# 《物联网与嵌入式技术》

# 大作业课题报告

## 客户端图片抓取与阿里云配置


学 生 姓 名：魏佳琪

学科、 专业：生物医学工程

学　　号：22350014


完 成 日 期：2024.5.5


大连理工大学

Dalian University of Technology

# 目　　录

# 1 负责任务介绍

## 1.1 软件部分小组分工安排

负责进行软件部分的小组分工安排，以及软件部分的整体进度把控。创建和维护 Github 仓库便于小组协作。

## 1.2 客户端图片抓取功能实现

辅助客户端实现按用户需求抓取图片的功能，以便后续将图片进行处理和发送到服务端。

## 1.3 阿里云服务端搭载

将服务端搭载到阿里云平台，制作相应的 docker，进行端口的调整和协议的改变。

# 2 客户端图片抓取

## 2.1 客户端功能介绍

（1） 图片抓取

① 每隔固定时间进行整个屏幕抓取。

② 按照用户意图进行主动的屏幕部分截取。以便于实现近似于“点读”的功能。检测到热键后，创建一个全屏透明窗口，允许用户通过拖动鼠标来选择屏幕上的一个区域进行截图。

（2） 图片预处理与发送

① 使用 OpenCV 进行图片完整性检查和压缩。

② 对图片进行分包后使用 TCP 协议发送到服务器。

此报告中实现的即图片抓取功能。

## 2.2 固定时间图片抓取实现

### 2.2.1 使用的库

（1） OpenCV

OpenCV（Open Source Computer Vision Library）是一个开源计算机视觉库，用于处理图像和视频数据。它提供了丰富的图像处理和计算机视觉算法，包括但不限于图像捕获、图像处理、特征检测、目标跟踪等功能。

在本项目代码中，cv2 主要用于图像的编码、解码和颜色空间转换，例如将屏幕截图从 BGR 颜色空间转换为 RGB 颜色空间。在这部分功能的实现中，主要使用它将图片存储为 cv2 格式，便于后续图片处理。

（2） pyautogui

PyAutoGUI 是一个用于自动化鼠标和键盘操作的 Python 库，可以模拟人类的键盘输入、鼠标移动和点击等操作。它能够实现屏幕截图、窗口控制、鼠标移动、键盘输入等功能。

在本项目代码中，pyautogui 用于捕获屏幕截图和获取鼠标位置，以便确定选定区域的坐标。

（3） time

time 是 Python 的标准库之一，提供了处理时间和日期的功能。它可以用于获取当前时间、计算时间间隔、设置定时器等操作。

在本项目代码中，time 主要用于计时和设置定时器，以实现定时截图和重试功能。

### 2.2.2 代码实现

（1） 设置参数和变量

设置了超时时间、最大重试次数。

（2） 定义截图函数

capture_screen(region=None)：此函数使用 pyautogui.screenshot 进行全屏的截图，然后使用 cv2.cvtColor 将图像从 BGR 转换成 RGB 格式。

（3） 实时屏幕捕获

主循环中，程序不断检测是否到了自动捕获全屏的时间间隔，如果是，则调用 capture_screen()函数进行全屏捕获。

（4） 图像编码和文件保存

捕获到的图像首先被编码成 JPEG 格式，然后保存到本地文件系统中，便于 debug。同时，捕获到的 cv2 格式图像将被用于进行下一步处理。

## 2.3 用户主动图片抓取实现

### 2.3.1 使用的库

在 2.2 使用的库的基础上，新增了下列库以实现主动的部分屏幕抓取功能。

（1） keyboard

keyboard 是一个 Python 库，用于全局监听和控制键盘。它可以用于记录按键、监听按键事件或模拟键盘输入，适用于 Windows、Linux 和 Mac OS X，主要特点和功能如下：

① 按键监听：可以监听特定的按键或组合键的按下和释放事件。例如，可以设定程序在用户按下特定的快捷键时执行特定的动作。

② 发送按键：能够模拟键盘按键输入，可以发送按键事件到系统，就如同用户真实地敲打键盘一样。

③ 记录和回放：支持记录用户的键盘活动，并能够回放这些活动。

④ 阻塞和非阻塞模式：可以在阻塞模式下监听按键，此时程序会在监听到指定按键事件前停止运行。非阻塞模式允许程序继续执行其他任务，同时在后台监听键盘事件。

⑤ 热键和快捷键：可以轻松设置热键或快捷键，当用户按下相应的键时触发预设的函数。

在本项目代码中，keyboard 库用于检测用户是否按下了"F12"键，用于触发屏幕捕获功能。

（2） Tkinter 的 Tk 和 Canvas

Tkinter 是 Python 的标准 GUI（图形用户界面）库，用于创建和管理窗口、按钮、文本框等 GUI 组件。它是一个封装了 Tcl/Tk 的工具包，允许 Python 程序员轻松创建用户友好的 GUI 应用程序。主要特点和功能包括：

① 组件丰富：提供了各种标准的 GUI 元素，如按钮、菜单、文本框、标签、滚动条等。

② 事件驱动：程序的运行基于事件驱动，这意味着程序在用户交互（如点击按钮、输入文本等）时响应并执行相应的代码。

③ 简单易用：相较于其他 Python GUI 库，Tkinter 因其简单性和广泛的支持成为初学者的首选。

④ 自定义布局：支持多种布局管理选项，如包管理（pack）、网格管理（grid）和绝对位置（place），使得组件的布局和定位更加灵活。

⑤ 跨平台：Tkinter 应用可以在 Windows、Linux 和 Mac OS X 上运行，无需修改代码。

在本项目代码中，Tkinter 用于创建一个全屏透明窗口，允许用户通过拖动鼠标来选择屏幕上的一个区域进行截图。这是通过 Tkinter 的 Canvas 组件实现的，用户的选择动作通过绑定的事件处理函数来捕捉并处理。

这两个库在该代码中的组合使用显示了如何通过键盘交互触发图形界面的操作，进而实现屏幕区域的选择和捕获，这在多种应用场景中非常有用，如屏幕录制软件、自动化测试等。

### 2.3.2 代码实现

（1） 设置参数和变量

设置了一个用于触发截图的热键（F12）。

（2） 定义截图函数

① 修改 capture_screen(region=None)：可以使用 pyautogui.screenshot 进行指定 region 的截图，然后使用 cv2.cvtColor 将图像从 BGR 转换成 RGB 格式。

② 定义 capture_selected()：此函数使用 Tkinter 创建一个全屏透明窗口，用户可以在窗口中拖动鼠标选择截图区域。选择完成后，调用 capture_screen 函数来截取指定区域的屏幕。

（3） 实时按键监听

主循环中，程序不断检测是否按下了定义的热键（F12）。如果检测到热键被按下，将调用 capture_selected()函数来让用户选择截图区域并捕获。

（4） 图像编码和文件保存

捕获到的图像首先被编码成 JPEG 格式，然后保存到本地文件系统中，便于 debug。同时，捕获到的 cv2 格式图像将被用于进行下一步处理。

# 3 阿里云服务端搭载

## 3.1 工作目的

本地服务器虽然能够稳定运行，但在处理高并发请求时表现不佳，且扩展性有限。为了解决这些问题，我们选择了阿里云作为新的托管平台。阿里云以其卓越的计算能力、高可用性和强大的安全保障，为我们的应用提供了一个理想的运行环境。

此外，此部分工作还包括了使用 Docker 容器化技术，这不仅简化了部署和管理过程，还增强了应用在不同环境间的一致性和可移植性。通过在 PyCharm 中配置 SSH 远程访问，我们能够直接在阿里云的 Docker 环境中开发和测试，这大大提高了开发效率和部署的便捷性。

总的来说，迁移到阿里云和采用 Docker 容器化的工作，旨在通过利用云计算资源来优化应用性能，提高服务的可靠性和可扩展性。

## 3.2 阿里云平台概述

### 3.2.1 阿里云的优势和选择原因

① 提供新用户资源免费试用：阿里云提供面向新用户的 ECS 实例免费试用服务，有多种套餐可以选择。

② 高可用性和可靠性：阿里云提供了高达 99.95%的服务可用性保障，这对于需要 24/7 运行的应用至关重要。通过多可用区的部署，阿里云能够确保即使在单个数据中心发生故障的情况下，服务也能持续可用。

③ 强大的计算能力：阿里云的计算服务（如 Elastic Compute Service, ECS）提供了多种配置选项，可以根据应用需求灵活选择 CPU、内存和存储资源。这种灵活性对于处理图像识别和数据转换任务尤其有价值，因为这些任务通常需要大量的计算资源和内存。

④ 全面的安全和合规措施：阿里云提供了全面的安全解决方案，包括网络安全、数据加密和访问控制等，确保用户数据的安全和隐私。此外，阿里云符合多个国际和地区的安全标准和合规要求，包括 ISO 认证和 GDPR。

⑤ 丰富的附加服务：阿里云不仅提供基础的计算和存储服务，还提供了包括数据库、人工智能、大数据分析等在内的丰富服务生态，这些服务可以帮助企业构建、部署和优化应用。

⑥ 优秀的客户支持和生态系统：阿里云有着广泛的合作伙伴网络和活跃的开发者社区，为用户提供技术支持和业务扩展的丰富资源。

### 3.2.2 阿里云实例参数

为了支持我们的图像处理和文本转换服务，我们选择了阿里云的 ecs.e-c1m2.large 实例规格，这为我们的应用提供了必要的计算能力和内存资源。具体配置如下：

（1） CPU 和内存：实例配备了 2 核（vCPU）和 4 GiB 内存，这样的配置可以有效处理并发请求和数据密集型的任务，确保应用运行的高效性和响应速度。

（2） 网络配置：

① 弹性公网 IP：增强了网络的可靠性和灵活性，使我们能够灵活地处理 IP 地址的变更和网络流量的管理。

② 操作系统：选择了基于 Alibaba Cloud Linux 3.2104 LTS 64 位的自定义镜像，这是一个为云优化的操作系统，提供了稳定和安全的运行环境。

③ 网络带宽：配置了最高 100 Mbps 的公网带宽，按使用流量计费。这种带宽配置保证了即使在数据传输高峰期也能保持良好的连接速度和低延迟。

④ 这些配置提供了一个隔离的网络环境，增加了网络的安全性，并允许我们灵活地管理子网和 IP 地址，适应不同的部署需求，共同支持了我们服务的可靠运行和扩展性，确保了用户在使用图像识别和文本转换功能时的顺畅体验。

## 3.3 服务端应用架构

### 3.3.1 目标架构

在迁移之前，我们的服务端应用主要运行在本地服务器上，采用传统的单体架构。该应用负责接收图像数据，通过图像处理技术转换为文本，然后再将文本转换为盲文。虽然这种架构在早期阶段表现可靠，但为了扩展应用场景，应对预想中的用户数量的增加和处理需求的上升，我们选择了将服务端配置在云服务器上。

为了提高应用的可伸缩性和可靠性，我们决定将应用迁移到阿里云，并采用更加现代化的微服务架构。在阿里云上，应用被拆分为几个独立的服务，每个服务负责应用的一个特定功能。这些服务通过 RESTful API 或消息队列进行通信，这样做不仅提高了系统的响应性和可维护性，还使得各个组件可以独立扩展。

### 3.3.2 核心组件

（1）图像接收服务：负责接收来自客户端的图像数据，并对数据进行初步的验证和预处理。这一服务确保了数据的完整性和格式正确性，为后续的处理流程打下基础。

（2）图像处理服务：使用开源库如 pytesseract 进行 OCR（光学字符识别），将图像中的文字转换为电子文本。这一服务的设计使得其可以独立于其他服务进行扩展，根据处理需求动态调整资源。

（3）文本到盲文转换服务：使用自定义库 MangWen 将文本转换为盲文。这一服务不仅转换文本，还优化了盲文的格式以适应不同的显示设备和输出需求。

（4）数据存储服务：负责存储处理过程中产生的数据和最终结果。我们选择了阿里云的高性能数据库服务，以保证数据的安全性和高可用性。

### 3.3.3 支持服务

（1）日志和监控服务：整合了阿里云的日志服务和性能监控工具，实时监控应用的健康状况和性能指标。这些工具帮助我们快速定位问题并进行优化。

（2）安全服务：实施了多层次的安全策略，包括网络安全组、数据加密和访问控制，确保服务和数据的安全。

### 3.3.4 部署和运维

在阿里云平台上，我们采用 Docker 容器化技术部署各个服务。这不仅简化了部署过程，还使我们能够利用 Kubernetes 等容器编排工具进行自动化管理。容器化带来的灵活性和可移植性极大地提高了我们的运维效率和系统的可靠性。

## 3.4 Docker 容器化与 PyCharm 集成

### 3.4.1 Docker 容器化

为了提高我们服务端应用的部署效率和环境一致性，我们采用了 Docker 容器化技术。Docker 提供了一个轻量级的虚拟化解决方案，允许我们在隔离的环境中打包和运行应用，包括其依赖和配置。

我们的 Dockerfile 基于官方的 Python 3.8-slim 镜像，这是一个精简版的 Python 镜像，优化了容器的大小和运行效率。以下是 Dockerfile 的主要配置步骤：

（1）基础镜像和工作目录：从官方 Python 3.8-slim 镜像开始，设置容器内的工作目录为/app。

（2）安装依赖：安装了 tesseract-ocr 及其中文语言包，以支持图像到文本的转换功能。此外，为了支持远程开发，我们还安装了 openssh-server，并进行了相应的配置以允许 SSH 访问。

（3）配置 SSH 服务：生成 SSH 密钥，并调整配置以允许 root 用户通过密码登录。这一步是关键，因为它使得我们可以通过 PyCharm 或其他 IDE 远程连接到容器。

（4）复制文件和安装 Python 依赖：将所需的 Python 脚本和依赖文件复制到容器中，并使用 pip 安装这些依赖。

（5）端口暴露和服务启动：在容器中暴露 22 端口（SSH 服务）和 5000 端口（通过宿主与单片机通信），并设置容器启动时同时运行 SSH 服务和我们的 Python 服务端应用。

### 3.4.2 PyCharm 集成

（1）集成优势

利用 Docker 和 PyCharm 的集成，我们能够通过 SSH 直接在阿里云的 Docker 环境中进行代码开发和测试。这种集成提供了几个关键优势：

① 环境一致性：开发人员可以在本地 PyCharm IDE 中直接连接到远程 Docker 容器，这保证了开发和生产环境的完全一致。

② 便捷的开发体验：通过 PyCharm 的远程解释器功能，开发人员可以直接在 IDE 中运行和调试代码，就像在本地环境中一样，无需手动配置复杂的远程环境。

③ 高效的资源利用：将开发环境放在云端，可以更有效地利用阿里云的计算资源，尤其是在处理大规模数据或需要高性能计算时。

④ 安全和访问控制：通过配置 SSH 密钥和安全策略，我们可以确保只有授权的开发人员能够访问远程开发环境。

（2）集成步骤

① 准备远程 Docker 环境：如 3.4.1 所述。

② 在 PyCharm 中配置 SSH 连接：转到 File > Settings (或 PyCharm > Preferences on macOS)；在设置菜单中，选择 Project: [项目名] > Python Interpreter；点击右上角的齿轮图标，选择 Add；在弹出的窗口中，选择 SSH Interpreter，在 New SSH Configuration 的窗口中输入 Docker 容器的 SSH 连接详情。这里有一点需要注意，即 POST 需要是宿主映射到 docker 的端口，而不是 docker 的端口号。

③ 配置解释器路径：一旦连接建立，PyCharm 会询问远程主机的 Python 解释器路径。通常，这会是 /usr/bin/python3 或 /usr/local/bin/python3，具体取决于 Docker 镜像中 Python 的安装位置。选择对应的 Python 解释器后，点击 OK。

④ 同步项目文件：在 PyCharm 中，可以设置自动上传本地更改到远程环境。具体操作为转到 Tools > Deployment > Configuration，设置远程服务器配置，并开启自动上传。

## 3.5 通信端口设置

（1）阿里云安全组配置：

在阿里云平台上，安全组充当虚拟防火墙，用于控制进出实例的数据流。为了确保应用的安全性和连通性，我们调整了安全组规则，允许特定端口的流量通过。

对于本项目，我们开放了 8022 端口用于和客户端的 SSH 连接，以及 5000 端口用于和单片机传输数据。

（2）Docker 容器的端口映射：

在创建 Docker 容器时，我们通过 docker run 命令映射了宿主机和容器之间的端口。这样做是为了确保从外部网络到容器的网络流量能够通过指定的端口正确路由。

使用的命令如下：docker run -it --name ttb -v /ttb/:/workspace/qa -p 8022:22 -p 5000:5000 -d ttb-app

该命令中的-p 8022:22 将容器的 22 端口映射到宿主机的 8022 端口，用于 SSH 连接；-p 5000:5000 将容器的 5000 端口映射到宿主机的 5000 端口，用于和单片机传输数据。

# 4 总结

在本项目中，我主要负责了客户端图片抓取功能的实现及服务端的搭载到阿里云平台上。

客户端图片抓取功能允许用户根据需求进行定时或主动的图片抓取。通过使用 OpenCV、pyautogui 和 keyboard 等库，设计了一种能够在用户指定时间自动捕获屏幕图片，并允许用户通过热键激活，选择特定屏幕区域进行截图的功能。此外，Tkinter 库被用于创建一个全屏透明窗口，让用户可以直观地选择截图区域，从而增强了用户交互的便捷性与准确性。

至于服务端部分，为了应对可能的高并发请求并增强系统的可扩展性，我将服务端迁移到了阿里云。阿里云不仅因其卓越的计算能力和高可用性被选为我们的云服务平台，其全面的安全和合规措施也确保了项目的安全性需求得到满足。通过利用 Docker 容器化技术，我简化了部署和管理流程，提高了开发与测试的效率。此外，通过在 PyCharm 中配置 SSH 远程访问，使得团队能够直接在阿里云的 Docker 环境中进行开发和测试，这不仅保证了开发环境的一致性，还利于快速迭代和问题定位。

总的来说，这部分工作不仅增强了系统的功能性和可靠性，而且通过引入先进的云计算资源，有效地提升了服务端的处理能力和系统的整体性能。这些技术的成功实施为整个项目的顺利执行奠定了坚实的基础，确保了项目目标的达成。