

Universidade de Aveiro

Departamento de Eletrónica, Telecomunicações e Informática



Projeto Engenharia de Computadores e Telemática

Módulo da Camara Reolink

Autor: Rafael Amorim, 98197
01/05/2023

Índice

Introdução	1
Arquitetura do módulo	1
Modelo da Camera.....	2
Implementação	3
Transmissão com OBS:	3
MQTT:.....	3
Reolink API:	4
Anexo	8

Figuras

Figura 1: Diagrama Camara	1
Figura 2: Modelo Camara	2
Figura 3: Meio de comunicação entre módulos.....	3
Figura 4: Descoberta do Gateway	8
Figura 5: Configurar gateway do Router	9
Figura 6: Exemplo de Configuração.....	9

Introdução

O presente relatório tem por objetivo descrever a solução implementada na aplicação Gym at Home para a captura e transmissão em tempo real dos exercícios realizados por idosos. Para atender aos requisitos estabelecidos, optou-se pelo uso da camara Reolink E1 Zoom, que apresenta todas as características necessárias, tais como capacidade PTZ, áudio bidirecional, conectividade Wi-Fi de 2,4/5 GHz e resolução Super HD 5MP, entre outras, que serão detalhadas posteriormente na seção específica da camara utilizada.

O módulo da camara pode ser dividido em duas partes distintas. A primeira concentra-se no método de transmissão, enquanto a segunda aborda o aproveitamento da API Reolink para o acesso às funcionalidades disponibilizadas pela camara.

Arquitetura do módulo

O diagrama apresentado na Figura 1 ilustra de forma breve o processo de comunicação do sistema. Conforme se pode observar, existe um servidor central que se encontra conectado a dois routers distintos. Um dos routers estabelece a ligação com a residência do fisioterapeuta, local onde serão inseridos os planos de treino destinados ao idoso em causa. O outro router, por sua vez, estabelece a conexão com as casas dos idosos, permitindo assim a captação de imagem através de câmaras e a reprodução da aplicação, assim como a transmissão da mesma para o computador do fisioterapeuta.

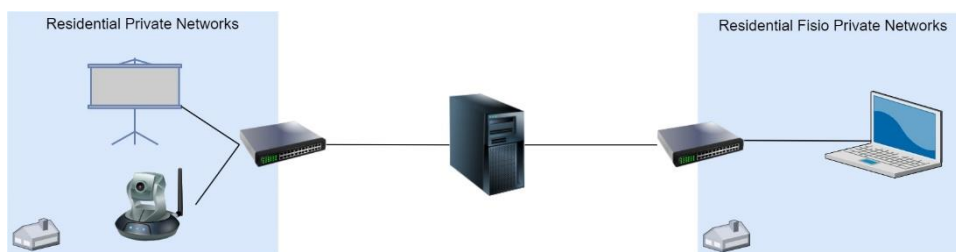


Figura 1: Diagrama Camara

A camara é subdividida em três módulos:

- Configuração da camara, este sub módulo permite configurar a camara para que possa ser conectada à rede Wi-Fi;
- Streaming, este sub módulo permite que o fluxo de vídeo da camara seja transmitido em tempo real para a plataforma de exercícios, permitindo que os utilizadores possam visualizar sua performance enquanto realizam os exercícios.
- Controle PTZ, este sub módulo permite controlar oralmente a posição, o zoom e a inclinação da camara para garantir que os idosos sejam visualizados corretamente durante a realização dos exercícios.

Modelo da Camera



Figura 2: Modelo Camara

O modelo E1 Zoom, produzido pela marca Reolink, é uma camara de segurança de última geração que possui recursos avançados. Este dispositivo apresenta uma lente de zoom ótico de 3x, permitindo a ampliação e visualização de objetos distantes com alta clareza. Ademais, a camara possui resolução de imagem de 5 megapixéis e grava vídeos em Full HD (1080p). Além disso, está equipada com um sensor de movimento PIR e um LED infravermelho para capturar imagens nítidas em ambientes com baixa luminosidade.

O dispositivo E1 Zoom ainda conta com um recurso de áudio bidirecional, permitindo que o utilizador ouça e fale com pessoas presentes no ambiente monitorado. Esse recurso é especialmente útil para monitorar crianças ou animais de estimação. A camara pode ser facilmente configurada por meio do aplicativo móvel Reolink, além de permitir acesso remoto em tempo real por meio de dispositivos móveis ou computadores. Além disso, a camara é compatível com o assistente virtual Amazon Alexa, permitindo controle por voz.

De maneira geral, a camara E1 Zoom da Reolink apresenta-se como uma excelente opção para aqueles que procuram uma solução de segurança avançada, com recursos práticos e de fácil utilização.

Implementação

Transmissão com OBS

Para este projeto, foi utilizado o Youtube como plataforma de streaming. Iniciou-se uma transmissão no canto superior direito da página e, em seguida, sincronizou-se com a conta do OBS. Para fazer isso, foram acessadas as configurações e selecionou-se o Serviço como "Youtube - RTMPS" e o servidor como "Primary Youtube ingest server". Depois, adicionou-se uma "Fonte Multimídia" nas Fontes, atribuiu-se um nome e desmarcou-se a opção de "Ficheiro Local". Na entrada, foi inserido o seguinte link:

```
rtsp://admin:password@<Endereço_IP>:<Port>/h264Preview_01_main
```

Na opção de formato de entrada, foi inserida a chave da transmissão que é fornecida pelo Youtube de forma omitida. Por fim, basta iniciar a transmissão."

MQTT

No contexto do nosso projeto Gym-Home, a comunicação entre o assistente de voz, a interface de usuário (UI) e o assistente da câmara é realizada através do protocolo MQTT (Message Queuing Telemetry Transport). Esse protocolo de comunicação é amplamente utilizado em sistemas IoT (Internet of Things) e é baseado em mensagens, permitindo a transmissão de informações de forma assíncrona entre dispositivos.

Desta forma, quando o assistente de voz emite uma instrução para controlar a câmara, ela é enviada como uma mensagem do tipo "publisher" através do MQTT. Essa mensagem é então recebida pelo "subscriber" myTest.py, que é responsável por executar a ação correspondente na câmara.

Por meio dessa arquitetura, conseguimos estabelecer uma comunicação eficiente e segura entre os diferentes componentes do sistema, garantindo uma experiência de uso satisfatória para o usuário final.

Na Figura 3 temos apresentado o diagrama conforme estabelecidas as comunicações entre módulos através do MQTT.

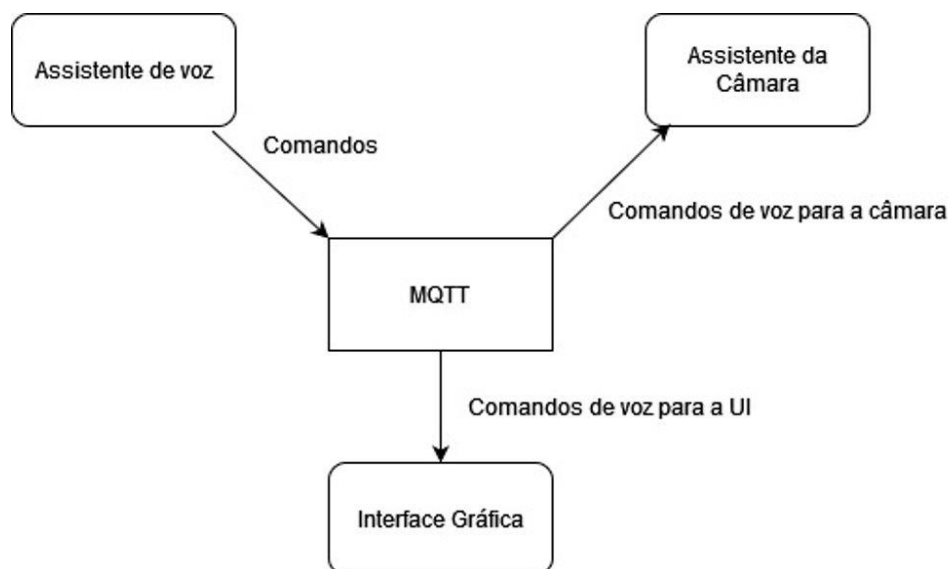


Figura 3: Meio de comunicação entre módulos

Reolink API

A Reolink API é um conjunto de ferramentas de programação que permite o controle remoto da câmara de segurança Reolink. No nosso projeto Gym-Home, optamos por utilizar a linguagem de programação Python e um repositório existente chamado [reolinkapi](#), que oferece diversas operações para controlar a câmara. No entanto, para o nosso propósito, focamos exclusivamente nas funcionalidades de Pan-Tilt-Zoom (PTZ).

Antes de tudo, foi necessário criar um arquivo `secrets.cfg`, que contém informações como o endereço IP, a porta, o nome de utilizador e a palavra-passe da câmara, a fim de estabelecer uma conexão com a câmara instalada na residência do idoso e, posteriormente, controlá-la.

O código abaixo, encontrado em `Gym-Home\Cam\reolinkapiV5\reolinkapi\mixins\ptz.py`, demonstra as funções ao qual tiramos partido para controlar o movimento da câmara:

```

def move_right(self, speed: float = 25) -> Dict:
    """
    Move the camera to the right
    The camera moves self.stop_ptz() is called.
    :return: response json
    """
    return self._send_operation('Right', speed=speed)

def move_left(self, speed: float = 25) -> Dict:
    """
    Move the camera to the left
    The camera moves self.stop_ptz() is called.
    :return: response json
    """
    return self._send_operation('Left', speed=speed)

def move_up(self, speed: float = 25) -> Dict:
    """
    Move the camera up.
    The camera moves self.stop_ptz() is called.
    :return: response json
    """
    return self._send_operation('Up', speed=speed)

def move_down(self, speed: float = 25) -> Dict:
    """
    Move the camera down.
    The camera moves self.stop_ptz() is called.
    :return: response json
    """
    return self._send_operation('Down', speed=speed)

def stop_ptz(self) -> Dict:
    """
    Stops the cameras current action.
    :return: response json
    """
    return self._send_noparm_operation('Stop')

def auto_movement(self, speed: float = 25) -> Dict:
    """
    Move the camera in a clockwise rotation.
    The camera moves self.stop_ptz() is called.
    :return: response json
    """
    return self._send_operation('Auto', speed=speed)

```

Para executar o script, é necessário navegar até o diretório correto, que neste caso é:

Gym-Home\Cam\reolinkapiV5\tests e, em seguida, executar o arquivo myTest.py. Após a execução, o script ficará em espera para receber instruções via Mosquitto.

O seguinte código apresenta como se resolveu questões de conectividade e de configurações com a camara.

```
import os
import unittest
import paho.mqtt.client as paho
import time
import json
from configparser import RawConfigParser
from reolinkapi import Camera
from reolinkapi.mixins.ptz import PtzAPIMixin

def read_config(props_path: str) -> dict:
    config = RawConfigParser()
    assert os.path.exists(props_path), f"Path does not exist:
{props_path}"
    config.read(props_path)
    return config

class TestCamera(unittest.TestCase, PtzAPIMixin):

    def __init__(self):
        self.setUpClass()
        self.setUp()

    def cam(self):
        return self.cam

    def setUpClass(cls) -> None:
        cls.config = read_config('../secrets.cfg')

    def setUp(self) -> None:
        self.cam = Camera(self.config.get('camera', 'ip'),
self.config.get('camera', 'username'), self.config.get('camera',
'password'))

    def test_camera(self):
        """Test that camera connects and gets a token"""
        self.assertTrue(self.cam.ip == self.config.get('camera',
'ip'))
        self.assertTrue(self.cam.token != '')
        print("fez o test Camera")
```


Logo de seguida temos no mesmo ficheiro o código responsável pelas ações da camara recebidas como instruções através do mosquito.

```
def on_message(mosq, obj, msg):
    camar = TestCamera()

    print ("%20s %d %s" % (msg.topic, msg.qos, msg.payload))
    mosq.publish('pong', 'ack', 0)
    message = msg.payload
    message = json.loads(message)
    print (message ['comando'])

    if message['comando'] == 'esquerda':
        camar.cam.move_left()
        time.sleep(1)
        camar.cam.stop_ptz()
    elif message['comando'] == 'direita':
        camar.cam.move_right()
        time.sleep(1)
        camar.cam.stop_ptz()
    elif message['comando'] == 'cima':
        camar.cam.move_up()
        time.sleep(1)
        camar.cam.stop_ptz()
    elif message['comando'] == 'baixo':
        camar.cam.move_down()
        time.sleep(1)
        camar.cam.stop_ptz()
    else:
        print("Comando inválido")

if __name__ == '__main__':

    client = paho.Client()
    client.on_message = on_message
    client.connect("127.0.0.1", 1883, 60)
    client.subscribe("comandos/voz/camara", 0)

    while client.loop() == 0:
        pass

    unittest.main()
```

Para testar o script, pode-se utilizar o arquivo publisher.py, localizado no mesmo diretório. É possível variar as instruções enviadas, como "esquerda" ou "cima", por exemplo.

```
import paho.mqtt.publish as publish

msgs = [{'topic': "comandos/voz/camara", 'payload': "jump"}]
host = "localhost"

if __name__ == '__main__':
    string = {'comando': 'direita'}
    import json
    publish.single(topic="comandos/voz/camara", payload= json.dumps(string), hostname=host)
```

Configurar o router para redes diferentes:

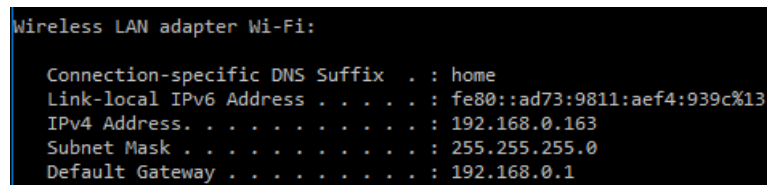
Para configurar a câmara de segurança Reolink para funcionar em redes diferentes, é necessário conectá-la primeiro à rede onde ela será permanentemente instalada, por exemplo, na casa da Dona Maria. Em seguida, é necessário abrir o terminal e executar o comando "ipconfig" para identificar o endereço IP do router, conforme apresentado na Figura 4.

Com base no endereço "Default Gateway" obtido, deve-se acessar a página de configuração do router por meio do navegador. Em seguida, para configurar um endereço externo usando os dados da câmara, como o endereço e a porta interna, bem como o protocolo, devem ser seguidos os passos apresentados nas Figura 5 e na Figura 6.

Posteriormente, deve-se modificar o arquivo secrets.cfg para incluir o endereço IP público. Por fim, deve ser testada a conexão externamente por meio do seguinte link: <http://admin:GymHome.2@84.90.27.10:80>.

É importante ressaltar que a configuração do router pode variar de acordo com o modelo e fabricante, portanto, é recomendável consultar o manual do equipamento para obter informações específicas.

Anexo



```
Wireless LAN adapter Wi-Fi:

Connection-specific DNS Suffix . : home
Link-local IPv6 Address . . . . . : fe80::ad73:9811:aef4:939c%13
IPv4 Address. . . . . : 192.168.0.163
Subnet Mask . . . . . : 255.255.255.0
Default Gateway . . . . . : 192.168.0.1
```

Figura 4: Descoberta do Gateway

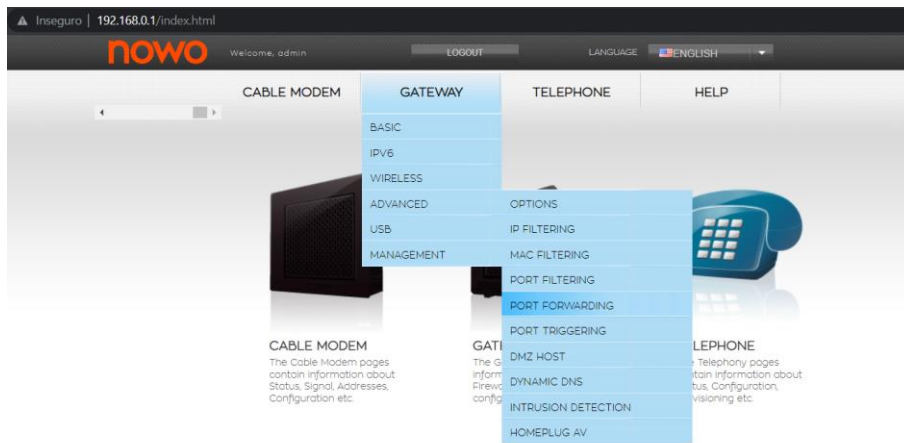


Figura 5: Configurar gateway do Router

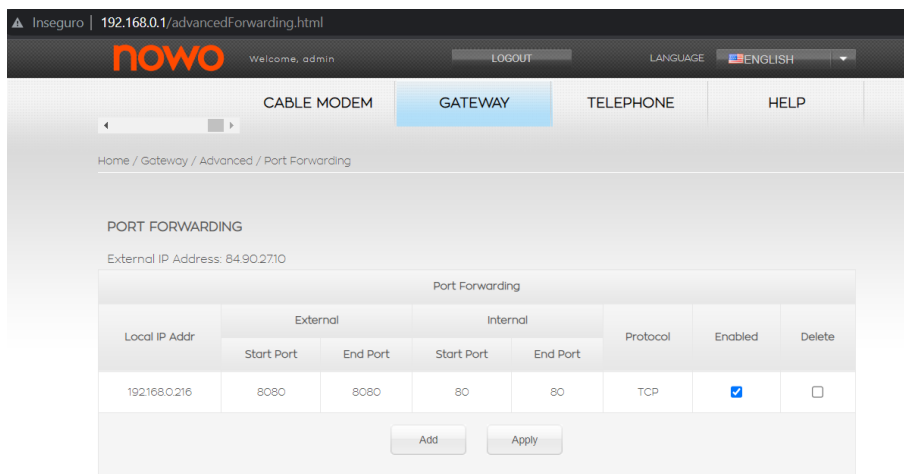


Figura 6: Exemplo de Configuração