Device&Tools:

RPi oscilloscope

Introduce: analog-to-digital voltage converter on Raspberry Pi platform with an Ethernet interface

Components:

1. Analog-to-digital voltage converter (converter)

Hardware:

1. Raspberry Pi platform
2. AD7705 module

Features of the converter firmware:

1. Language – Java 11 SE only
2. The pi4j library (https://pi4j.com/)

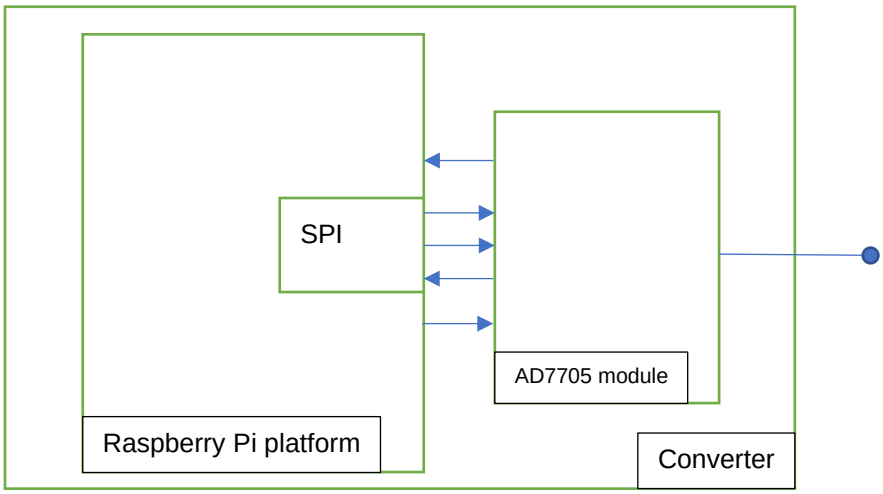
Features of the converter hardware:

1. A simplest ADC module with a low cost was used to evaluate the capabilities of this ADC module

Technical characteristics of the oscilloscope:

1. Sampling interval: 2, millisecond
2. Input voltage range: 0 ÷ 3.3, V
3. Supply voltage of the converter: 5, V
4. Generating test signals (sinus wave, triangle wave, meander wave) with variable amplitude and frequency

Hardware flowchart:



One of the possible clients:

1. Mobile client (https://www.upwork.com/o/profiles/users/~01456a027e322ac49a/?p=1437037080736825344)
2. Desktop client (https://www.upwork.com/o/profiles/users/~01456a027e322ac49a/?p=1412630003922558976)

can be used to visualize the data

Figure 1. Input signal

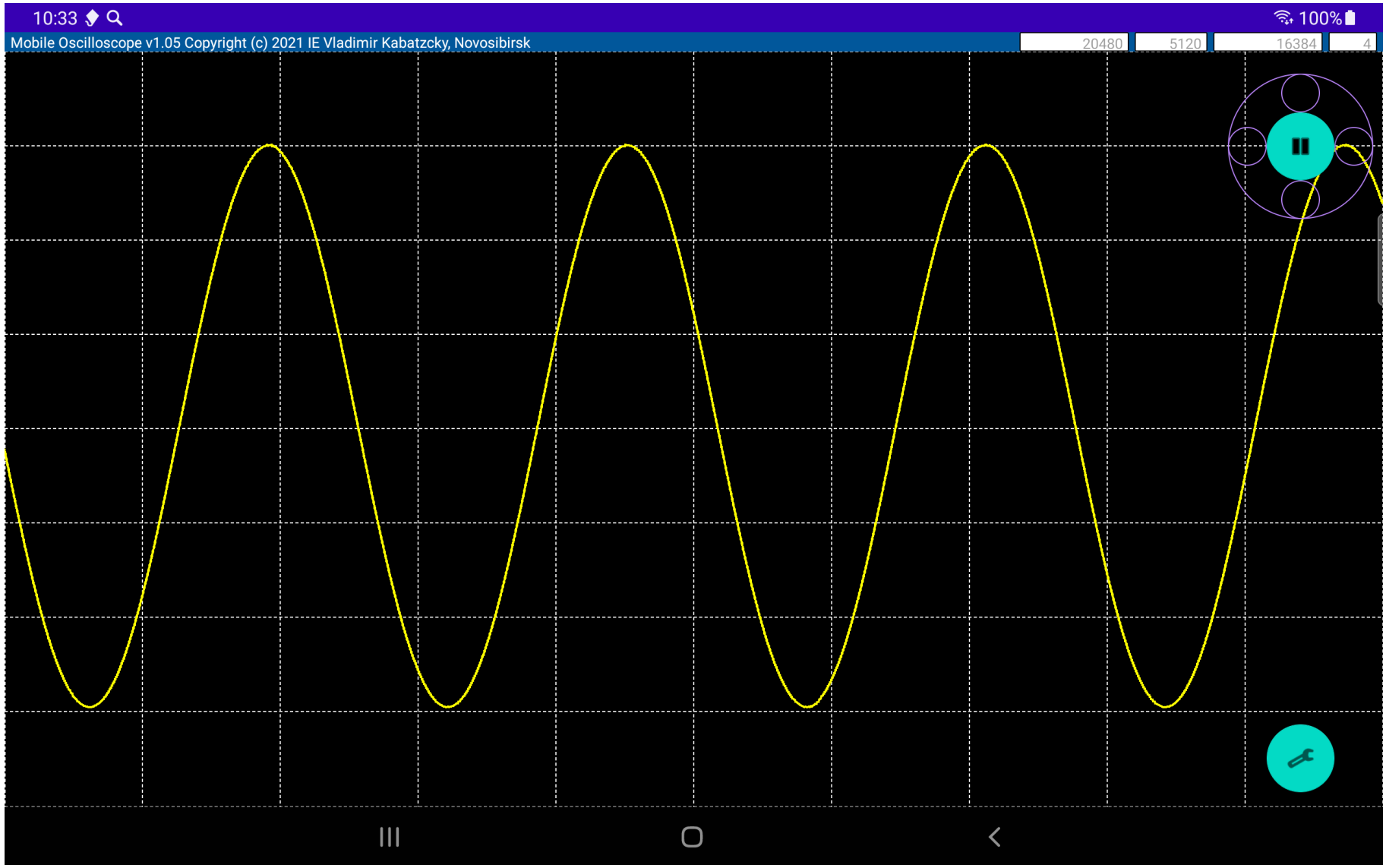


Figure 2. About

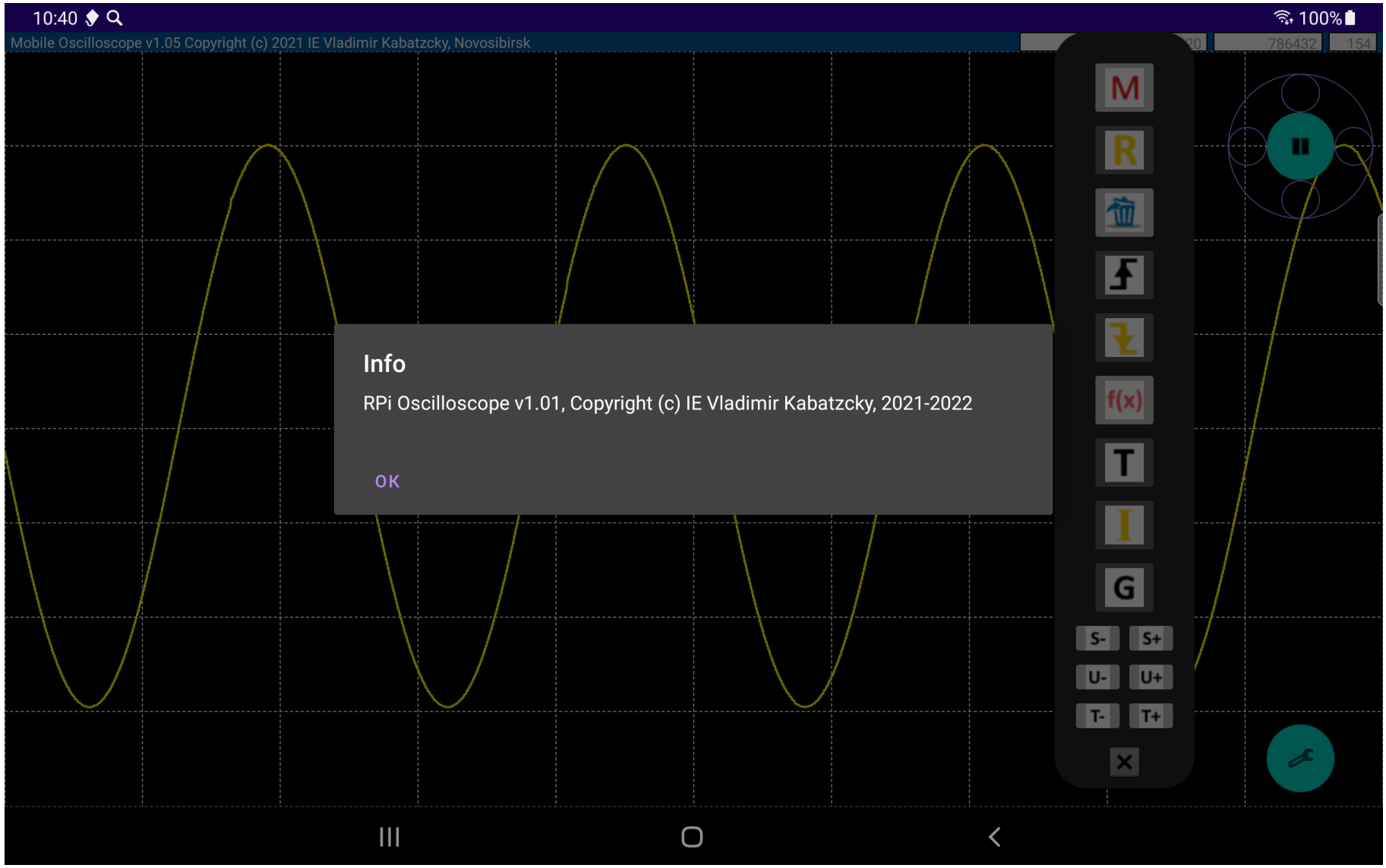


Figure 3. Signal parameters

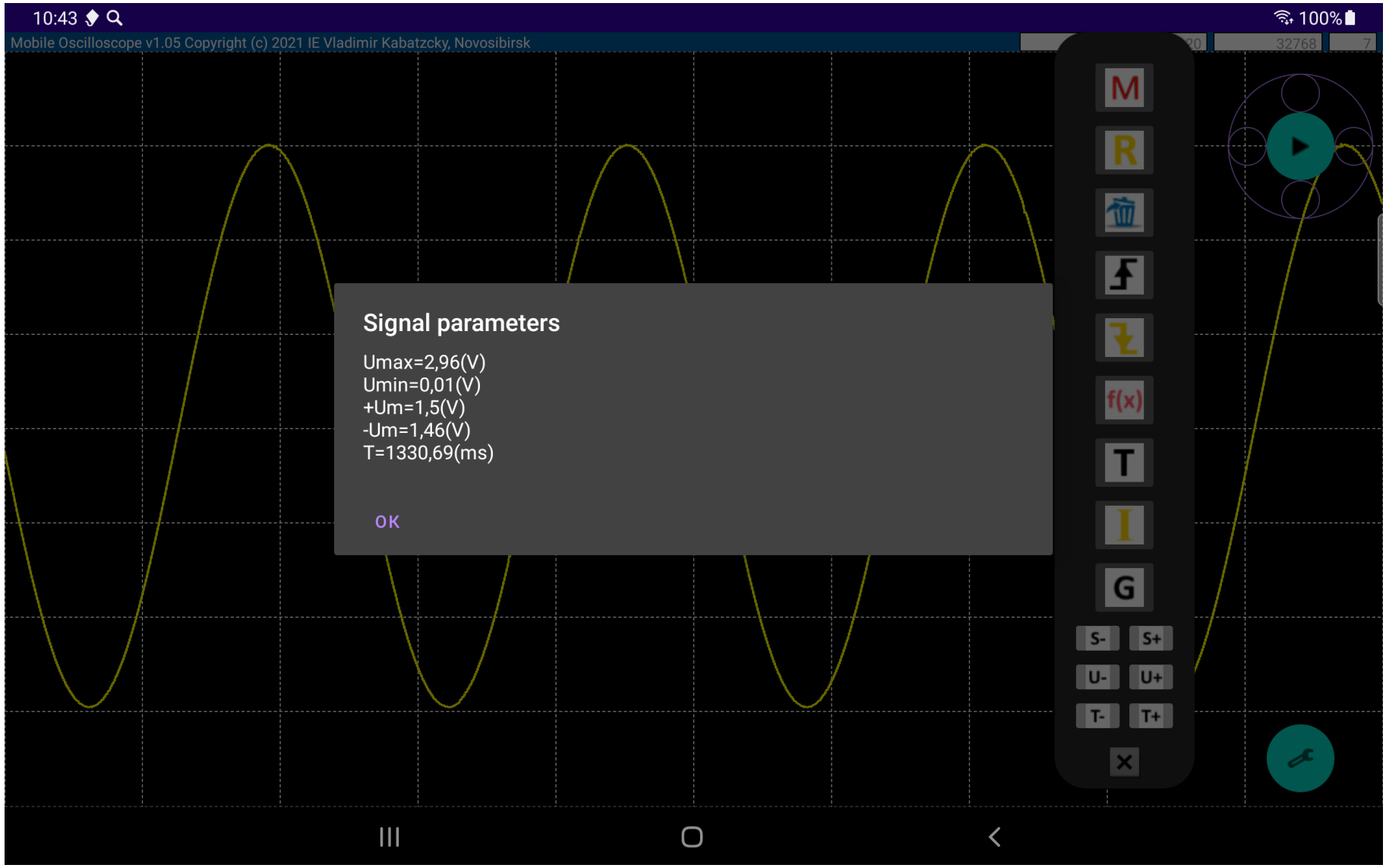


Figure 4. Code example

```
package ru.fieldgarden.fgrpi2oscilloscope;
…
/**
 *
 * @author Copyright (c) 2021 IE Vladimir Kabatzcky
 */
public class Main {
    private static final Map<String, Boolean> osParams = new HashMap();
    static {
        osParams.put("isWindows", false);
        osParams.put("isLinux", false);
        osParams.put("isHpUnix", false);
        osParams.put("isPiUnix", false);
        osParams.put("isSolaris", false);
        osParams.put("isSunOS", false);
        osParams.put("archDataModel32", false);
        osParams.put("archDataModel64", false);
        getOSParams();
    }

    private static final int DATA_SERVICE_PERIOD = 10;//Attentio!!!10(ms)

    private static DataHandlingThread mDataThread;
    private static ScheduledExecutorService mDataService;

    private static AD7705Device device;

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        if (!osParams.get("isPiUnix")) {
            System.exit(0);
            return;
        }
```

```
        device = new AD7705Device();
        device.init();

        String fname = System.getProperty("user.dir") + File.separator
                + "conn.properties";
        System.out.println(fname);
        String serverIP = "10.0.0.2";
        serverIP = getServerIP(fname, serverIP);
        System.out.println("server IP is read:" + serverIP);

        CmdHandlingThread cmdThread = new CmdHandlingThread();
        cmdThread.setServerIP(serverIP);
        cmdThread.start();

        mDataThread = new DataHandlingThread();
        mDataThread.setDevice(device);

        mDataService = Executors
                .newSingleThreadScheduledExecutor();
        mDataService.scheduleAtFixedRate(mDataThread, 0, DATA_SERVICE_PERIOD,
TimeUnit.MILLISECONDS);
        cmdThread.setDataHandlingThread(mDataThread);

        while (true) {
            try {
                int code = System.in.read();
                char ch = (char) code;
                System.out.println("ch: " + ch + ", code: " + code);
                switch (ch) {
                    case 'x':
                        cmdThread.abort();
                        while (cmdThread.isAlive()) {
                            try {
                                TimeUnit.MILLISECONDS.sleep(50);
                            } catch (InterruptedException ex) {
                                Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
                            }
```

```
                }
                if (mDataService != null) {
                    mDataService.shutdown();
                }
                device.deinit();
                System.exit(0);
                break;
            case 'r':
                cmdThread.reset();
                device.init();
                break;
        }
        try {
            TimeUnit.MILLISECONDS.sleep(100);
        } catch (InterruptedException ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        }
    } catch (IOException ex) {
        Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
    }
  }
}

private static String getServerIP(String fname, String defaultIP) {
  try (FileInputStream fis = new FileInputStream(fname)) {
    Properties prop = new Properties();
    prop.load(fis);
    return prop.getProperty("serverIP", defaultIP);
  } catch (FileNotFoundException ex) {
    System.out.println(ex.getMessage() + "!");
    return defaultIP;
  } catch (IOException ex) {
    System.out.println(ex.getMessage() + "!");
    return defaultIP;
  }
}
```

```
private static void getOSParams() {
    final String os = System.getProperty("os.name").toLowerCase();
    System.out.println("os.name:" + os);
    if (os.contains("windows")) {
        osParams.replace("isWindows", Boolean.TRUE);
    }
    if (os.contains("linux")) {
        osParams.replace("isLinux", Boolean.TRUE);
    }
    if (os.contains("hp-ux")) {
        osParams.replace("isHpUnix", Boolean.TRUE);
    }
    if (os.contains("hpux")) {
        osParams.replace("isHpUnix", Boolean.TRUE);
    }

    if (os.contains("solaris")) {
        osParams.replace("isSolaris", Boolean.TRUE);
    }
    if (os.contains("sunos")) {
        osParams.replace("isSunOS", Boolean.TRUE);
    }
    final String model = System.getProperty("sun.arch.data.model");
    System.out.println("sun.arch.data.model:" + model);
    if (model.equals("32")) {
        osParams.replace("archDataModel32", Boolean.TRUE);
    }
    if (model.equals("64")) {
        osParams.replace("archDataModel64", Boolean.TRUE);
    }
    if (osParams.get("isLinux")) {
        final File file = new File("/etc", "os-release");
        try (FileInputStream fis = new FileInputStream(file);
                BufferedReader br = new BufferedReader(new InputStreamReader(fis))) {
            String string;
```

```
            while ((string = br.readLine()) != null) {
                System.out.println("string:" + string);
                if (string.toLowerCase().contains("raspbian")) {
                    if (string.toLowerCase().contains("name")) {
                        osParams.replace("isPiUnix", Boolean.TRUE);
                        break;
                    }
                }
            }
        } catch (Exception ex) {
            Logger.getLogger(Main.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
    osParams.forEach((k, v) -> System.out.println(k + ":" + v));
  }
}
```

Thanks for attention!