嵌入式物联网大作业

0 小组成员

姓名	学号
侯为栋	191250045
于环宇	191250185
蒋睿兮	191250063
李福梁	191250068
李智强	191250078

1 项目介绍

我们使用 EdgexFoundry 作为边缘平台, Atlas200DK 作为端设备.

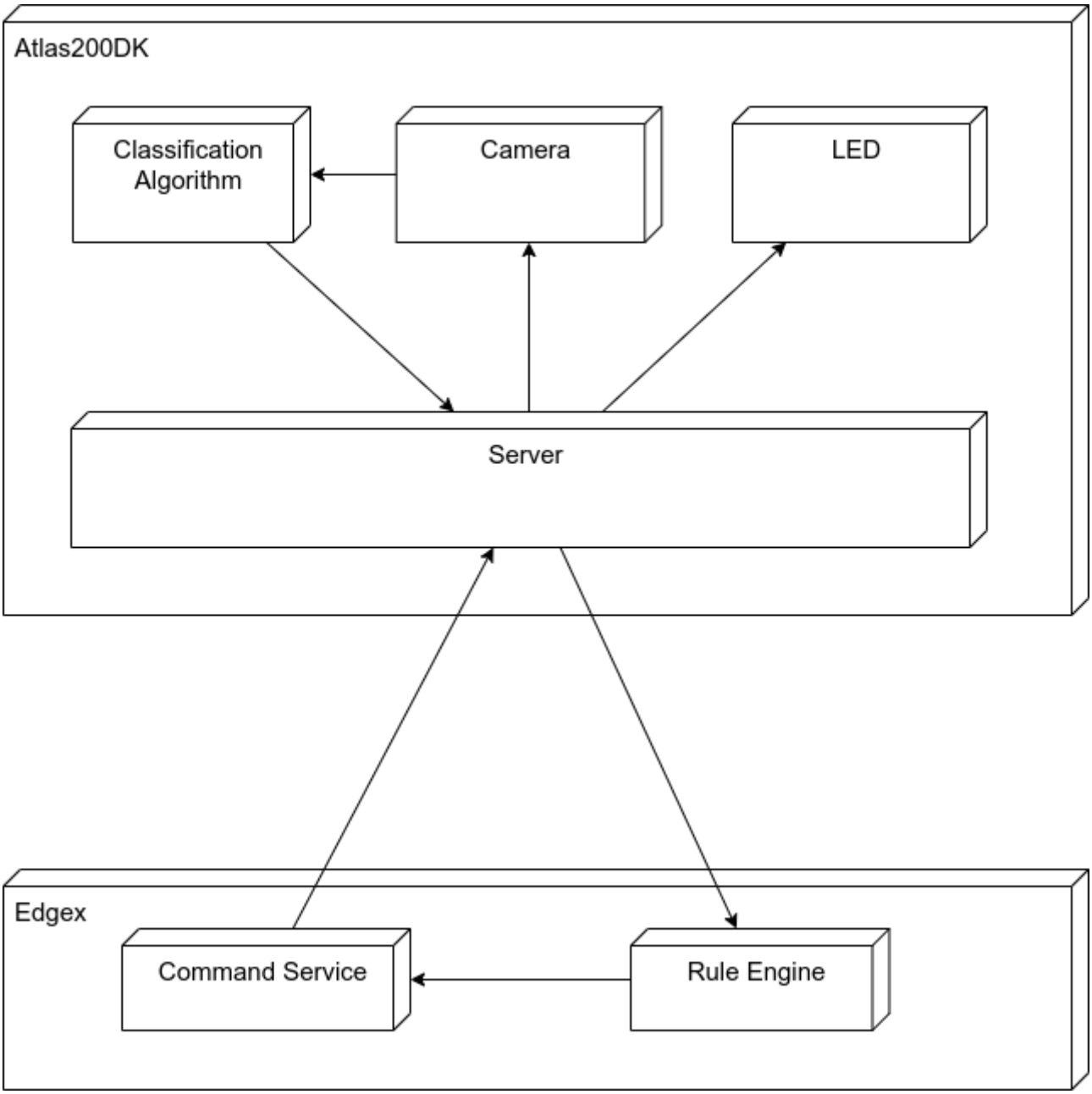
我们设想的场景是垃圾分类. Atlas200DK 通过扩展摄像头拍摄垃圾, 并且使用其搭载的垃圾分类算法进行分类; 随后 Atlas200DK 将分类结果传给 edgex, 并通过 edgex 的规则引擎控制 Atlas200DK 上的 LED.

例如, Atlas200DK 拍到一张电池的照片, 它通过分类算法判断出这是有害垃圾后, 将 "有害垃圾" 这一信息传递给 edgex. Edgex 的规则引擎接收到这一信息后, 便会调用 Atlas200DK 的某个 LED 接口.

LED 只是我们展示分类结果的方式, 在实际运用中, 可以换成语音提示, 文字提示等.

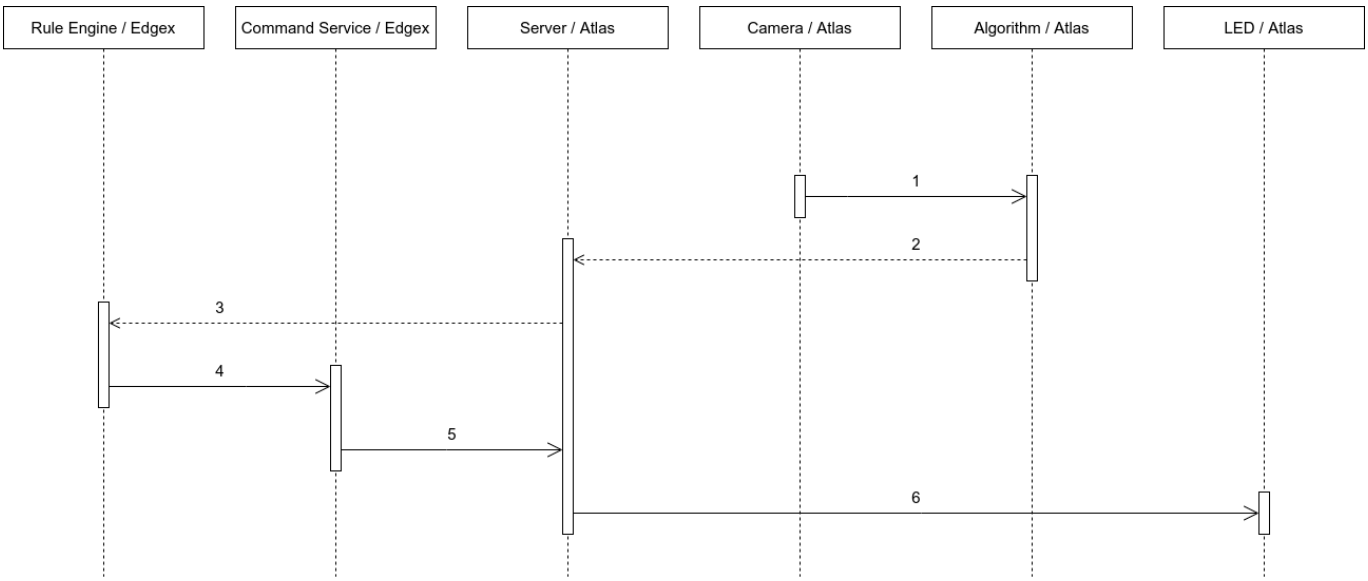
上述过程是一个持续的状态: Atlas200DK 开始运行后会一直检测图像状态, 若图像发生改变 (在具体场景中, 可能是摄像头一直监控垃圾投放口, 如果有手伸入准备扔垃圾时就会触发检测) 就会运行分类算法并将结果发给 edgex, 从而导致自己 LED 状态的变化.

项目部署图如下:



我们将 Atlas200DK 抽象为一个 REST 设备, 通过一个服务器来接收 REST 调用.

一次流程如下图所示:



1. Atlas200DK 上的摄像头将发生目标照片传递给分类算法;
2. 分类算法将分类结果传递给 server;
3. Server 将分类结果以 event 的方式发送给 edgex 规则引擎;
4. Edgex 通过规则引擎请求 edgex command service 进行对应的服务调用;
5. Edgex command service 找到正确的 Atlas200DK 地址并向其服务端发送 LED 修改请求;
6. Server 接收到请求后转发给 LED, LED 状态得到修改.

上述流程会在 Atlas200DK 摄像头检测到图像发生改变时触发.

我们的演示视频在 billbill 上.

2 总体部署流程

我们将总体部署的流程先简要介绍一遍, 方便读者有的放矢.

1. 在某台可以访问到 Atlas200DK 的机器上安装启动 edgex, 请参考 `3 EdgeX 安装部署`;
2. 进入 `edgex` 文件夹完成 edgex 的配置, 请参考 `4 EdgeX 配置`;
3. 进入 Atlas200DK, 完成基本环境部署, 请参考 `5.1 Atlas 运行环境配置`;
4. 将 `atlas` 文件夹移动到 Atlas200DK 中, 启动 camera 和 server, 请参考 `5.3 Camera` 和 `5.5 Server`;
5. 获取 GPIO 引脚的控制权限, 请参考 `5.2 LED`.

垃圾分类算法在完成 5.1 后应该可以直接运行. 但是如果出现问题还请详细阅读文档与源码.

3 EdgeX 安装部署

参考官方文档的安装教程.

我们使用 hanoi 发行版.

前提条件: 安装 docker, docker-compose, git, make.

运行以下命令:

```
git clone https://github.com/edgexfoundry/edgex-compose.git
cd edgex-compose
git checkout hanoi
```

需要微调我们需要的 `docker-compose` 文件的网络配置:

```
vim docker-compose-hanoi-no-secty.yml
# 以下为 vim 命令
# :%s/127.0.0.1://g
# :wq
```

通过以下命令启动服务:

```

```

```
make run no-secty
make run-ui
```

通过以下命令查看容器启动情况:

```
docker-compose -p edgex -f docker-compose-hanoi-no-secty.yml ps
```

4 EdgeX 配置

参考官方文档的 Walkthrough.

我们已经将所有命令整理成 shell 脚本, 如果不想细究可以直接执行然后跳过.

运行以下命令:

```
cd edgex
bash -x all.sh SERVER_ADDR SERVER_PORT EDGEX_ADDR PROFILE_PATH RULE_PATH
```

其中:

- `SERVER_ADDR` 对应 edgex 的 ip 地址;
- `SERVER_PORT` 对应 server 开的 port;
- `EDGEX_ADDR` 对应 edgex 所在机器的 ip 地址;
- `PROFILE_PATH` 对应要上传的 device profile 的路径, 这里可以直接填 `./atlas_profile.yaml`;
- `RULE_PATH` 对应 `rule.sh` 的路径, 这里可以直接填 `./rule.sh`

若想了解具体流程请 RTFSC.

5 Atlas

Atlas 为华为 Atlas200DK 开发版的简称, 我们在 Atlas 上接入了 LED 和摄像头.

请将本项目的 `atlas` 文件夹拷贝到 Atlas200DK 环境上.

本项目 `atlas` 中所有子项目的编译文件都没有 ignore, 方便读者在无法编译的时候可能可以直接运行.

5.2 后的所有命令都是在 Atlas200DK 上运行; 5.1 看语境.

5.1 Atlas 运行环境配置

> 请读者充分发挥自己的 debug 能力...

假设读者拥有一块完全没有配置过的 Atlas200DK 开发版.

请务必在用户 `HwHiAiUser` 上完成配置, 否则会出现很多麻烦. 样例很多地方是硬编码路径.

这里只阐述和 Atlas200DK 直接相关的, 也就是运行环境.

请参考官方文档的运行环境配置进行基础环境搭建.

我们使用的 camera 来自官方样例, 注意 tag 是 `v0.3.0`.

请参考上述官方样例中的 README 进行环境搭建, 尤其注意环境准备和依赖安装.

按照官方文档的流程会缺失 `ascend_ddk` 的配置, 这里可以参考这篇文章.

最终生成 `ascend_ddk` 文件夹并且传输到 Atlas200DK 中, 并扩展环境变量:

```
export
LD_LIBRARY_PATH=/home/HwHiAiUser/ascend_ddk/arm/lib:/home/HwHiAiUser/Ascend
/acllib/lib64:${LD_LIBRARY_PATH}
```

我将最后运行环境应该有的环境变量罗列一下, 但不保证一定正确:

```
export LD_LIBRARY_PATH=/usr/lib64
export
LD_LIBRARY_PATH=/home/HwHiAiUser/ascend_ddk/arm/lib:/home/HwHiAiUser/Ascend
/acllib/lib64:${LD_LIBRARY_PATH}
export
LD_LIBRARY_PATH=/home/HwHiAiUser/Ascend/acllib/lib64:${LD_LIBRARY_PATH}
export PYTHONPATH=/home/HwHiAiUser/Ascend/pyACL/python/site-
packages/acl:$PYTHONPATH
export ASCEND_AICPU_PATH=/home/HwHiAiUser/Ascend
export PKG_CONFIG_PATH=/usr/share/pkgconfig/
export PYTHONPATH=$HOME/ascend_ddk/:$PYTHONPATH
export PATH=$PATH:/home/HwHiAiUser/ascend_ddk/arm/bin
```

5.2 LED

通过使用 Atlas200DK 上的 40 个 GPIO 接口来控制 led.

控制程序具体介绍见 `atlas/led` 的 README.

请至少看完上述文档中介绍如何获得 GPIO 引脚的操作权限的部分.

可以方便地使用命令来控制:

```
cd atlas
bash led.sh NUM
```

`NUM` 对应五种类别的垃圾, 可取值为 `0, 1, 2, 3, 4`.

5.3 Camera

可以使用以下命令直接控制摄像头拍照:

```
cd atlas
bash camera.sh NAME
```

`NAME` 为自定义照片名, 运行结束后照片会存放在 `camera.sh` 所在目录下.

但是我们需要启动一个一直运行摄像头的程序. 该程序一直进行拍照并对比图片差异, 如果两张图片差异过大就认为是检测到了垃圾, 并开启垃圾分类流程.

使用如下命令启动:

```
cd atlas
bash detect.py EDGEX_ADDR DEVICE VALUEDESCRIPTOR
```

其中:

- `EDGEX_ADDR` 对应 edgex 部署地址;
- `DEVICE` 对应设备注册名, 这里可以直接填 `atlas`;
- `VALUEDESCRIPTOR` 对应值描述符, 这里可以直接填 `idx`.

如果对后两者有不解, 那你需要细究 EdgeX 配置一节.

5.4 垃圾分类算法

本处使用的垃圾分类算法参考官方垃圾分类样例, 不过我们基本重写了对外接口.

本算法用到 python 依赖 `atlas_utils`. 如果在 4.1 漏掉的话请参考官方文档对应部分补全.

使用以下命令直接使用垃圾分类算法:

```
cd atlas
python3 rubbish/classify.py /path/to/xxx.jpg
```

会返回 `0, 1, 2, 3, 4`, 分别对应干垃圾, 可回收物, 湿垃圾, 有害垃圾, 其他垃圾.

代码量不多, 可以很方便地 RTFSC.

5.5 Server

直接运行即可:

```
cd atlas
python3 server.py
```

如果出现依赖问题直接 `pip3` 安装即可.