Организация автоматизированного тестирования встраиваемого программного обеспечения

Кижнеров Павел Александрович
группа 21.M07-мм
руководитель Терехов А. Н.

СПбГУ

4 июня 2022 г.

Введение

Данная работа выполняется в рамках разработки прошивки для новой версии фитнес-браслетов, разрабатываемых американской компанией-производителем для продажи на американском и мировом рынках

Введение: особенности разработки

- запуск кода возможен только после загрузки скомпилированного бинарного исполняемого файла в память контроллера
- мониторинг результатов осуществляется посредством чтения последовательного порта

Введение: особенности разработки

Следствия:

- увеличивается трудоемкость отладки
- снижаеся концентрацию внимания разработчика
- снижается наглядность отладки
- усложняется процесс разработки

Постановка задачи

- автоматизация тестов, мониторинг
- интеграция техники символьного исполнения
- анализ результатов тестов, подсчет метрик

Обзор: традиционные решения

- Gtest
- Catch
- Mettle
- Boost.Test

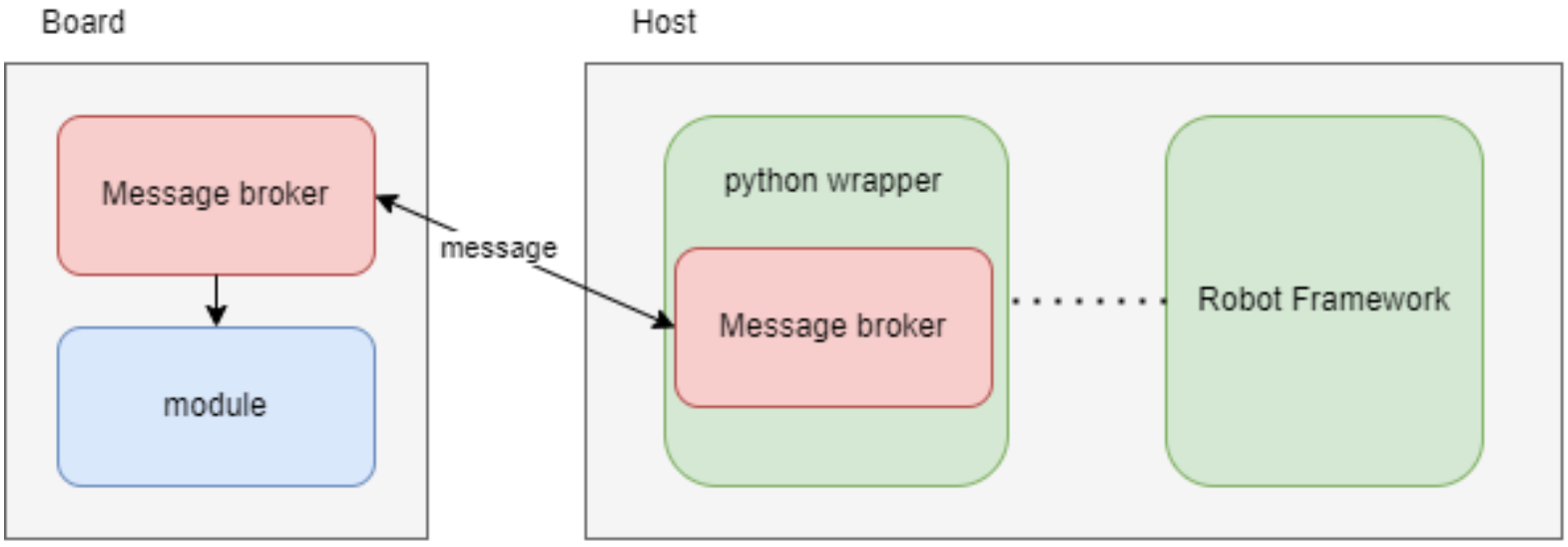
Для платформонезависимого кода используется Gtest

Обзор: решения для встраиваемых систем

- TETware RT
- OpenTest
- autotestnet
- DejaGnu
- Robot Framework

Каждый требует реализацию интерфейса коммуникации с контроллером

Реализация: диаграмма взаимодействия



- message broker обеспечивает надежную доставку пакетов
- python wrapper интегрирует message broker в Robot Framework
- message - TLV пакет, вызывающий требуемую функцию в прошивке

Реализация: символьное исполнение

Используется KLEE
Порядок подготовки:

- подключение библиотеки KLEE в проект
- компилятор - clang
- компоновщик - lld
- отключение отпимизации
- настроить генерацию LLVM IR промежуточных файлов

В рамках данной работы контекст - CMake

Реализация: символьное исполнение

Порядок использования:

- определение символьных аргументов функции в коде
- сборка проекта
- компоновка промежуточных LLVM IR файлов в единый
- передача LLVM IR биткода проекта в символьный движок
- запуск на сгенерированных входных данных

Результаты

- message broker
- обертка над message broker
- интеграция обертки с Robot Framework
- интеграция с Jenkins CI/CD
- скрипты настройки окружения KLEE
- настроена сборка проекта для KLEE
- найдены и исправлены ошибки в ключевых функциях

Результаты

```
19  int8_t
20  hb_klee_test(void)
21  {
22      int num;
23  #ifdef HB_KLEE
24      klee_make_symbolic(&num, sizeof(num), "num");
25  #endif
26      printf("%s\n", hb_screen_helper_sprintf_num(num));
27      return 0;
28  }
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ ТЕРМИНАЛ ПОРТЫ

```
-476788 044
-786424 000

KLEE: done: total instructions = 117800
KLEE: done: completed paths = 64
KLEE: done: partially completed paths = 0
KLEE: done: generated tests = 64
```

```c
int8_t
hb_klee_test(void)
{
    int num;
#ifdef HB_KLEE
    klee_make_symbolic(&num, sizeof(num), "num");
#endif
    printf("%s\n", hb_screen_helper_sprintf_num(num));
    return 0;
}
```

ПРОБЛЕМЫ ВЫХОДНЫЕ ДАННЫЕ КОНСОЛЬ ОТЛАДКИ **ТЕРМИНАЛ** ПОРТЫ

```
-3816 090
-6729 598
-6144 000
-8309 008
405217 014
411165 008
817405 664
603261 000
1735425 952
-97012 126
1001106 076
2116183 000
1470057 008
```