Федеральное государственное автономное образовательное учреждение
высшего образования
Санкт-Петербургский политехнический университет Петра Великого
Высшая школа автоматизации и робототехники

НАУЧНО-ИССЛЕДОВАТЕЛЬСКАЯ РАБОТА
АВТОМАТИЧЕСКОЕ РАЗВЕРТЫВАНИЕ КОДА НА МИКРОКОНТРОЛЛЕРЫ ARDUINO

по направлению подготовки 15.03.06 «Мехатроника и робототехника»
направленность (профиль) 15.03.06_01 «Проектирование и конструирование мехатронных модулей и механизмов роботов»

Выполнил
студент гр. 3331506/10102 В.А. Блохина

Руководитель
Доцент ВШАиР кф-м.н. М.С. Ананьевский

Санкт-Петербург
2024

Содержание

1. ВВЕДЕНИЕ

КЛЮЧЕВЫЕ СЛОВА: АВТОМАТИЧЕСКОЕ РАЗВЕРТЫВАНИЕ, ARDUINO МИКРОКОНТРОЛЛЕРЫ, БИЛД (BUILD), ДЕПЛОЙ (DEPLOY), CI/CD

Тема научно-исследовательской работы: «Автоматическое развертывание кода на микроконтроллеры Arduino в среде Linux».

Данная работа посвящена автоматизации работы с Arduino микроконтроллерами и ускорению доставки рабочего кода на платы.

В последние годы в разработке стала особо популярна практика непрерывной интеграции (Continuous Integration) и непрерывной доставки (Continuous Delivery) кода (CI/CD). Цель CI – обеспечение автоматизированной сборки кода, в нашем случае его компиляции. CD обеспечивает автоматическое развертывание кода на серверах, в нашем случае это Arduino микроконтроллеры. Так как постоянный ручной деплой кода на несколько плат времязатратный процесс, было решено внедрить практику CI/CD.

Помимо автоматизации процесса в работе было уделено внимание хранению кода и контролю его целостности. Был создан репозиторий кода, в котором хранятся файлы с расширением .ino (скетчи). Репозиторий имеет определенную структуру для поддержания порядка в файлах.

Целостность кода обеспечивается сравнением хеш-сумм файлов во время автоматической сборки. При несоблюдении своевременного обновления файла, хранящего значение хеш-суммы, процесс автоматической сборки прерывается во избежание подмены файлов с кодом.

Файл с кодом, а также репозиторий, хранящий .ino файлы и README.md файл с подробным описанием размещен в публичном репозитории Github.com (URL: https://github.com/soomrack/M2024/tree/main/blokhina_va/srw, актуально на 13.05.2024).

2. ХОД РАБОТЫ

2.1. Постановка задачи

В работе можно выделить три крупные задачи, которые требуется решить:

1. Автоматический билд кода под определенную плату:

Каким образом будут компилироваться файлы, в какой среде, какие инструменты будут задействованы в этом процессе и какой язык лучше подойдет для решения задачи.

2. Проверка скетчей на оригинальность:

Как будет контролироваться версия скетчей, как пользователь будет узнавать об изменениях в файлах.

3. Автоматический деплой кода сразу на несколько подключенных плат.

Как пользователь будет контролировать загрузку скетчей на плату, как будет выводиться информация о рабочих портах.

Для решения поставленных задач была выбрана операционная система семейства Linux Ubuntu 22.04 LTS. Для написания скрипта, осуществляющего автоматическое развертывание, выбран язык командной оболочки bash. Также для проведения работы требовалась утилита arduino-cli для работы с Arduino платами через терминал Linux с помощью команд.

Ход работы состоял из следующих шагов:

1. Установка утилиты arduino-cli;
2. Загрузка скрипта и репозитория кода на рабочую станцию (сервер);
3. Создание файлов с хеш-суммами для каждого файла с кодом;
4. Определение необходимых переменных для работы скрипта, в т.ч. и название скрипта для билда и деплоя;
5. Проверка существования указанного скетча с расширением .ino в рабочей директории (указывается на первом шаге);
6. Проверка совпадения хеш-сумм во избежание подмены файла с кодом;
7. Вывод в терминал списка всех портов, к которым подключены Arduino платы;

8. Создание директории для хранения скомпилированных файлов;
9. Компиляция кода;
10. Деплой кода на подключенные микроконтроллеры.

2.2. Подготовка системы

Для начала разберем первоначальную подготовку системы и окружения. Так как предполагается, что ведется работа с несколькими подключенными платами, необходимо убедиться, что на рабочей станции (сервере) достаточно usb портов для подключения нужного количества плат. Либо же воспользоваться usb разветвителями. Пример подключения приведен на Рисунок 1.

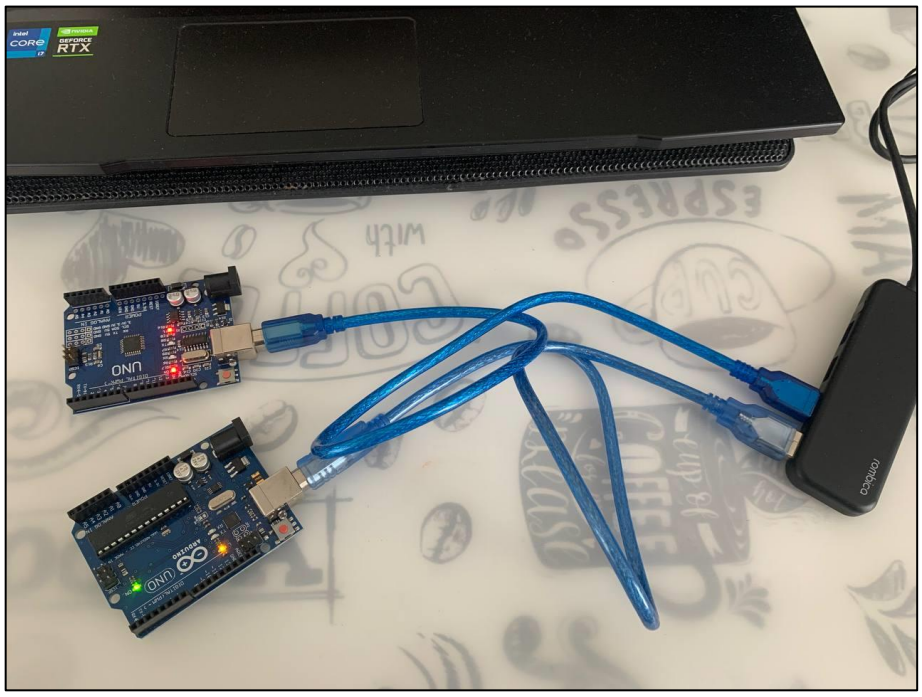

Рисунок 1 – Подключение нескольких arduino плат

Администратор системы (им может быть сам пользователь) должен убедиться в том, что используется оболочка bash. Это можно сделать командой `echo $SHELL`. Затем нужно установить утилиту arduino-cli, сделать это можно следуя официальной документации разработчика. Система готова к дальнейшему использованию.

Далее нужно расположить сам файл скрипта в желаемой директории, например $HOME. В этой же директории необходимо создать подкаталоги sketches и builds. Директорию sketches заполнить в соответствии с требуемой

иерархией каталогов и скетчей Arduino, то есть каждый скетч нужно поместить в одноименную директорию.

В скрипте реализована проверка хеш-сумм скетчей во избежание подмены файлов. Файл хеш-суммы каждого скетча пользователь создает и обновляет самостоятельно. Храниться хеш-сумма скетча должна в файле md5sum в той же директории, где и сам скетч.

2.3. Выполнение скрипта

Когда все файлы созданы и добавлены можно запускать скрипт командой:

```
./auto-upload.sh my_arduino_sketch
```

Теперь приступим к детальному разбору кода, который приведен в Приложении 1.

Определяем переменные: PATH_TO_ARDUINO_CLI необходима для того, чтобы далее добавить ее в переменную PATH для вызова из любой директории системы. Это сделано на случай, если переменная не была добавлена пользователем изначально. Переменные PATH_TO_SKETCHES и PATH_TO_BUILDS указывают на директории, в которых хранятся загружаемые скетчи и скомпилированные файлы соответственно.

Далее в блоке if-else проводится проверка на то, что пользователь передал параметр, иначе выводится ошибка. Переменная SKETCH вводится и принимает значение первого параметра для удобства чтения кода.

В следующем блоке if проверяется существование указанного скетча. При негативном исходе скрипт завершается с ошибкой. Третий блок if-else проверяет существует ли файл хеш-суммы, при положительном исходе осуществляется проверка совпадения актуальной хеш-суммы указанного скрипта и той, что указана в файле. Все не пройденные проверки сопровождаются выводом описания ошибки.

В переменной BOARD по умолчанию указана имя платы Arduino Uno.Если пользователь использует другие виды плат, их наименование можно посмотреть утилитой arduino-cli.

Команда

```
PORT_ARRAY=$(arduino-cli board list | awk 'NR>1{print $1}')
```

сохраняет в переменной PORT_ARRAY массив всех портов, к которым подключены платы. В блоке if проверяется наличие в массиве PORT_ARRAY хотя бы одного элемента, то есть хотя бы одного порта, к которому подключена плата.

Далее в директории builds создается каталог по названию скетча, для хранения в нем скомпилированных файлов.

Следующей командой проводится билд.

```
arduino-cli compile --build-path $PATH_TO_BUILDS/$SKETCH -b $BOARD $PATH_TO_SKETCHES/$SKETCH
```

В цикле for осуществляется загрузка скомпилированных файлов на плату. Сначала пользователь получает предупреждение с возможностью отмены, что на определенный порт будет произведен деплой, и при вводе в терминал параметра “y” производится загрузка файлов на плату. Цикл повторяется для всех элементов массива PORT_ARRAY.

Выполнение скрипта завершено. Можно проверять микроконтроллеры на корректную работу.

3. ЗАКЛЮЧЕНИЕ

В результате проведения научной работы были исследованы возможности и средства автоматической сборки кода для плат Arduino и загрузка готовых файлов на микроконтроллеры. Решение использовать язык оболочки bash оправдало себя, т.к. его возможностей было достаточно для работы с утилитой arduino-cli.

При необходимости можно изменить структуру репозитория кода на более удобную, также как и изменить расположение хранения файлов с хеш-суммами. Иерархия файлов и директорий, определенная в скрипте, не является строгой.

Единственным недостатком в работе является то, что не все разработчики кода под Arduino микроконтроллеры обладают базовыми навыками работы с терминалом Linux. Однако большую часть работы берет на себя разработанный CI/CD скрипт, оставляя на пользователя только создание репозитория кода и ввода команды выполнения скрипта.

Тем не менее, даже этих действий можно избежать и свести работу пользователя в терминале к нулю, введя в процесс разработки удаленный репозиторий хранения кода, инструменты CI/CD, а также отдельный настроенный на работу с Arduino платами сервер.

СПИСОК ЛИТЕРАТУРЫ

1. Mendel Cooper «Advanced Bash-Scripting Guide», Independently published, November 9, 2019
2. Jeremy Blum «Exploring Arduino», Wiley, July 12, 2013;
3. Дэниел Джей Барретт «Linux. Командная строка. Лучшие практики», пер. А.Гаврилов, Питер, 2023.
4. Дмитрий Колисниченко «Командная строка Linux», БХВ, 2023.

Приложение

Приложение 1

```
#!/bin/bash
PATH_TO_ARDUINO_CLI=$HOME/arduino
export PATH="$PATH_TO_ARDUINO_CLI:$PATH"
PATH_TO_SKETCHES=./sketches
PATH_TO_BUILDS=./builds
if [ -z "$1" ]; then
    echo -e "Pass the name of the sketch as the first
parameter.\nExample: ./auto_upload.sh my_sketch"
    exit 1
fi
SKETCH=$1
if ! [ -f $PATH_TO_SKETCHES/$SKETCH/$SKETCH.ino ]; then
    echo "Sketch '$SKETCH' does not exist."
    exit 1
fi
if [ -f $PATH_TO_SKETCHES/$SKETCH/md5sum ]; then
    if ! [[ $(md5sum
$PATH_TO_SKETCHES/$SKETCH/$SKETCH.ino|awk '{print $1}')
== \
    $(cat $PATH_TO_SKETCHES/$SKETCH/md5sum|awk '{print
$1}') ]]; then
        echo "Hash sum are not equal. Update md5sum
file in the folder where the sketch is located."
        exit 1
    fi
else
    echo "Create md5sum file in the folder where the
sketch is located."
    exit 1
fi
BOARD="arduino:avr:uno" # default
PORT_ARRAY=$(arduino-cli board list | awk 'NR>1{print
$1}')
if [ ${#PORT_ARRAY[@]} == 0 ]; then
    echo "No boards connected. No available ports."
    exit 1
fi
if ! [ -d $PATH_TO_BUILDS/$SKETCH ]; then
    mkdir $PATH_TO_BUILDS/$SKETCH
fi
```

```
arduino-cli compile --build-path
$PATH_TO_BUILDS/$SKETCH -b $BOARD
$PATH_TO_SKETCHES/$SKETCH
for port in ${PORT_ARRAY[@]}
    do
        echo "Deploying on port $port, continue? [y,n]"
        read input
        while ! [[ "$input" == "y" || "$input" == "n"
]]; do
            echo "Just input 'y' or 'n'"
            read input
        done
        if ! [ "$input" == "y" ]; then
            echo "Process interruption..."
            exit 1
        fi
        arduino-cli upload -p $port -b $BOARD --input-
dir $PATH_TO_BUILDS/$SKETCH
done
```