

## Contents

1. Objectives .....	2
2. Hardware.....	2
3. Software and Toolchain .....	2
4. Prepare Linux environment for VEGAbord SDK.....	3
4.1 Download SDK and Toolchain .....	3
4.2 Extract SDK.....	3
4.3 Extract toolchain.....	3
4.4 Set environment variables for VEGAbord SDK and Toolchain .....	3
4.5 Download and extract GNU-MCU Eclipse IDE .....	3
4.6 Configure toolchain and OpenOCD in Eclipse.....	4
4.7 J-Link Software for Linux.....	4
5. Connect your board and debugger to computer .....	4
6. Build & Flash using Terminal .....	5
7. Verify serial output.....	5
8. Build & Flash from Eclipse.....	6
8.1 Open eclipse.....	6
8.2 Import an existing project. ie, the hello_world path:.....	6
8.3 Build the project.....	6
8.4 Flash the application using Eclipse.....	6
9. Bluetooth Low Energy demo application .....	6
9.1 Download NXP's IoT Toolbox.....	7
9.2 Build and Flash the Heart Rate Sensor application .....	7
9.3 Run the HRS application .....	7
9.4 Modify Advertising Name .....	8

## 1. Objectives

- Get software and tools
- Configure environment for VEGAboard SDK
- Build and Flash demo applications using GCC commands
- Build and Flash demo applications using Eclipse
- Build and Flash Bluetooth LE application

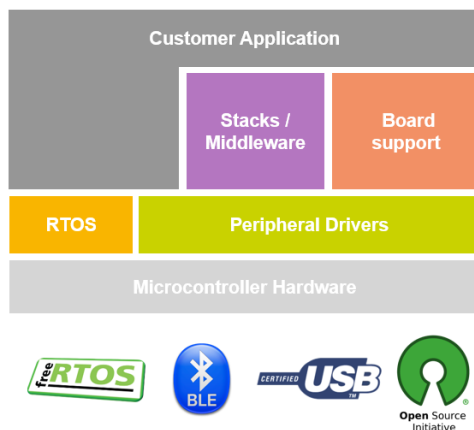
## 2. Hardware

- VEGAboard (rv32m1\_vega)
- Mobile (Android or iOS)
- 2 Micro-USB cables
- Debug probe (Guide uses J-Link EDU mini)
- Internet access

## 3. Software and Toolchain

Software and toolchain available at <https://open-isa.org/downloads>

- Linux/Mac SDK:
  - o rv32m1\_sdk\_riscv\_installer.sh
- Toolchain (Prebuilt GCC and OpenOCD)
  - o Toolchain\_Linux.tar.gz
- GNU MCU Eclipse IDE - <https://github.com/gnu-mcu-eclipse>
- Segger J-Link software for Linux - <https://www.segger.com>



## 4. Prepare Linux environment for VEGAboard SDK

The instructions from this guide are executed on a Linux Terminal (Ubuntu). Refer to <https://open-isa.org> for Windows environment instructions.

Install some required Linux software packages:

```
sudo apt install curl cmake openjdk-11-jre
```

Note: You may need to update your software database ('apt-get update').

### 4.1 Download SDK and Toolchain

Download software packages from open-isa.org github

```
curl -L https://github.com/open-isa-org/open-isa.org/releases/download/1.0.0/rv32m1_sdk_riscv_installer.sh > \
$HOME/rv32m1_sdk_riscv_installer.sh

curl -L https://github.com/open-isa-org/open-isa.org/releases/download/1.0.0/Toolchain_Linux.tar.gz > \
$HOME/Toolchain_Linux.tar.gz
```

### 4.2 Extract SDK

Extract and install SDK.

```
cd $HOME
chmod +x rv32m1_sdk_riscv_installer.sh
./rv32m1_sdk_riscv_installer.sh
```

### Accept license ###

After the license was accepted, create a /vega folder and extract them there

```
mkdir vega && cd vega
tar xf ../rv32m1_sdk_riscv.tar.gz
```

### 4.3 Extract toolchain

Extract toolchain compressed file into the /vega folder

```
cd $HOME/vega
mkdir toolchain && cd toolchain
tar xf ../../Toolchain_Linux.tar.gz
tar xf riscv32-unknown-elf-gcc.tar.gz
rm riscv32-unknown-elf-gcc.tar.gz
tar xf openocd.tar.gz
rm openocd.tar.gz
```

### 4.4 Set environment variables for VEGAboard SDK and Toolchain

```
export RV32M1_SDK_DIR=$HOME/vega/rv32m1_sdk_riscv
export PATH=$PATH:$HOME/vega/toolchain
export RISCV32GCC_DIR=$HOME/vega/toolchain/riscv32-unknown-elf-gcc
export PATH=$PATH:$RISCV32GCC_DIR/bin
```

### 4.5 Download and extract GNU-MCU Eclipse IDE

```
cd $HOME
curl -L https://github.com/gnu-mcu-eclipse/org.eclipse.epp.packages/releases/download/v4.6.1-20190923-2019\
-09/20190923-1700-gnumcueclipse-4.6.1-2019-09-R-linux.gtk.x86_64.tar.gz > gnumcueclipse-4.6.1_x86_64.tar.gz
tar xf gnumcueclipse-4.6.1_x86_64.tar.gz
rm gnumcueclipse-4.6.1_x86_64.tar.gz
```

## 4.6 Configure toolchain and OpenOCD in Eclipse

```
mkdir -p $HOME/eclipse/configuration/.settings/  
  
echo "eclipse.preferences.version=1" > \  
$HOME/eclipse/configuration/.settings/ilg.gnuclipse.debug.gdbjtag.openocd.prefs  
  
echo "install.folder=$HOME/vega/toolchain" >> \  
$HOME/eclipse/configuration/.settings/ilg.gnuclipse.debug.gdbjtag.openocd.prefs  
  
echo "eclipse.preferences.version=1" > \  
$HOME/eclipse/configuration/.settings/ilg.gnuclipse.managedbuild.cross.riscv.prefs  
  
echo "toolchain.path.512258282=$HOME/vega/toolchain/riscv32-unknown-elf-gcc/bin" >> \  
$HOME/eclipse/configuration/.settings/ilg.gnuclipse.managedbuild.cross.riscv.prefs
```

## 4.7 J-Link Software for Linux

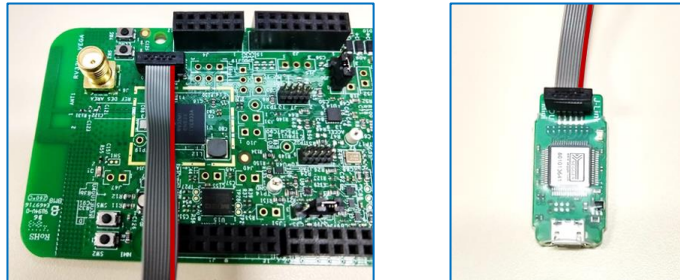
Go to <https://www.segger.com/downloads/jlink#J-LinkSoftwareAndDocumentationPack>  
Download the latest J-Link Software and Documentation pack for your system (DEB installer in this case)

Install the J-Link pack (assuming it was downloaded to ~/Downloads folder):

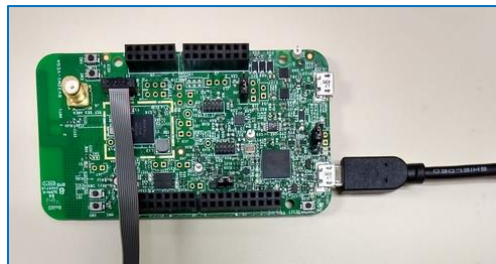
```
sudo dpkg -i $HOME/Downloads/JLink_Linux_Vxxx_x86_64.deb
```

## 5. Connect your board and debugger to computer

Connect the J-Link debug probe to J55 header from VEGAboard as shown below.

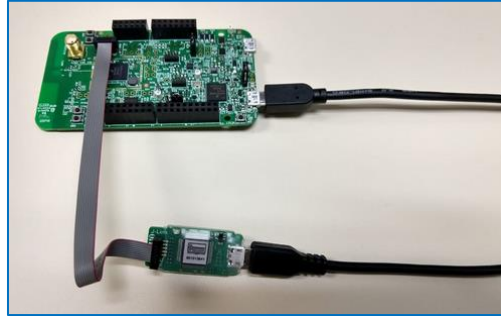


Connect an USB cable to J12 port from VEGAboard and then to computer as shown below.



Connect the J-Link debug probe to the computer.

This is how the setup should look like, both USB cables connected to the PC.



## 6. Build & Flash using Terminal

Go to the demo application folder (ie. hello\_world)

```
cd $RV32M1_SDK_DIR/boards/rv32m1_vega/demo_apps/hello_world/ri5cy/riscvgcc
```

Build the application using 'build\_debug' script

```
./build_debug.sh
```

Flash the application using OpenOCD + GDB

```
openocd -f $HOME/vega/rv32m1_sdk_riscv/boards/rv32m1_vega/rv32m1_ri5cy.cfg
```

Open another terminal session (don't forget to configure the env variables from Step 4.4) or Press Ctrl+Z and type 'bg' in the terminal to keep openocd running in background.

```
cd $RV32M1_SDK_DIR/boards/rv32m1_vega/demo_apps/hello_world/ri5cy/riscvgcc/debug  
riscv32-unknown-elf-gdb hello_world.elf
```

While on **(gdb)** application:

```
target remote localhost:3333  
load  
monitor reset  
quit
```

## 7. Verify serial output

Open a Serial Terminal to verify output. This guide will use "Screen" application.

Terminal settings: Baud-rate: 115200, Data: 8bits, Parity: None, Flow Control: None.

Note: Locate your interface under: /dev/**ttyxxx**:

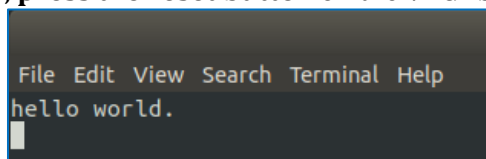
```
ls /dev/tty*
```

#Make sure you select the port corresponding to your setup, in this case: /dev/**ttyACM0**

```
sudo apt install screen
```

```
sudo screen /dev/ttyACM0 115200
```

After the terminal is ready, **press the reset button** on the VEGAboard.



To end the session in screen, press '**Ctrl+a**', type '**:quit**' and press '**Enter**'.

## 8. Build & Flash from Eclipse


Make sure you already configured toolchain and openOCD described in step 4.6.

### 8.1 Open eclipse


```
cd $HOME/eclipse
./eclipse
```

Select any workspace.


### 8.2 Import an existing project. ie, the hello\_world path:

1. Click on File -> Import 
2. Select General -> Existing projects into workspace
3. Click on Browse and navigate to the hello\_world demo folder in this path:  
\$HOME/vega/rv32m1\_sdk\_riscv/boards/rv32m1\_vega/demo\_apps/hello\_world/ri5cy/riscveclipse
4. Make sure "hello\_world\_ri5cy\_rv32m1\_vega" project is selected in the Projects section
5. Click 'OK' and 'Finish'

### 8.3 Build the project

Click the "Hammer" to build your application 

### 8.4 Flash the application using Eclipse

1. Click on Run -> Debug Configurations
2. Click on "GDB OpenOCD Debugging"
3. Select "hello\_world\_ri5cy\_rv32m1\_vega debug openocd"
4. Click on "Debug"
5. Click on "Resume" to debug the application or "Terminate" to close the debug session  
and run the application 
6. Verify serial output. See section 7 for instructions to setup screen application

## 9. Bluetooth Low Energy demo application

The RV32M1-VEGA platform is a dual-core; it contains two RISC-V cores: RI5CY and ZERO-RISCY. The Bluetooth LE demo applications use both cores; hence each demo application has two linked projects: one for the RI5CY core (in the 'ri5cy' folder) and one for the ZERO-RISCY core (in the 'zero\_riscy' folder). For the Bluetooth LE application to work, both projects need to be loaded. The RI5CY core will act as host and the ZERO\_RISCY will act as controller.

By now, you know how to build and flash applications to your VEGAboard. Follow these steps to run a Heart Rate Sensor application using the VEGAboard and the IoT Toolbox mobile application.

### IMPORTANT NOTE

The Bluetooth LE Controller library is not included in the rv32m1\_SDK package. If you would like to use Bluetooth LE, please send an email to [support@open-isa.org](mailto:support@open-isa.org) to request the library (lib\_ble\_rv32m1\_controller\_cm0p.a) and place it in the following location:  
rv32m1\_sdk\_riscv/middleware/wireless/bluetooth\_1.4.0/controller/lib/

### 9.1 Download NXP's IoT Toolbox

Download NXP's IoT Toolbox application to your smartphone  
The application is available for Android and iOS.

### 9.2 Build and Flash the Heart Rate Sensor application

Build and flash the Bluetooth LE **Heart Rate Sensor** application to your VEGAboard.

The projects are located at:

rv32m1\_sdk\_riscv/boards/rv32m1\_vega/wireless\_examples/bluetooth/heart\_rate\_sensor/freertos/**ri5cy**  
and

rv32m1\_sdk\_riscv/boards/rv32m1\_vega/wireless\_examples/bluetooth/heart\_rate\_sensor/freertos/**zero\_riscy**

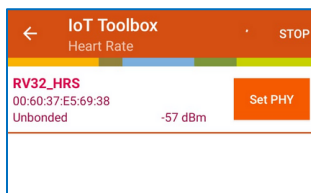
Build and flash **both** applications to your VEGAboard. You can either use GCC from terminal or Eclipse IDE.

### 9.3 Run the HRS application

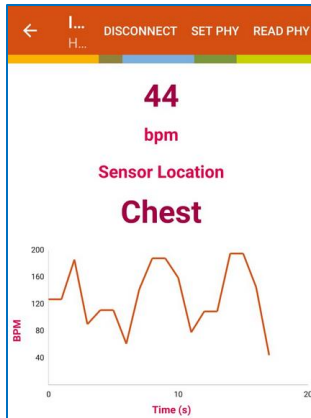
1. Open the IoT Toolbox application and select the Heart Rate app



2. Press the 'Reset' button on your VEGAboard
3. Verify your device is being advertised and displayed in the application



4. Select the **RV32\_HRS** device to start a connection
5. Once connected, verify the data is displayed on the IoT Toolbox



## 9.4 Modify Advertising Name

You may see many advertising devices from different boards around you, let's change the ADV NAME of your device to make sure you are connecting to your board.

1. Select the heart\_rate\_sensor RI5CY project
2. Open the file "wireless\_examples\bluetooth\heart\_rate\_sensor\freertos\app\_config.c" in the heart\_rate\_sensor demo.
3. Locate the structure around line 75. You will find the Advertising name '.aData', modify this to identify your board.

Note: the length cannot be larger than 14 including the ending character (\0).

4. Don't forget to adapt the '.length' variable.
5. Compile the RI5CY application and download it to your board. You don't need to download the controller side (ZERO\_RISCY) application.
6. Run steps from 8.4 and verify the advertising name should be the one you configured.

```

app_config.c
70  .aData = (uint8_t *)adData1
71  },
72  {
73      .adType = gAdShortenedLocalName_c,
74      // .length = 9,
75      // .aData = (uint8_t*)"RV32 HRS"
76      .length = 14,
77      .aData = (uint8_t*)"RV32_MyADV123" //Max length: 14 characters including end character '\0'
78  }
79  };
80

```

