - Caixa fechada 30
Figura 40 - Caixa Aberta 30

Lista de Tabelas

Tabela 1 - Requisitos Implementados 5
Tabela 2 - Plano de trabalho 22

1 Identificação do Problema

Com este trabalho pretende-se fazer a monitorização de impressoras 3D de construção de casas (Figura 1). Esta monitorização será realizada através do cálculo de propriedades nestas construções.
Nas impressões queremos monitorizar a consistência e o volume do betão utilizado em cada casa, vamos também fazer a gestão de stock dos agregados utilizados para fazer o betão.

Figura 1 - Impressora 3d COBOD

1.1.1 Porque queremos calcular o volume de betão?

Queremos calcular o volume de betão utilizado para permitir obter o custo real e detalhado da casa.
Com a receita do betão podemos calcular o volume de agregados utilizado e comparar este resultado com o volume real de agregados utilizado. Quanto maior for a diferença maior são as perdas de material que ocorrem desde a mistura até à impressão.

1.1.2 Porque queremos calcular a consistência do betão?

Queremos monitorizar a consistência do betão para que não haja falhas na impressão da casa. Se o betão estiver com uma consistência muito elevada pode causar um entupimento no braço da impressora e esta falha levaria a custos adicionais de manutenção. No caso de a consistência do betão ser muito baixa poderá comprometer a estabilidade da casa, estando a utilizar recursos essenciais para a construção de outras casas e afetando assim os lucros expectáveis.

1.1.3 Porque queremos saber o stock de agregados disponíveis?

Queremos acompanhar a quantidade de agregados disponíveis nos silos para saber a quantidade de material utilizado.
Ao calcular esta quantidade poderemos saber e prever quando precisaremos de fazer *restock* do mesmo, adicionalmente sabemos a quantidade de material utilizada para fazer uma impressão com um volume, o que nos permite saber se existiram perdas de material.

1.2 Análise Comparativa

Chegando à conclusão do trabalho percebemos que existiram alguns aspectos do nosso projecto que não conseguimos implementar por falta de tempo, tais como o cálculo da consistência de betão e o encapsulamento do algoritmo de cálculo de área de impressão de forma que fosse possível chamar o algoritmo através de uma chamada à API.

Apesar de no problema inicial não haver referência à criação de uma API em Python esta alteração teve de ser efetuada porque a nossa REST API inicial está escrita em .NET Core, *framework* essa que não tem suporte ao OpenCV.

Levantamento e análise de Requisitos

1.3 – Requisitos Funcionais

1.3.1 Requisito 1 (RF1) – Obter dados para arduino através de sensores

Para realizar os cálculos de volume de betão e agregados (RF5 e RF6) será necessário fazer uma recolha de dados.

Para recolha dos dados foram usados sensores ligados aos arduinos para realizar a tarefa de obter os dados necessários para o cálculo dos volumes (RF5 e RF6).

1.3.2 Requisito 2 (RF2) – Enviar dados do arduino para uma REST API

Após a recolha dos dados através de sensores (RF1), precisamos de os enviar para uma REST API onde será realizado os cálculos do volume de betão e de agregados (RF5 e RF6).

O arduino que estamos a utilizar é um NODEMCU (ESP32) que contém WiFi permitindo que os dados recolhidos sejam enviados via WiFi para a REST API.

1.3.3 Requisito 3 (RF3) – Obter JWT

Para receber um JWT o arduino deverá fazer um pedido de autenticação ao servidor com *user* e *password*. Após a verificação de credenciais, se estas forem válidas o servidor responderá com um JWT que pode ser usado pelo arduino para enviar os dados para o servidor de forma segura.

1.3.4 Requisito 4 (RF4) – Calcular o volume de agregados

Após a REST API receber os dados do arduino via WiFi (RF2) estes serão adicionados na base de dados e usados para o cálculo do volume de agregados.

1.3.5 Requisito 5 (RF5) – Calcular o volume do betão

Ao receber uma imagem enviada da câmara localizada na impressora e os dados do sensor localizado no braço da impressora, o servidor irá processar a imagem para saber as medidas da impressão que em conjunto com os dados do sensor irá permitir o cálculo do volume de betão.

1.3.6 Requisito 6 (RF6) – Calcular a consistência do betão

Ao receber uma imagem enviada da câmara localizada na impressora, o servidor irá processar a imagem e efetuar os cálculos descritos em 3.3 de modo a obter um valor para a consistência do betão.

1.3.7 Requisito 7 (RF7) – Adicionar dados recolhidos pelos microcontroladores na base de dados

Ao receber um pedido POST no *endpoint* adequado, o servidor avalia os dados recebidos e determina a informação que deverá ser introduzida na(s) tabela(s).

1.3.8 Requisito 8 (RF8) – Obter imagens da camara localizada na impressora

O servidor REST deverá ter um *endpoint* onde será possível receber imagens da camara.

1.3.9 Requisito 9 (RF9) – Calcular área de impressão através de processamentos de imagem

De modo a obter a área de impressão devemos aplicar um conjunto de filtros e funções nas imagens recebidas pela camara.

1.3.10 Requisito 10 (RF10) – Criar REST API Python para correr o algoritmo de cálculo de área de impressão

Como a nossa REST API está escrita em .NET Core necessitamos de um servidor que corra em python o algoritmo desenvolvido. Este servidor B deve conseguir responder a pedidos de área vindo do servidor A.

1.4 – Requisitos Não-Funcionais

1.4.1 Requisito 1 (RNF1) – A REST API será escrita em .NET Core

Para facilidade de integração nos serviços já existentes da empresa Yucoders decidimos adotar a *framework* .NET Core, pois esta é já atualmente utilizada pela empresa.

1.4.2 Requisito 2 (RNF2) – Hospedar a API na AWS

Optámos por hospedar a API na AWS visto que a empresa já tem serviços nesta plataforma.

1.4.3 Requisito 3 (RNF3) – O envio de dados para a API deverá utilizar JWT

Utilizamos JWT de forma a assegurar a segurança da API, protegendo assim o nosso servidor contra interceções de dados.

1.4.4 Requisito 4 (RNF4) – O envio de dados para a API deverá ser feito no formato JSON

1.4.5 Requisito 5 (RNF5) – Para processamento de imagem deve ser utilizado a biblioteca OpenCV

1.5 Requisitos Implementados

Requisito	Implementado?
RF1	Sim
RF2	Sim
RF3	Não
RF4	Sim
RF5	Sim
RF6	Não
RF7	Sim
RF8	Não
RF9	Sim
RF10	Não
RNF1	Sim
RNF3	Não
RNF4	Sim
RNF5	Sim

Tabela 1 - Requisitos Implementados

Para a continuidade do trabalho deveriam ser implementados os requisitos em falta. Poderia ser feita alguma filtragem de dados utilizando *data science* e algoritmos de ML tanto nos dados recebidos dos arduinos como nos volumes calculados de forma a melhorar a precisão das soluções.

2 Viabilidade e Pertinência

Na realização deste Trabalho Final de Curso recorremos ao uso de várias tecnologias e hardware especificados no ponto no ponto 3.5. As tecnologias que serão implementadas não deverão trazer custos para a universidade pois serão Open-Source.
No caso, haverá algumas plataformas que carregam custos, porém estas já são atualmente utilizadas pela empresa em parceria - Yucoders.
O hardware que será utilizado especificado no ponto 3.5, terá custo mínimo comparativamente a outras soluções encontradas no mercado no ponto 4.
Este problema foi-nos apresentado pela empresa Yucoders, e a solução descrita foi aceite pela mesma. A solução será posteriormente utilizada numa aplicação de gestão e suporte à decisão
No futuro poderá ser implementada alguma ciência de dados aos valores obtidos pelos sensores de forma a eliminar *outliers* e melhorar a precisão dos resultados.
Podemos observar em https://youtu.be/O8AiK2mLbAU a primeira casa impressa em África. Esta construção utiliza apenas materiais locais e betão, e vem trazer a solução para a falta de habitação de milhões de habitantes por um baixo custo.
Na Figura 2 podemos observar a montagem da impressora após a impressão de uma parede para testar o betão.

Figura 2 – Montagem da impressora

Figura 3 - Primeira impressão concluída

2.1 Resultado Final

2.1.1 Cálculo do volume dos tanques

Apesar da estrutura base estar desenvolvida esta componente terá de passar por vários testes fisicos e presenciais com o material envolvido antes de ser posta em “produção”, estes testes serão efetuados pela YuCoders.

2.1.2 Cálculo do volume de betão

O algoritmo desenvolvido serve como prova de conceito para o cálculo do volume de betão utilizado, porém ainda existe algum espaço para melhorias e solução de bugs, tal como 2.1.1 necessita de testes reais em condições verdadeiras ao ambiente onde será instalado.

3 Solução Desenvolvida

Consideramos que entre as soluções desenvolvidas a que requereu mais esforço e foi mais inovadora foi o tópico 3.2, isto porque pois não encontrámos nenhuma implementação ou versão semelhante ao problema resolvido.

Nas soluções apresentadas aplicámos conhecimento das disciplinas de Computação Gráfica, Arquiteturas de Computadores Avançadas, Base de Dados e Computação Distribuída. Isto porque as nossas soluções são um conjunto de servidores e microprocessadores que comunicam entre si, e guardam os dados em base de dados. Relacionamos também a computação gráfica pois foram necessários alguns conceitos base para o 3.2 que foram lecionados nesta cadeira.

3.1 Cálculo do volume do betão – Objetivo Inicial

Iremos calcular o volume de betão de uma impressão através da multiplicação da sua área pela sua altura.
Para podermos calcular a área da impressão necessitamos de colocar 4 marcadores aruco (Figura 5) no chão da impressora a uma distância fixa conhecida, que nos permitirão fazer uma escala de metros para pixel a partir de uma fotografia superior da impressão.
Para obter a altura da impressão vamos colocar um sensor ultrassónico no bocal da impressora.
Para medir a área da impressão vamos utilizar a fórmula de Pick.

$$A = I + \frac{B}{2} - 1$$

Onde I são os pontos que estão dentro da figura (*interior lattice points*) e B são os pontos que pertencem à borda (*boundary lattice points*).

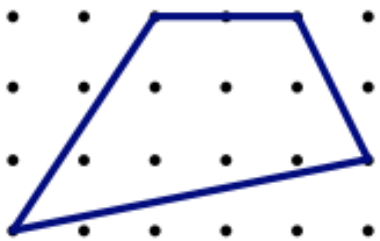

Figura 4 - Polígono Irregular

Por exemplo na - Polígono Irregular Figura 4 - Polígono Irregular podemos observar que I = 7 e B = 5, logo se a distância entre pontos for igual a 1 a área será A = 8,5.

O processo será o seguinte:

- Colocar os marcadores nos 4 cantos da impressora;
- Fotografar de um plano superior a impressão;
- Calcular escala px-m (pixéis para metro);
- Processar imagem para revelar contornos da impressão;
- Medir a área em pixéis através de contornos;
- Calcular área real através da escala obtida;
- Medir altura atual do bocal (altura da impressão);
- Multiplicamos a altura da impressão pela área total, obtendo o volume de betão.

Figura 5 - Exemplo marcador aruco

3.2 Cálculo do volume do betão – Solução Desenvolvida

Após alguns testes desenvolvidos percebemos que não seria possível fazer a implementação do algoritmo recorrendo à utilização de câmaras normais, isto porque devido a condições demasiado variáveis, seria bastante difícil diferenciar as paredes de ruídos na imagem como por exemplo sombras para fazer o "recorte das paredes".

Para resolver este problema encontrado utilizámos a camara Intel Real Sense D435i, a utilização desta camara resolve o problema fazendo uso da estereoscopia tendo uma noção de profundidade, permitindo-nos assim saber a distância real da lente a cada pixel na imagem.

Para o desenvolvimento de uma prova de conceito, utilizamos duas caixas para simular as paredes de uma construção.

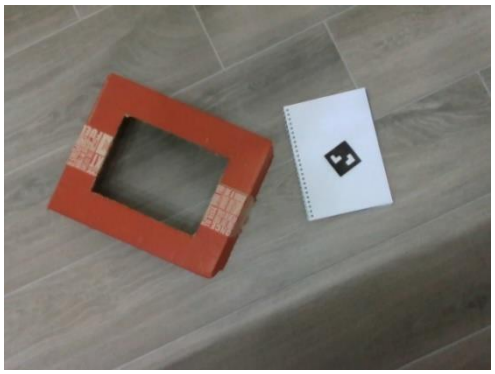

Figura 6 - Caixa aberta

Figura 7 - Caixa Fechada

3.2.1 Passos do algoritmo

3.2.1.1 Obter imagens

Através da camara podemos obter dois *frames* de uma imagem: *color frame*, *depth frame*.

Color frame é a imagem em si, enquanto o *depth frame* é uma matriz de distâncias para cada posição nesta.

Queremos também obter a *depth scale* que é uma escala utilizada pela *intel* para converter os valores disponíveis no *depth frame* para metros

```
def detect_live():
    # For camera stream
    dc = RealSense()
    while True:
        ret, depth_image, color_image = dc.get_frames()
```

Figura 8 - Obtenção de *Frames*

3.2.1.2 Encontrar objeto mais perto da lente e recortar a imagem para utilizar apenas esse objeto

```
def clip(color_image, depth_image, depth_scale):
    non_zero_depth = depth_image[np.nonzero(depth_image)]
    min_value = np.amin(non_zero_depth) / 1000  # in meters
    threshold = 0.10  # 10cm

    clipping_distance_in_meters = min_value + threshold if min_value > 0.2 else 0.2

    clipping_distance = clipping_distance_in_meters / depth_scale

    white_color = 255
    depth_image_3d = np.dstack(
        (depth_image, depth_image, depth_image))  # depth image is 1 channel, color is 3 channels
    bg_removed = np.where((depth_image_3d > clipping_distance) | (depth_image_3d <= 0), white_color, color_image)
    return bg_removed
```

Figura 9 - Função de recorte

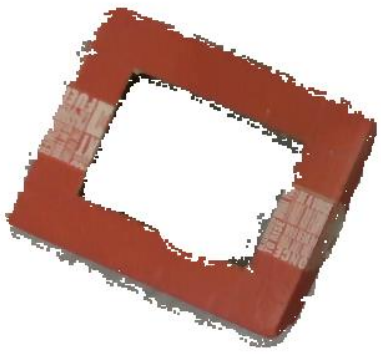

Figura 10 - Caixa Aberta Recortada

Para encontrarmos o objeto mais perto da camara vamos verificar qual o menor valor na matriz *depth frame* (excluindo zeros) e adicionamos um pequeno *threshold* para o recorte. Podemos ver na Figura 10 apenas a parte de cima da caixa sendo esta a parte mais perto da lente e o único objeto sendo mostrado na foto.

3.2.1.3 Normalizar a imagem

Para diminuir erros na deteção de contornos fazer converter a imagem para preto e branco e aplicar um desfoque através de um filtro chamado *GaussianBlur* que permite reduzir o ruído e os detalhes da imagem.

3.2.1.4 Obter contornos

Para a obtenção de contornos vamos utilizar duas funções disponíveis no OpenCV:

- ***findContours*** – Função que permite encontrar contornos de um objeto em um fundo preto ou branco, detetando a mudança na cor da imagem e marcando-a como contorno.
- ***Canny*** – Função que permite detetar contornos numa imagem.

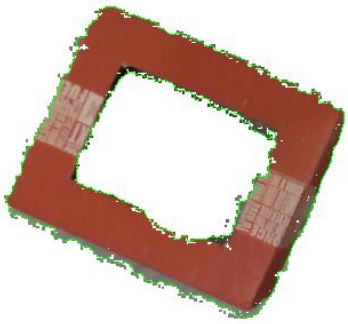

Figura 11 – Contornos

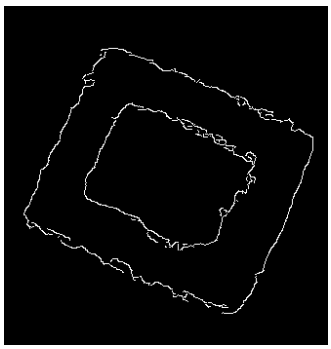

Figura 12 - Canny Edges

3.2.1.5 Verificar se se trata se de uma caixa aberta

Para obtermos a área total teremos de separar a imagem em duas caixas, obtendo a área total subtraindo à área da caixa exterior a área do buraco.

Ficamos a saber se a caixa tem um buraco se existir um contorno interior, sabemos esta informação através do retorno da função *findContours* que executamos no ponto 3.2.1.4.

3.2.1.6 Diminuir o ruido

Podemos ainda tentar diminuir mais o ruido através de interpolação

```
def smooth_contour(contour):
    x, y = contour.T
    # Convert from numpy arrays to normal arrays
    x = x.tolist()[0]
    y = y.tolist()[0]

    tck, u = splprep([x, y], u=None, s=1.0, per=1)

    u_new = np.linspace(u.min(), u.max(), 40)

    x_new, y_new = splev(u_new, tck, der=0)

    # Convert it back to numpy format for opencv to be able to display it
    res_array = [[[int(i[0]), int(i[1])]] for i in zip(x_new, y_new)]
    return np.asarray(res_array, dtype=np.int32)
```

Figura 13 - Diminuição de ruido

3.2.1.7 Criação de uma bounding box

Uma *bounding box* é a caixa com o mínimo de área para que contenha todos os nossos contornos encontrados, criamos uma box tanto para o buraco como para o perímetro exterior da caixa.

Foi preciso utilizar esta caixa para saber quais os seus cantos e em que coordenadas se encontram.

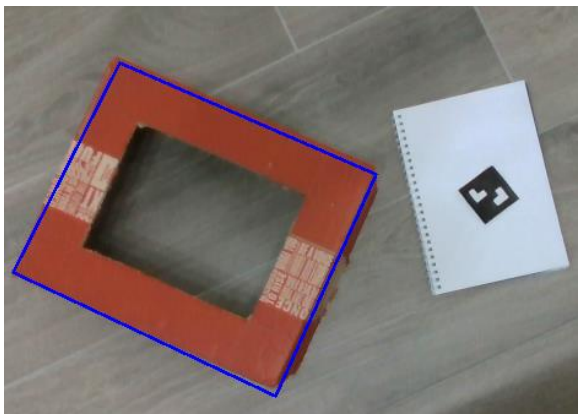

Figura 14 - Bounding box exterior

3.2.1.8 Ajuste de cantos

Visto que pode haver falhas ou ruído na imagem após encontrar os cantos precisamos de verificar que estes se encontram dentro das nossas *canny edges*, visto que este algoritmo é mais preciso.

```python
def adjustBoxPoints(box, canny):
    new_corners = []
    for x, y in box:
        k = 1
        point_adjusted = False
        if not checkInbounds(x, y, canny):
            raise PointOutOfCannyBounds

        if canny[y, x] == 0:
            while not point_adjusted:
                neighbors = canny[y - k:y + k + 1, x - k:x + k + 1]

                indices = np.where(neighbors != 0)
                if len(indices[0]) == 0 or len(indices[1]) == 1:
                    k = k + 1
                    continue

                indices = [(x + idx - k, y + idy - k) for idx, idy in zip(indices[1], indices[0])]

                # Closest index where value != 0
                the_chosen_one = min(indices, key=lambda it: euclidean_distance(x, y, it[0], it[1]))
                new_corners.append(the_chosen_one)
                point_adjusted = True
        else:
            new_corners.append((x, y))
    return np.array(new_corners)
```

Figura 15 - Função para ajuste de cantos

3.2.1.9 Converter para coordenadas reais

Através da API da camara conseguimos passar as coordenas dos cantos de um plano 2D para um plano 3D em metros.

```python
def true_coords(x_2d, y_2d, depth_image, intrinsics):
    # depth value in meters specified in docs
    if not checkInbounds(x_2d, y_2d, depth_image):
        raise ContourOutOfBounds

    depth_scale = intrinsics.depth_scale
    depth = depth_image[int(y_2d), int(x_2d)] * depth_scale

    if depth == 0:
        depth = find_closest_depth(x_2d, y_2d, depth_image) * depth_scale

    x_3d, y_3d, z_3d = convert_depth_to_phys_coord_using_realsense(x_2d, y_2d, depth, intrinsics)
    return x_3d, y_3d
```

Figura 16 - Conversão de coordenadas

Para a conversão precisamos obrigatoriamente de um valor de altura (*depth*) para o pixel. Como em algumas zonas existe algum ruído teremos de verificar se o valor não é nulo, caso seja, vamos procurar o valor na vizinhança mais perto que não seja nulo.

```python
def find_closest_depth(x, y, depth_image):
    # Given depth matrix
    # Find k neighbors check for depth
    # If not found search in k+1 neighbors
    x = int(x)
    y = int(y)
    depth = 0
    k = 1
    while depth == 0:
        # Get neighbors
        neighbors = depth_image[y - k:y + k + 1, x - k:x + k + 1]
        non_zero_neighbors = neighbors[np.nonzero(neighbors)]
        # Only found zeros
        if len(non_zero_neighbors) == 0:
            # Increment k
            k = k + 1
            continue
        depth = min(non_zero_neighbors)
    return depth
```

Figura 17 - Procura de altura

3.2.1.10 Cálculo de área

Tendo os quatros cantos do retângulo conseguimos facilmente saber a largura, altura, e consequentemente a área.

```python
def get_width_height(corners):
    # Corners might be mixed
    tl, tr, br, bl = order_points(np.array(corners))

    width = euclidean_distance(tl[0], tl[1], tr[0], tr[1])
    height = euclidean_distance(tl[0], tl[1], bl[0], bl[1])
    return width, height
```

Figura 18- Cálculo de área através dos cantos de um retângulo

3.3 Cálculo da consistência do betão

Sabemos que o bocal da impressora tem uma largura fixa, logo se a largura da impressão num determinado local for superior ao expectado conseguimos deduzir que o betão foi feito com uma consistência inferior ao desejado.
Adaptando o método de cálculo de volume no ponto 3.2, vamos tirar fotos de um plano superior e comparar a largura da impressão com a largura do bocal da impressora.

3.4 Cálculo do volume de agregados

Iremos calcular o volume dos agregados utilizados para a composição do betão através de sensores resistentes ao ambiente, serão posicionados no topo dos silos de cada agregado. Para podermos calcular o volume de agregado dentro de um silo temos que saber as dimensões do silo assim como o tipo de material que estamos a medir.
Como existem vários tipos de silos, criámos um modelo que suportasse as diferentes características físicas, tais como:

- Cilindro Vertical
- Cilindro Horizontal
- Oval Vertical
- Oval Horizontal
- Retangular

Para que pudéssemos calcular o volume restante de material para cada tipo de silo.

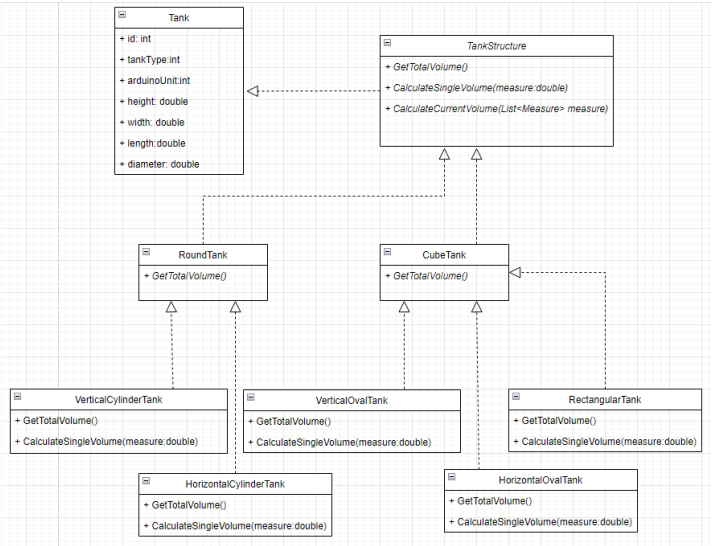

Figura 19 - Diagrama UML Tanques

Podemos observar na Figura 19 - Diagrama UML Tanques o modelo de dados que utilizámos para fazer a relação entre os tipos de tanque de tanque.

3.4.1 Volume de água

No caso da medição do volume de água, vamos necessitar de um sensor de distância ultrassónico que irá medir a distância da superfície da água ao topo do silo (onde está posicionado o sensor), com essa distância conseguimos calcular a altura da água através das medidas do silo e a partir de aí a quantidade de água existente no silo.

3.4.2 Volume de sólidos

No caso da medição do volume de sólidos, vamos necessitar sensores de distância fotoelétricos com um ângulo entre eles de forma a medir a distância em várias zonas do silo. Precisamos de fazer a medição em diferentes zonas pois dependendo do tipo de material que se encontra no silo este poderá não estar sempre nivelado.

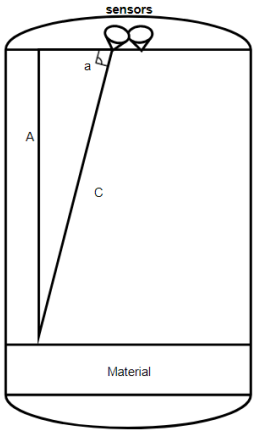

Figura 20 - Diagrama de tanque de agregados

Como os sensores laser fazem um ângulo α com o vetor normal, para obtermos a distância real do topo do silo ao material utilizamos a equação trigonométrica,

$$\sin\theta = \frac{cateto\ oposto}{hipotenusa}$$

No nosso caso,

$$\sin a = \frac{A}{C}$$

Onde a = 90 – α e C = valor obtido pelo sensor.

Obtida a distância A, calculamos o volume do tanque para essa medida. No caso de várias medidas obtidas por sensores diferentes calculamos o volume para cada e fazemos a sua média.

3.4.3 Fluxo de execução

3.4.3.1 Configuração

```
1  //Connection Config
2  #define SSID_NAME ""
3  #define PWD ""
4  #define SERVER_URL "https://192.168.1.66:80"
5  //Arduino Config
6  #define SENSOR_TYPE 2
7  #define SITE_NAME "Luanda Power Build Ultrassonic"
8  #define THRESHOLD 5
```

Figura 21 - Ficheiro config.h

SSID_NAME e PWD dizem respeito à ligação à rede wireless.

SENSOR_TYPE é o id do tipo de sensor ligado ao respetivo arduino, disponível no README do repositório

SITE_NAME é uma pequena descrição que permita identificar o sítio onde o arduino está localizado

THRESHOLD é a diferença em centímetros entre medidas que irá dar dizer se enviamos a medida obtida pelo sensor ou não, este threshold tem objetivo de limitar a quantidade de dados enviados para a API, de forma a não inundar o servidor com dados redundantes.

3.4.3.2 Efetuar ligação à rede wireless

3.4.3.3 Adicionar o arduino à base de dados

Caso seja a primeira vez que o arduino se conecta com o servidor, este vai identificar-se enviando a variável SITE_NAME, ao qual o servidor responde gerando um ID para este arduino.

Após receber o ID, o arduino guarda-o na sua memória interna persistente (EEPROM)

3.4.3.4 Adicionar os sensores à base de dados

Pela mesma lógica da identificação do arduino ao servidor vamos efetuar a mesma operação, no entanto agora para os sensores ligados ao arduino.

3.4.3.5 Obtenção e envio de medidas

Tendo identificado a unidade por completo o arduino pode começar a obter medidas e enviá-las para o servidor.

Após ser enviada uma medida essa e guardada para comparação com a próxima, se esta for maior por um threshold (configurado em config.h) é enviada ao servidor, caso contrário nada acontece até a próxima medida, onde o processo se repetirá.

3.4.3.6 Cálculo de volume

Ao receber as medidas, o servidor vai buscar o objeto do tanque que está associado ao arduino que enviou os dados. A partir do id do tipo de tanque transforma esse objeto num dos modelos especificados na Figura 19 - Diagrama UML Tanques, para poder calcular o volume atual do tanque.

```
[HttpPost(Name = "CreateVolume")]
0 references
public async Task<ActionResult<long>> Create(VolumeDto volumeDto)
{
    var tank = await _context.Tanks.FirstOrDefaultAsync(t => t.ArduinoUnitId == volumeDto.ArduinoUnitId);

    var volume = new Volume
    {
        Measures = volumeDto.MeasuresDtos.Select(m => new Measure
        {
            Value = m.Value
        }).ToList(),
        ArduinoUnitId = volumeDto.ArduinoUnitId
    };
    if (tank == null)
    {
        return BadRequest("No tank associated with this arduino");
    }
    foreach (MeasureDto m in volumeDto.MeasuresDtos)
    {
        if (m.Value > tank.Height)
        {
            return BadRequest("Obtained measure overflows tank");
        }
    }

    var tankTypeName = GetTankTypeNameById(tank.TankTypeId);
    volume.VolumeValue = TankFactory.ParseTank(tank, tankTypeName)!.CalculateCurrentVolume(volume.Measures);
    Console.Write("Calculated " + volume.VolumeValue);

    _context.Volumes.Add(volume);
    await _context.SaveChangesAsync();
    return Ok(volume.Id + " | " + volume.VolumeValue);
}
```

Figura 22 - Volume Controller

Na Figura 23 - Cálculo de volume para VerticalOvalTank podemos observar um exemplo da implementação da função CalculateSingleVolume no objeto VerticalOvalTank.

```
2 references
protected override double CalculateSingleVolume(double measure)
{
    var a = (Height - Width);
    var r = Width / 2.0;
    var filled = Height - measure;
    var singleMeasure = new Measure
    {
        //only used to calculate if the filled portion is less than half of cylinder
        Value = measure - a
    };
    var measures = new List<Measure>();
    measures.Add(singleMeasure);


    var filledHorizontalRect = new RectangularTank
    {
        Length = Length,
        Height = filled - r,
        Width = Width
    };

    var horizontalCylinder = new HorizontalCylinderTank()
    {
        Length = Length,
        Diameter = Width,
    };

    //exactly half of cylinder portion
    if (filled > r && filled < r + a)
    {
        return 0.5 * horizontalCylinder.GetTotalVolume() + filledHorizontalRect.GetTotalVolume();
    }
    if (filled > r + a && filled < Height)
    {
        var m = r - measure;
        var angle = 2 * Math.Acos(m / r); //radians;
        var segmentVolume = 0.5 * Math.Pow(r, 2) * (angle - Math.Sin(angle)) * Length / 1000.0;
        return Math.Round(GetTotalVolume() - segmentVolume, 4);
    }
    //only bottom half cylinder filled
    return horizontalCylinder.CalculateCurrentVolume(measures);
}
```

Figura 23 - Cálculo de volume para VerticalOvalTank

3.5 Microprocessor e sensors

3.5.1 Arduino NODEMCU (ESP32)

Decidimos utilizar o – NodeMCU (Figura 24) pois é uma Plataforma IoT de baixo custo e Open-Source, esta utiliza o chip ESP32.Este módulo será o cérebro dos nossos sensores, e será o que efetuará o envio dos dados através de WiFi para a nossa REST API.

Figura 24 – NodeMCU

3.5.2 Sensor ultrassónico

Para medir a altura do braço e o volume de água nos tanques, optámos por escolher um sensor de distância ultrassónico (Figura 25 - Jsn-sr04t) pois este tipo de sensor é imune à cor do alvo, refletividade e transparência, adicionalmente não são afetados por condições de luz da área e funcionam bem em ambientes sujos e húmidos.

Figura 25 - Jsn-sr04t

3.5.3 Sensor fotoelétrico

Para medir o volume de agregados nos tanques, optámos por escolher um sensor de distância laser (Figura 26 - GY-VL53L0XV2 L53L0X TOF). Os sensores de distância laser têm uma alta velocidade de resposta, intervalos de deteção mais longos e níveis de precisão mais elevados que os sensores ultrassónicos.

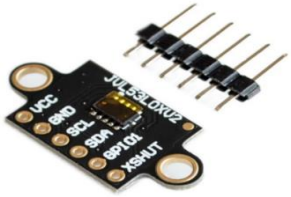

Figura 26 - GY-VL53L0XV2 L53L0X TOF

3.6 Arquitetura

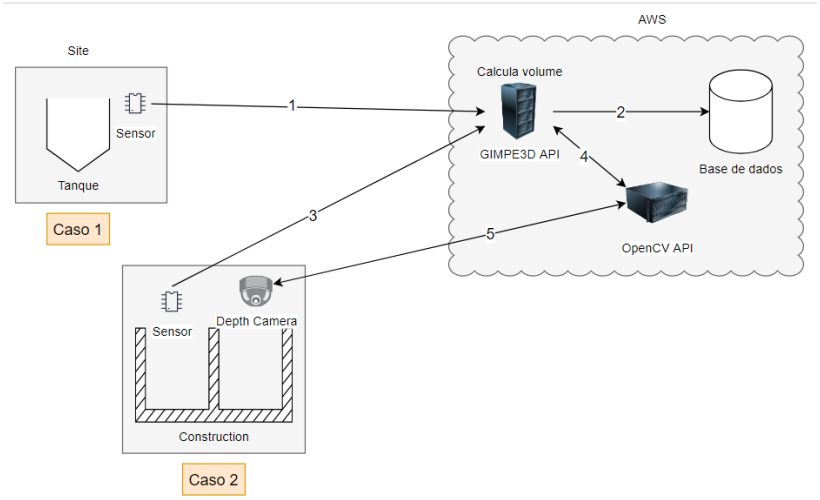

Figura 27 - Arquitetura

1 – Envio de medidas para o servidor principal (distância do topo do tanque ao material)
2 – Envio do volume para a base de dados
3 – Envio da altura da parede da construção
4 – Pedido de área à API de OpenCV / Envio de área calculada para servidor principal
5 – Pedido de fotos da construção

Após a recolha de dados pelos sensores ligados aos arduinos, vamos enviá-los para uma REST API hospedada na AWS escrita em .NET Core onde serão efetuados os cálculos discutidos no ponto 3.3 , 3.4 e 3.4. Estes dados processados serão usados para uma aplicação de gestão e suporte à decisão.

Para a solução dos problemas vamos utilizamos as seguintes tecnologias: SQL Server, .NET Core, AWS, OpenCV.

SQL Server é o sistema de gestão de base de dados que utilizaremos para gerir a nossa base de dados.

.NET Core é uma *framework* de uso geral para desenvolvimento, neste caso será utilizado para escrever a nossa REST API que após a receção dos dados calcula os volumes.

AWS é a plataforma de serviços *cloud* onde iremos hospedar a nossa API.

OpenCV é a biblioteca multiplataforma que iremos utilizar para efetuar o processamento de imagem no cálculo do volume no ponto 3.2 e no cálculo da consistência do betão no ponto 3.3. e será utilizada numa aplicação Python onde será também servida uma REST API.

Ao utilizar REST em vez de SOAP facilitamos a integração com o arduino, tal como a integração com outras aplicações existentes pois não existe a necessidade de alterar a infraestrutura da aplicação.

4 Benchmarking

4.1 Sensores

Após realizar uma pesquisa sobre tecnologias atualmente existentes, foi possível encontrar o sensor 3DLevelScanner S (Figura 29).

Este sensor mede vários pontos do material utilizando métodos acústicos avançados que permitem saber o volume de material e criar um modelo da superfície detetada.

Na Figura 28 podemos observar o modelo criado pelo software que acompanha este sensor. No entanto, esta tecnologia tem um alto custo de aquisição e não permite a integração com a nossa solução.

Face à concorrência identificada a nossa solução tem a possibilidade de fácil integração em sistemas visto que pode ser feita através de chamadas a uma REST API, no entanto a nossa implementação não inclui um software/website de visualização, sendo essa parte essa integração responsabilidade da empresa.

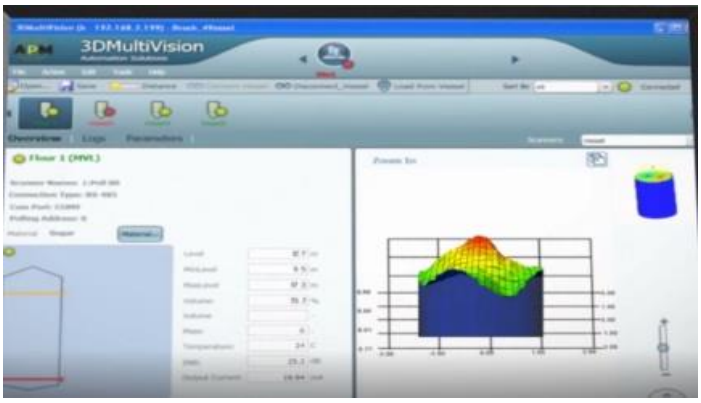

Figura 28 - 3DMultiVision

Figura 29 - 3DLevelScanner S

4.2 Impressão 3D

No tema de construção de casas recorrendo a impressoras 3D encontrámos a ICON, uma empresa americana. Ao aceder ao seu website podemos encontrar detalhes de vários projetos como por exemplo criar uma comunidade de 100 casas impressas como mostra a Figura 30.

Figura 30 – Comunidade conceptual criada por ICON

4.3 Cálculo de área de impressão

Na nossa pesquisa não encontrámos outras soluções que tivessem o mesmo objetivo de calcular a área ou o volume de impressões através de processamento de imagens capturadas da construção, sendo assim no nosso ver uma implementação inovadora.

5 Método e planeamento

Para conseguir desenvolver entregáveis a cada etapa de entrega neste projeto, utilizamos a metodologia Agile.

Implementamos esta metodologia com sprints de 2 a 3 semanas. No final de cada sprint o trabalho realizado foi apresentado tanto ao orientador quanto à empresa Yucoders.

Na seguinte tabela mostra o calendário de plano de trabalho e as tarefas propostas que foram entregues dentro das *deadlines* definidas por nós.

Tabela 2 - Plano de trabalho

Tarefa	Data Inicial	Data Final	Concluída
Setup photoeletric sensor	27/12/2021	20/01/2022	✓
Setup ultrassonic sensor	27/12/2021	20/01/2022	✓
Prepare the second report delivery	20/01/2022	28/01/2022	✓
Create REST API Server	28/01/2022	20/02/2022	✓
Send arduino data to server through Rest API	20/02/2022	05/03/2022	✓
Setup database	05/03/2022	10/04/2022	✓
Prepare the third delivery of the report	10/04/2022	24/04/2022	✓
Calculate the volume of aggregates	24/04/2022	10/06/2022	✓
Calculate the volume of concrete	24/04/2022	10/06/2022	✓
Calculate the consistency of concrete	24/04/2022	10/06/2022	X
Deploy .NET Core Web App (REST API Service) to AWS	10/06/2022	20/06/2022	X
Prepare the last delivery of the report	20/06/2022	29/06/2022	✓

6 Resultados

6.1 Cálculo da área de impressão

Como podemos observar no Anexo 3 como é de esperar em cada execução é obtida uma área diferente, pelo que é importante retirarmos vários frames e calcular várias áreas selecionando como área final a mediana das nossas amostras.

Apesar de na implementação prévia recorrermos ao uso de *Aruco markers* (Figura 5) para obter uma escala de pixéis para metro, no decorrer do desenvolvimento reparamos que a API da camara fornece uma função para converter pontos da imagem em coordenadas reais na qual já se encontra incluída a escala necessária, não sendo assim preciso nenhum marcador.

Para construções de grande proporção poderá ser necessária a inclusão do marcador para a localização dos pontos.

Como forma de comparação na figura abaixo estão especificadas as medidas reais das caixas.

Box	Width	Height	Area
Closed Box	27 cm	17 cm	459 cm
Open Box Outer	28 cm	23,5 cm	658 cm
Open Box Hole	18 cm	13,5 cm	243 cm
Open Box	-	-	415 cm

Figura 31 - Dimensões reais da caixa

6.2 Cálculo do volume de materiais em tanques

Podemos observar pelo ficheiro em anexo “Testing Values.xlsx” algumas medidas de teste e os respetivos volumes de agregados comparando-os com os volumes obtidos através dos nossos cálculos.

Id	VolumeValue (L)	ArduinoUnitId	CreatedAt	Real Volume Value (L)	Erro (%)
112	985,67	5	1,65054E+12	1005,31	1,95
113	1024,94	5	1,65054E+12	1060,29	3,33
119	1068,14	5	1,65054E+12	1099,56	2,86
121	1091,7	5	1,65054E+12	1138,83	4,14
123	1146,68	5	1,65054E+12	1178,1	2,67
126	1185,95	5	1,65054E+12	1217,37	2,58
128	1221,29	5	1,65054E+12	1256,64	2,81
131	1260,56	5	1,65054E+12	1295,91	2,73
135	1311,61	5	1,65054E+12	1335,18	1,77
137	1350,88	5	1,65054E+12	1374,45	1,71
141	1405,86	5	1,65054E+12	1413,72	0,56
143	1287,36	4	1,65054E+12	1280	0,57
144	1373,38	4	1,65054E+12	1350	1,73
148	1421,49	4	1,65054E+12	1400	1,54
150	1467,22	4	1,65054E+12	1450	1,19
152	1538,45	4	1,65054E+12	1500	2,56
156	1566,5	4	1,65054E+12	1550	1,06
158	1653,54	4	1,65054E+12	1600	3,35
164	1682,1	4	1,65054E+12	1650	1,95
166	1703,01	4	1,65054E+12	1700	0,18
168	1801,61	4	1,65054E+12	1750	2,95

Figura 32 - Resultados de teste

Bibliografia

[DEISI21] DEISI, Regulamento de Trabalho Final de Curso, Set. 2021.

[TaWe20] Tanenbaum,A. e Wetherall,D., *Computer Networks*, 6ª Edição, Prentice Hall, 2020.

[ULHT21] Universidade Lusófona de Humanidades e Tecnologia, www.ulusofona.pt, acedido em Out. 2021.

[VIKUNJA] Vikunja, www.vikunja.io, acedido em Nov. 2021.

[TRELLO] Trello, www.trello.com, acedido em Set. 2021

[OPENCV] OpenCV, www.opencv.org, acedido em Set. 2021

[ICON] ICON, www.iconbuild.com, acedido em Nov. 2021

[CALCULATOR SOUP] CALCULATOR SOUP https://www.calculatorsoup.com, acedido em Março 2022

[INCH CALCULATOR] INCH CALCULATOR, https://www.inchcalculator.com, acedido em Março 2022

[INTEL REAL SENSE PYTHON API] INTEL REAL SENSE PYTHON API, https://github.com/IntelRealSense/librealsense/tree/master/wrappers/python, acedido em Junho 2022

[REAL SENSE CALIBRATION] REAL SENSE CALIBRATION, https://learnopencv.com/camera-calibration-using-opencv/, acedido em Junho 2022

[CAMERA CALIBRATION] CAMERA CALIBRATION, https://docs.opencv.org/4.x/dc/dbb/tutorial_py_calibration.html, acedido em Junho 2022

[CONVERTING 2D COORDINATES] CONVERTING 2D COORDINATES, https://medium.com/@yasuhirachiba/converting-2d-image-coordinates-to-3d-coordinates-using-ros-intel-realsense-d435-kinect-88621e8e733a, acedido em Junho 2022

Glossário

LEI	Licenciatura em Engenharia Informática
LIG	Licenciatura em Informática de Gestão
TFC	Trabalho Final de Curso
IoT	Internet of Things
SQL	Structured Query Language
API	Application Program Interface
AWS	Amazon Web Services
REST	Representational State Transfer
SOAP	Simple Object Access Protocol
JWT	Java Web Token
ML	Machine Learning

Anexo 1 – Modelo de Dados

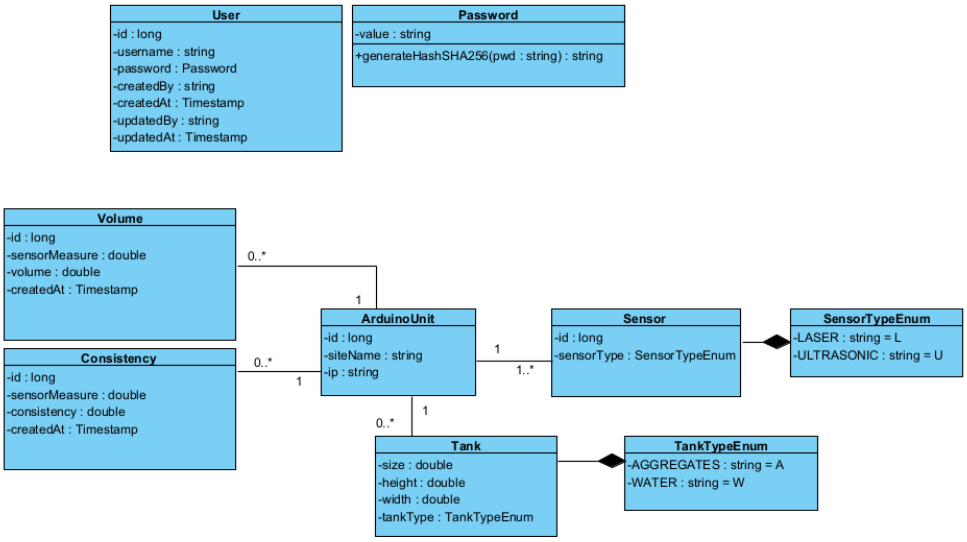

Figura 33 - Modelo de dados

Anexo 2 – Plano de Trabalho Gantt

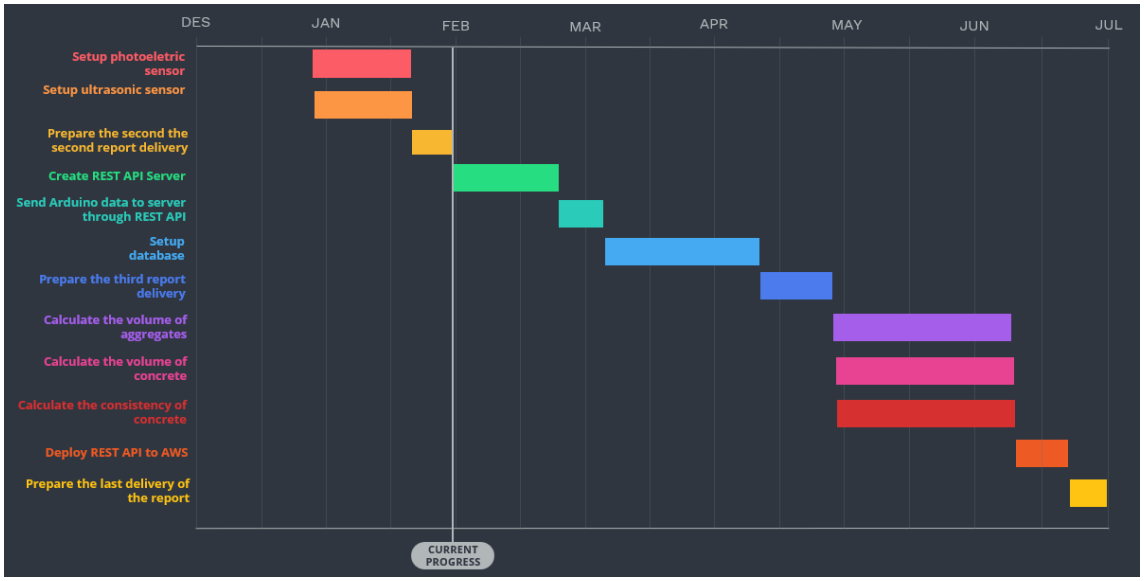

Figura 34 - Plano de trabalho Gantt

Anexo 3 – Resultados do cálculo da área

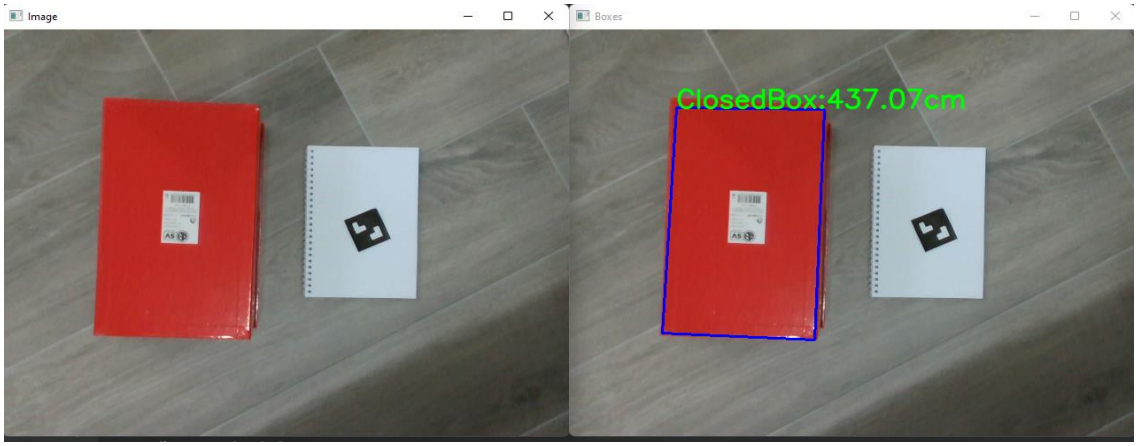

Figura 35 - Caixa Fechada

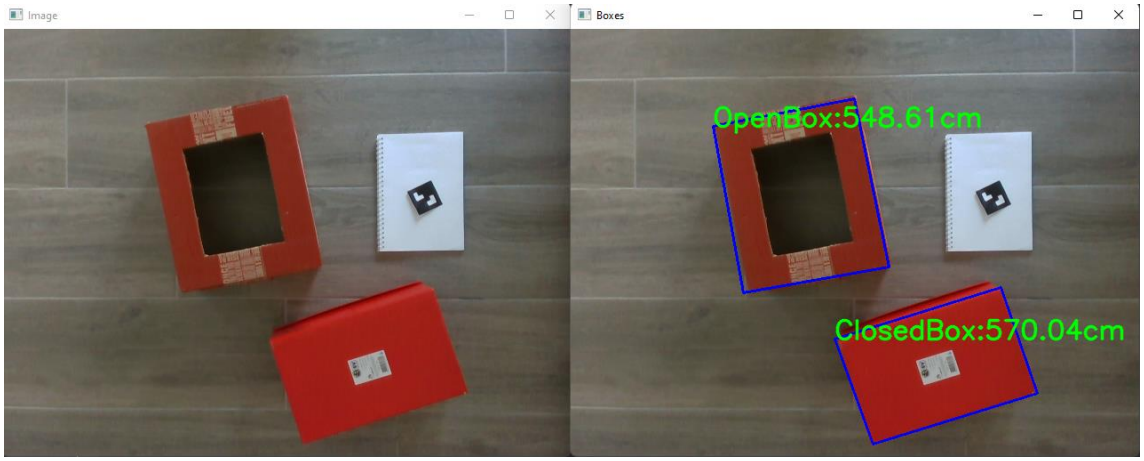

Figura 36 - Caixa aberta

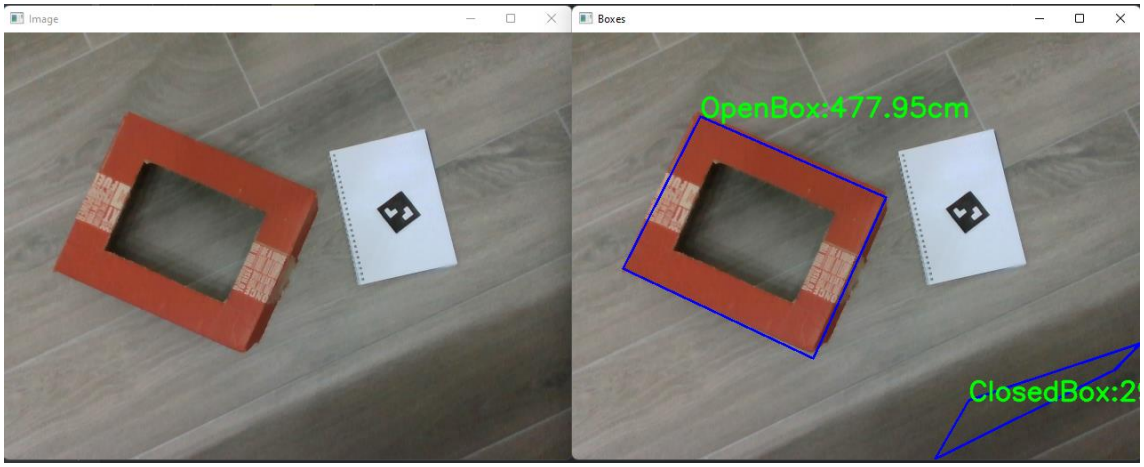

Figura 37- Caixa Aberta

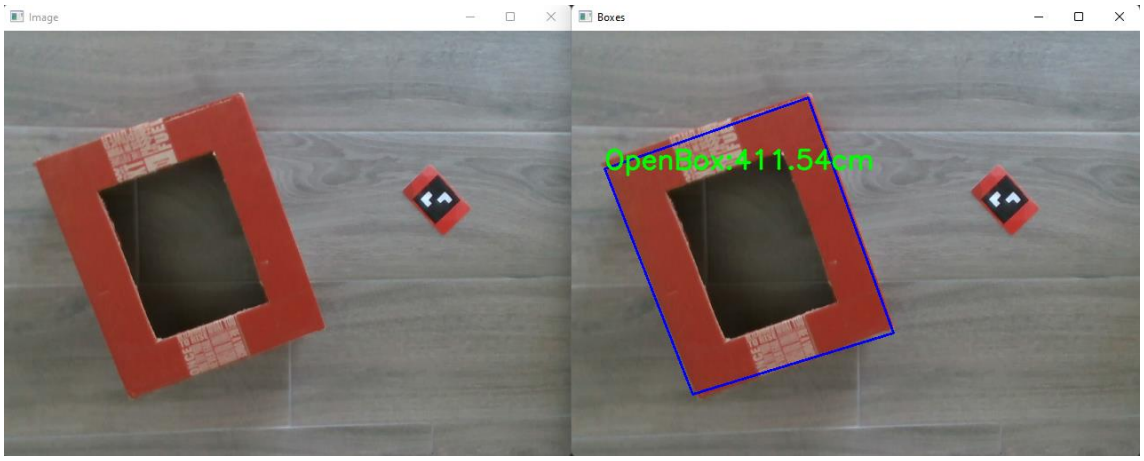

Figura 38- Caixa Aberta

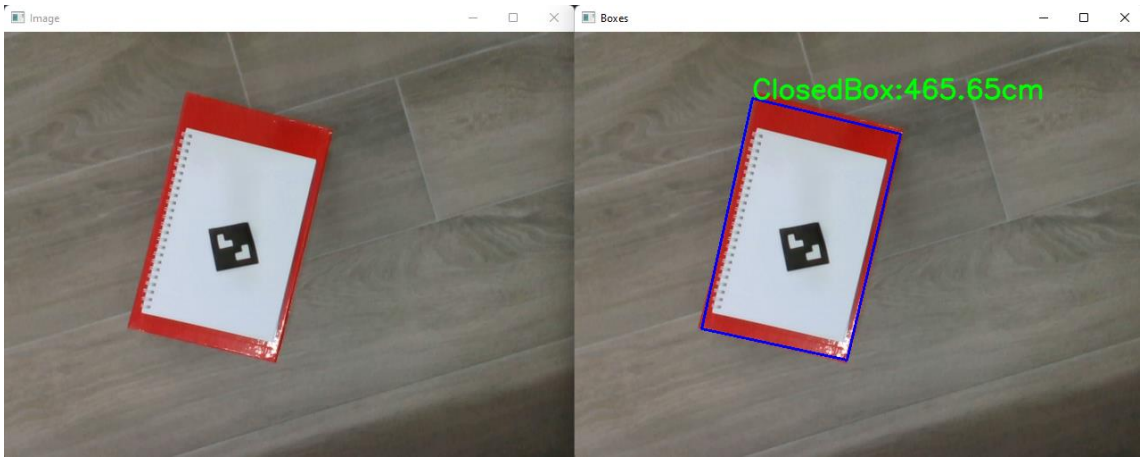

Figura 39 - Caixa fechada

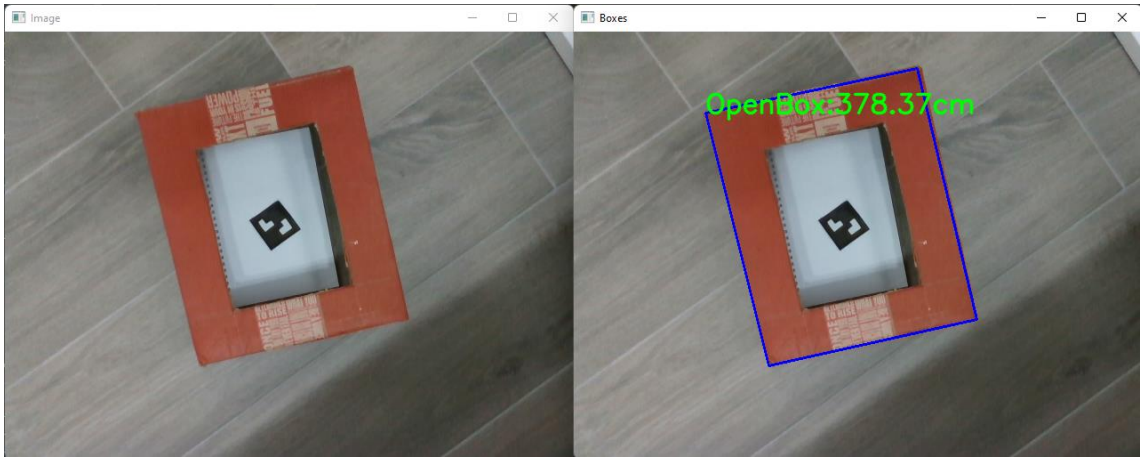

Figura 40 - Caixa Aberta