Файловая организация базовых программных средств в робототехнических платформах

github:ArtemZaZ

Январь 2020

Version 0.1b

1 Введение

С увеличением возможностей робототехнических платформ, а также их разнообразия в нашем отделе, увеличивается объем и сложность программных частей и модулей, необходимых для их эксплуатации. Одной из проблем, возникающих в результате этого, является отсутствие организации самих программных средств.

2 Edubot

В качестве примера и описания дальнейших концепций будем рассматривать робототехническую платформу Edubot (рисунок 1).

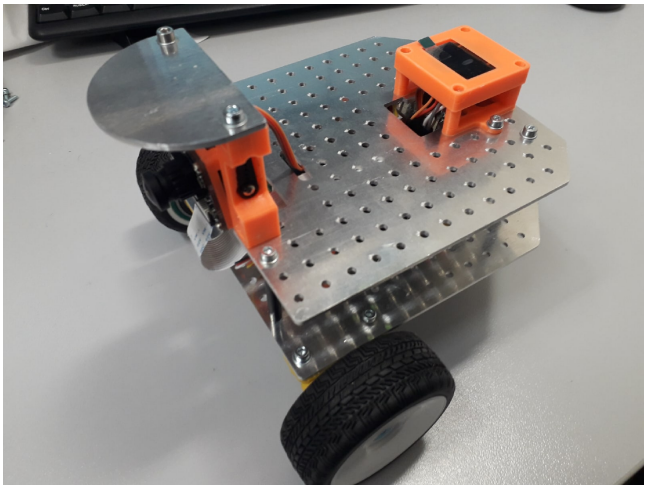

Рис. 1: Edubot

Структурную схему аппаратной части платформы можно увидеть на рисунке 2. В настоящей версии платформы аппаратная часть сожержит в себе одноплатный компьютер Raspberry pi в качестве базы, к одноплатному компьютеру присоединяется шилд, который

обеспечивает взаимодействие с перефирией: моторы, сервоприводы, кнопка, пищалка, датчик тока, а также предоставляет физические разъемы для дисплея и других I2C устройств. Также к одноплатному компьютеру подключена цифровая видеокамера.

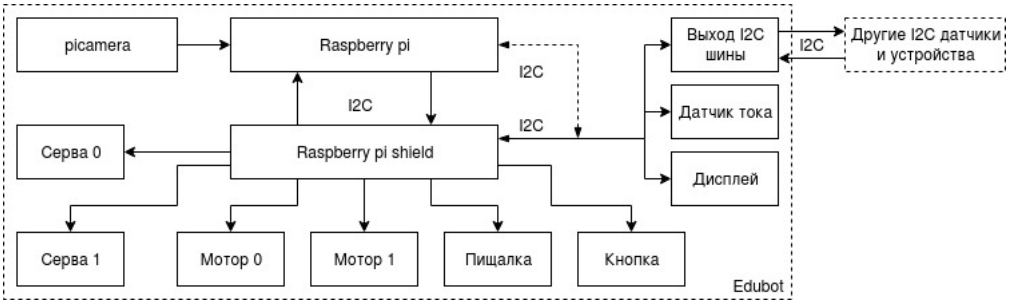

Рис. 2: Структурная схема аппаратной части платформы Edubot

Изначально, данная платформа, как и другие, выступает в качестве учебной платформы, предназначенной для обучения написания различных программ, поэтому, в дополнение к аппаратному обеспечению, для каждой платформы пишется базовый программный интерфейс на языке программирования python для взаимодействия и управления аппаратной частью (рисунок 3). Данный программный интерфейс представляет собой высокоуровневый интерфейс для взаимодействия с платформой: различные конфигурационные файлы, инструменты, а также низкоуровневый интерфейс, который просто обеспечивает взаимодействие с I2C устройствами (написан также на python).

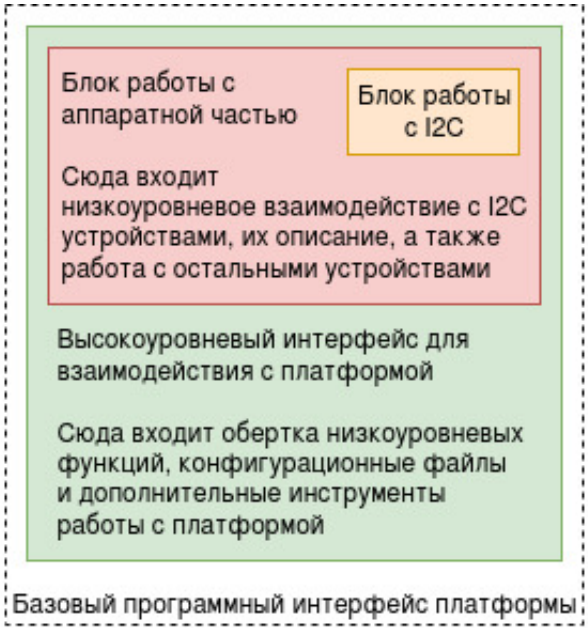

Рис. 3: Компоненты базового программного интерфейса

3 Проблема взаимодействия

Предполагается, что на одной платформе одновременно будут работать множество программ, которые будут иметь доступ к аппаратной переферии (в случае Edubot'а это такие программы: автоматическая программа вывода сетевой информации на дисплей,

различные серверы, программы управления и т.д.). Если каждая такая программа в себе будет иметь копию базового программного интерфейса или его часть (рисунок 4 (a)), то в этом случае при физическом, аппаратном или другом изменении конфигурации платформы (изменение версии платформы, смена платформы, замена аппаратных компонентов и т.д.) или же просто в случае обновления самого базового программного интерфейса, придется изменять или обновлять каждую копию данной программной части. Для выхода из этого положения, самым простым способом будет использовать одну и ту же копию программного интерфейса, просто импортируя ее из какого-либо одного места (рисунок 4 (b)).

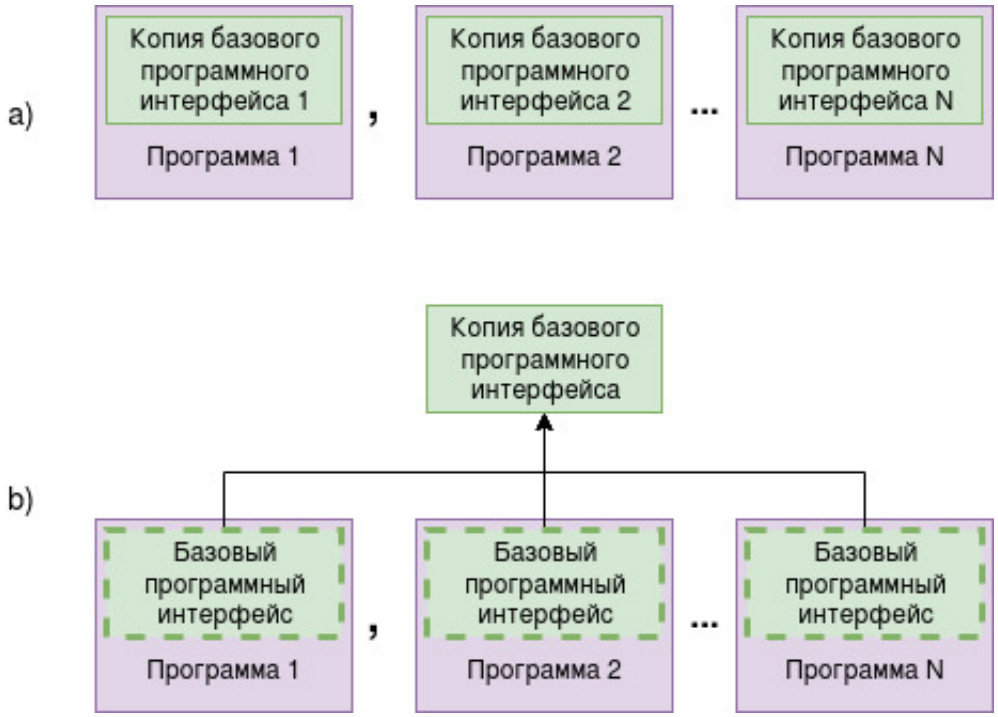

Рис. 4: Взаимодействие программных компонент

Если рассматривать платформу Edubot в качестве примера, то вышеописанное можно было бы просто реализовать, если низкоуровневый программный интерфейс выпустить в качестве стандартного пакета python, который бы устанавливался вместе со всеми пакетами python и импортировался бы оттуда. Однако, помимо самого низкоуровневого интерфейса присутствуют также различные дополнительные инструменты и конфигурационные файлы, которые варьируются от платформы к платформе, из-за чего придется аналогично рисунку 4 (a) использовать множество их копий в различных программах, что особо ситуацию не изменит.

По этой причине предлагается попробовать использовать определенную организацию файлов базового интерфейса и других вспомогательных программ для решения проблемы.

4 Файловая организация

Для обеспечения платформонезависимости управляющего кода, а также облегчения создания и деплоя управляющих и иных программ для робототехнических платформ предлагается использовать следующую файловую организацию (рисунок 5).

В домашней директории располагается директория platform, которая будет содержать все ресурсы и инструменты для работы с роботом. В директории auto располага-

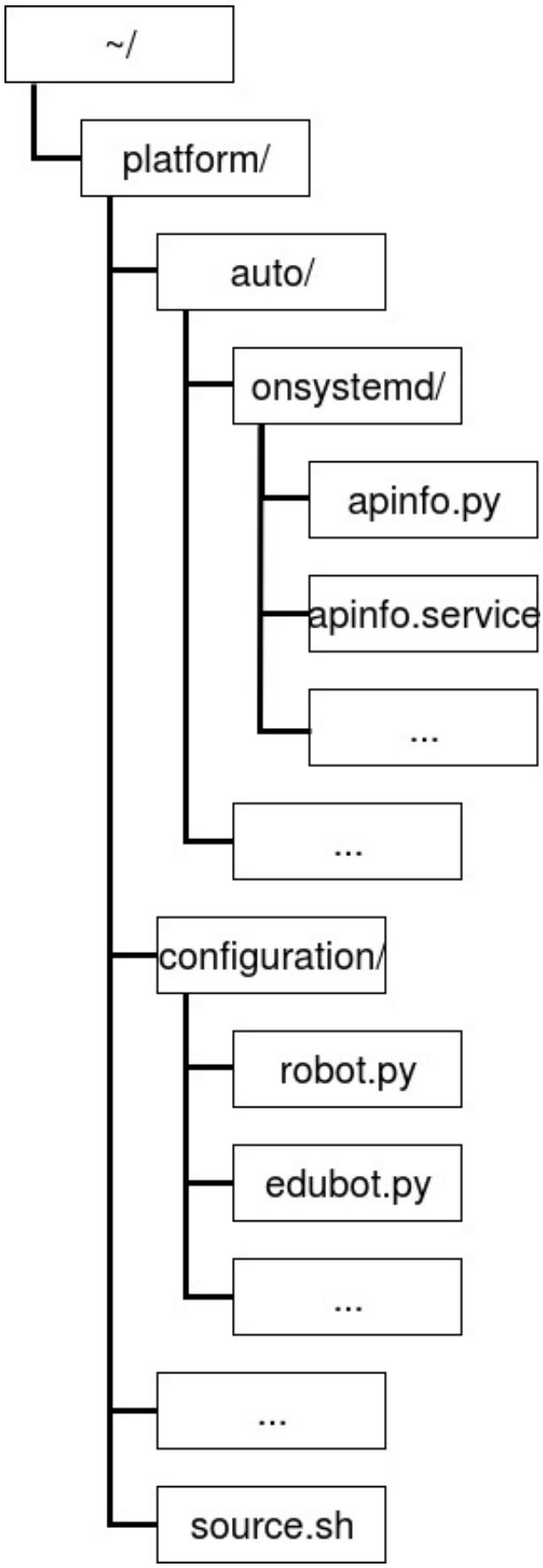

Рис. 5: Пример предлагаемой файловой организации

ются автоматические скрипты и сервисы, которые должны запускаться при включении робота, начале ssh сессии и т.д. В данном примере в auto в поддиректории onsystemd (скрипты, которые должны запускаться при включении робота) лежит скрипт автоматического вывода сетевой информации платформы на дисплей при включении: apinfo.py и его сервис apinfo.service. В поддиректории configuration находится файл конфигурации робота (robot.py) и необходимые для работы ресурсы. Пример конфигурационного файла для платформы Edubot можно увидеть в листинге 1. Файл source.sh автоматизирует некоторую работу: обрабатывает, переносит и запускает сервисы, настраивает другие автоматические программы, производит обновление ПО и т.д. Данная файловая организация пока находится на этапе тестирования и возможно будет дополняться и изменяться.

Листинг 1: Конфигурационный файл платформы Edubot robot.py

```
""" Edubot configuration """
from configuration.edubot import EduBot, MotorMode
import smbus

servoPosLen = 255
middleServoPos = int(servoPosLen / 2)

bus = smbus.SMBus(1)
robot = EduBot(bus)


def move(speed):
    robot.setParrot0(int(-speed))
    robot.setParrot1(int(speed))


def rotate(speed):
    robot.setParrot0(int(speed))
    robot.setParrot1(int(speed))


def setCamera(scale):
    scale = min(max(-1, scale), 1)
    robot.beep()
    robot.setServo0(int(middleServoPos - scale * servoPosLen))


def initializeAll():
    robot.online = True
    robot.start()
    robot.setMotorMode(MotorMode.MOTOR_MODE_PID)
    robot.setParrot0(0)
    robot.setParrot1(0)
    robot.setServo0(int(middleServoPos))
    robot.setServo1(int(middleServoPos))
```

```
def release():
    robot.exit()


@property
def online():
    return robot.online


@online.setter
def online(val):
    robot.online = val
```

5 Заключение

Представленная в документе концепция файловой организации для робототехнических платформ в ближайшее время пройдет проверку на валидность, в случае успеха, данный документ будет расширяться и дальше.