Raspberry Pi Debugger/Flasher DIY Toolkit Guide

(v1.2)

Last Updated: February 27, 2017

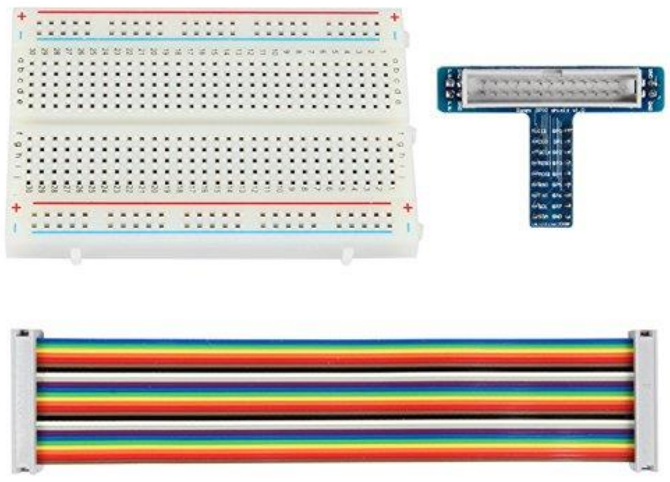
By: Spencer Manzon

What you need:

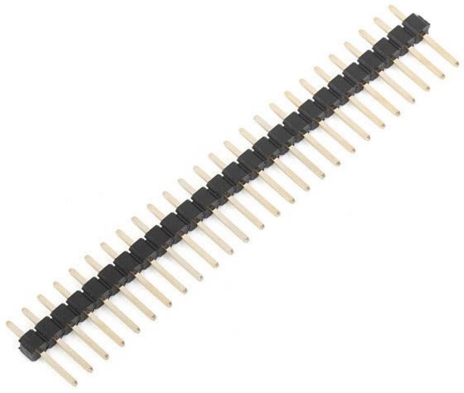
QTY	Name	EST. Price (Shipping Not Included, as of March 2016)
1	SEGGER J-Link EDU Debug Probe	$80 CAD
1	USB to TTL Serial Cable (3.3V Signal Line, 5V Power)	$5 CAD
1	Raspberry Pi Model B Breakout Board	$10 CAD
1	26-pin Ribbon Cable (0.1” pitch)	$5 CAD
1	Breadboard (Note: 400 pins is recommended)	$5 CAD
20	Pin Header	$2 CAD
> 30	Breadboard Pins (Male-to-Male)	$4 CAD
1	SNES Controller Port	$8 CAD

Above is the table of materials that are essential to build yourself a Pi HomeLab. Some extra materials are recommended as you build your Pi HomeLab. This is explained later in the guide.

USB to TTL Serial Cable (3.3V Signal Line, 5V Power)



Raspberry Pi Model B Breakout Board, Breadboard and 26-pin Ribbon Cable (0.1” pitch)



Pin Headers (Above) and Breadboard Pins (Below)

SEGGER J-Link EDU Debug Probe

SNES Controller Port

Hot glue, electrical tape, double sided tape and scrap wood or plywood are **recommended** for when you want to secure the setup to something solid and for when to insulate the wires from each other. Some people even used a pencil case or even a food container that you can get in the dollar store as an enclosure for the project. Be creative to make your Pi HomeLab your own!

Basic soldering skills and a soldering iron is required too for when you will make the custom connections for the various interconnects. Here are some soldering tutorials you can view:

http://www.howtogeek.com/63630/how-to-use-a-soldering-iron-a-beginners-guide/

https://learn.adafruit.com/adafruit-guide-excellent-soldering/tools

http://www.instructables.com/id/Soldering-wires-together/

Links to the essential parts:

http://www.amazon.ca/gp/product/B00GMEHA42?psc=1&redirect=true&ref_=ox_sc_act_title_1&smid=A1NGGED121BJA2

https://www.buyapi.ca/product/pl2303hx-usb-to-rs232-ttl-converter-cable-module/

http://engenuics.com/product/segger-j-link-edu/

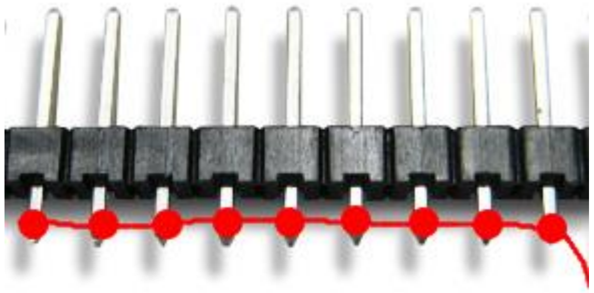
http://www.raphnet-tech.com/products/snes_controller_connector/index.php

WHAT TO DO:

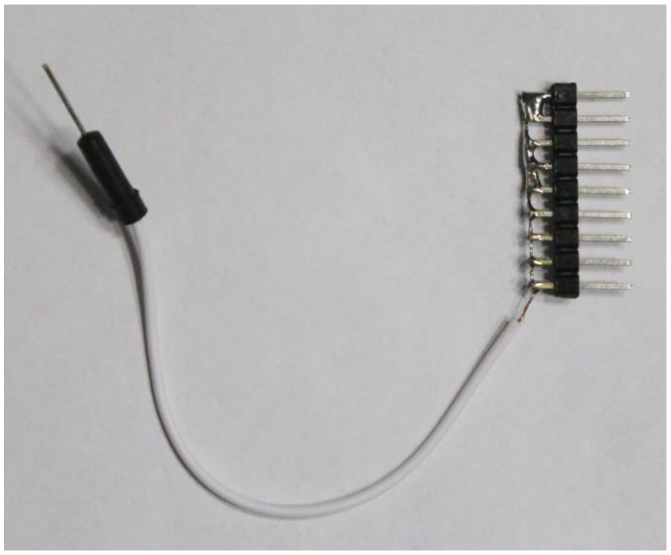
A. Prep Process

Make the required custom connectors for the J-Link unit.

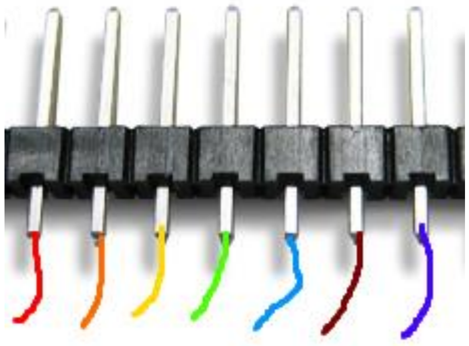
1. Break 9 pins from the pin header row. Solder all 9 pins together using the breadboard wires.



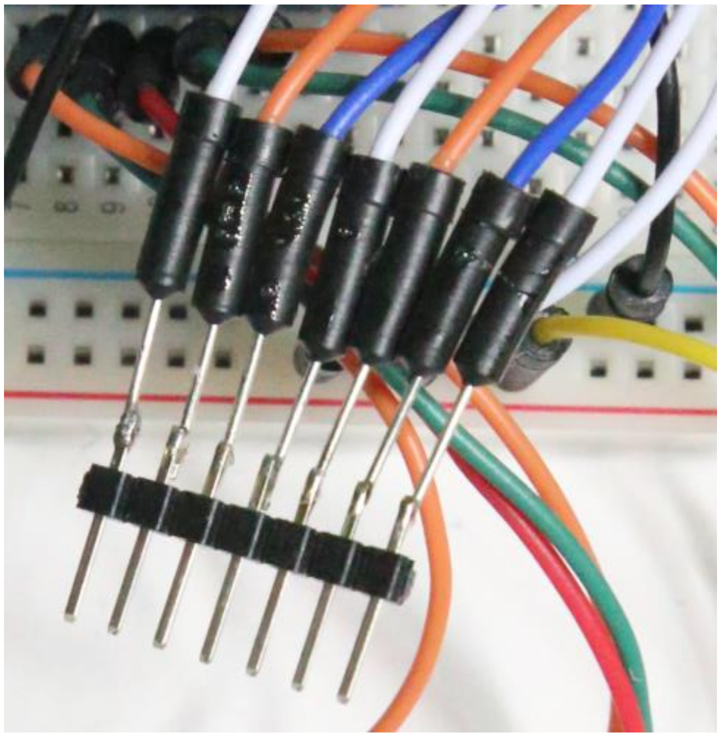
   It should look like this:



2. Break 7 pins from the pin header row. Solder a breadboard wire to each pin.



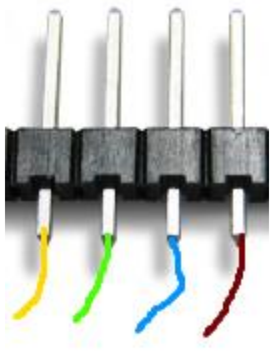
It should look like this:



Note:
NO, the wire color does not matter.

Make the custom connector for the USB to Serial connector.

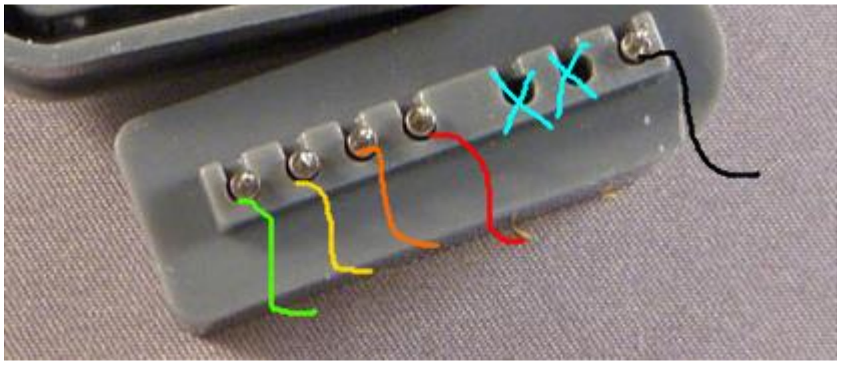
1. Break 4 pins from the pin header row. Solder a breadboard wire to each pin.



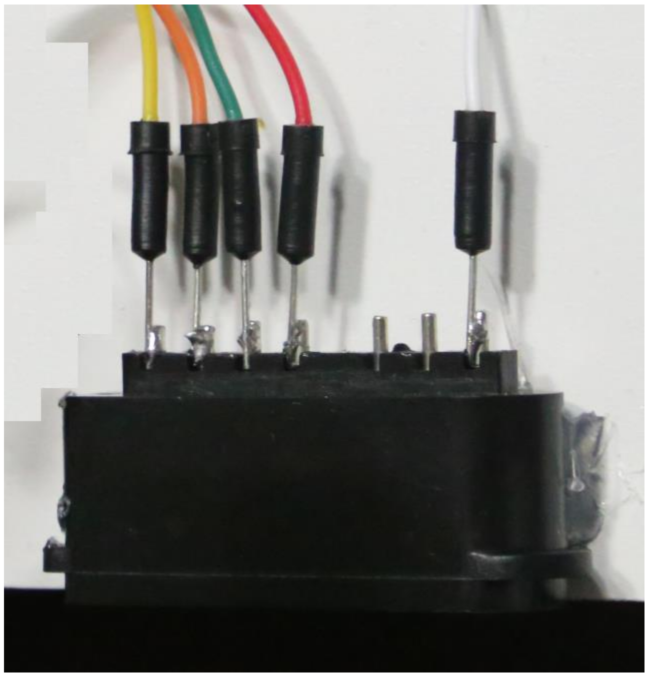
It should look similar to the ones above but only 4 pins.

Make the custom connector for the SNES connector.

1. Solder 5 breadboard wires to the designated pins:



It should look like this:



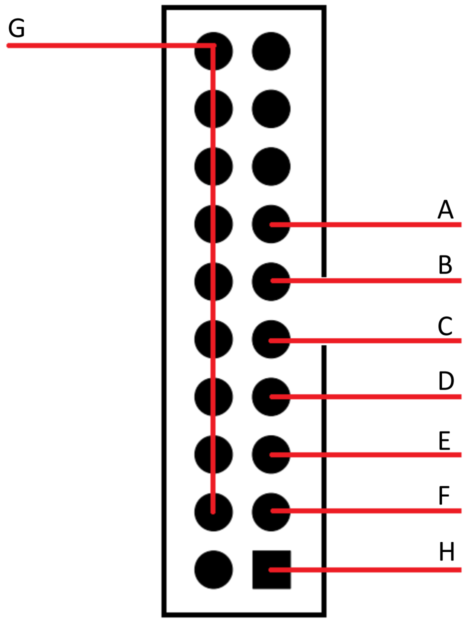
Note:
NO, the wire color does not matter.

Layout the parts in a piece of plywood. Make sure that the J-Link Unit ribbon cable, the USB to Serial connector and the SNES connector are able to connect to the breadboard. Secure the components using tape or hot glue. Be creative in this step!

One other note: Be mindful of what the wires are being in contact with! Insulate wires as needed.

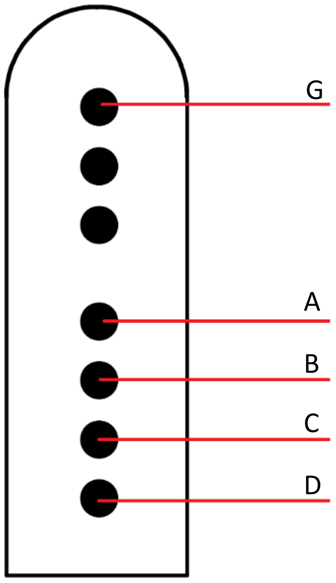
B. Wiring the whole thing

Wiring the J-Link Unit



	Official Pin Number with Alternate Names	Pin in the given Breakout Board
A	Pin 18 (BCM 24, ARM_TDO)	GP5
B	Pin 16 (BCM 23, ARM_RTCK)	GP4
C	Pin 22 (BCM 25, ARM_TCK)	GP6
D	Pin 13 (BCM 27, ARM_TMS)	GP2
E	Pin 7 (BCM 4, ARM_TDI)	GP7
F	Pin 15 (BCM 22, ARM_TRST)	GP3
G	Ground (GND)	Ground (GND)
H	Pin 17 (3.3V)	3.3V

Wiring the SNES Connector



	Official Pin Number with Alternate Names	Pin in the given Breakout Board
A	Pin 19 (BCM 10, SPIO_MOSI)	MOSI
B	Pin 21 (BCM 9, SPIO_MISO)	MISO
C	Pin 23 (BCM 11, SPIO_SCLK)	SCLK
D	Pin 17 (3.3V)	3.3V
G	Ground (GND)	Ground (GND)

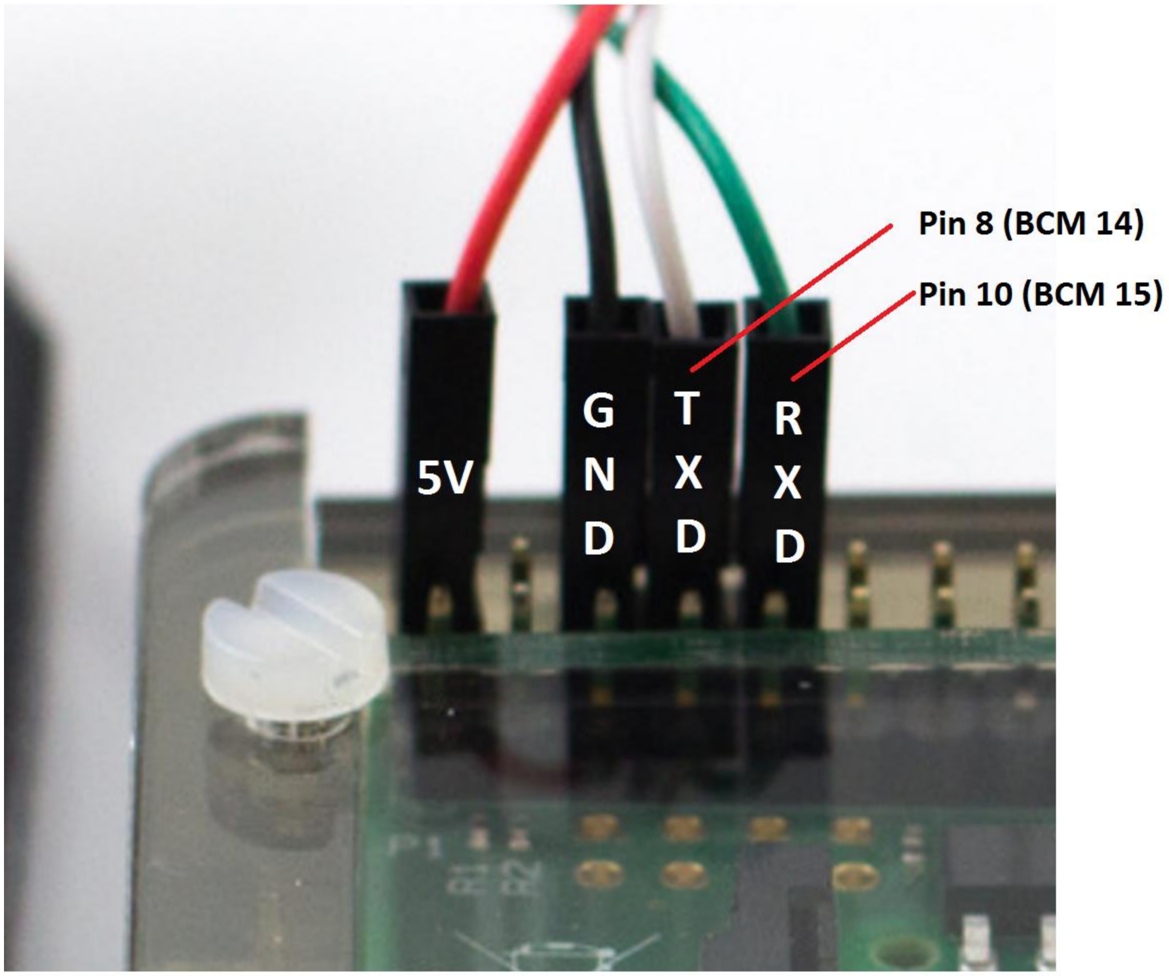
Wiring the UART Cable

For Raspberry Pi 1:

WARNING: This is where the 5V power to the Pi is coming from! Wire carefully.

For Raspberry Pi 2:

Do not connect the 5V wire. It would not be enough to power the Raspberry Pi 2. The Raspberry Pi 2 needs more power than what the 5V power from the UART USB can supply. Use an external power adapter to power the Raspberry Pi 2.



C. Software Installation

Do not plug anything yet. Restart the computer after the installation.

On Fedora 20

Download SEGGER J-Link Software and documentation pack for Linux and install. Any version that is available should be okay to use.

Execute the following commands in the terminal:

```
sudo yum install arm-none-eabi-gcc-cs.i686
sudo yum install arm-none-eabi-gdb.i686
sudo yum install screen
```

Note: These installs the required packages for x86. For x64, change “i686” to “x86_64”.

On Linux Mint

Download SEGGER J-Link Software and documentation pack for Linux and install. Any version that is available should be okay to use.

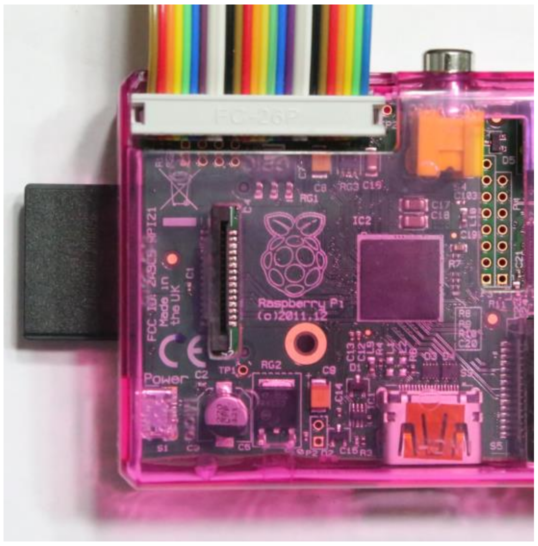
Install screen, i.e. execute: `sudo apt-get install screen`

Install the GNU ARM Embedded Toolchain (follow instructions):
https://launchpad.net/gcc-arm-embedded/

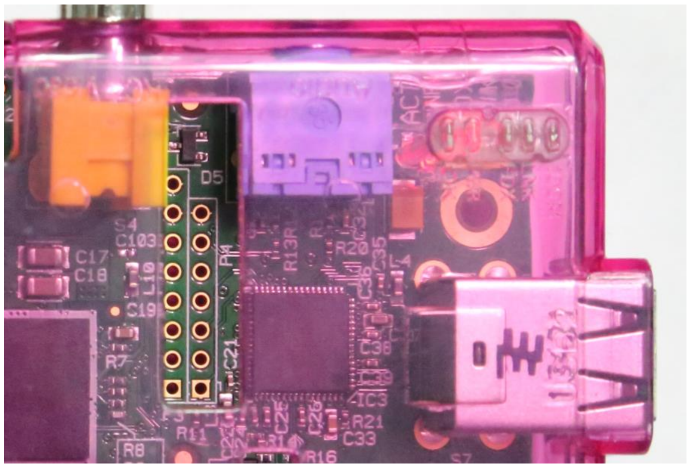
D. Testing all the things

Note: Tested in Linux Mint 13 with SEGGER J-Link V4.86a. The J-Link commands may be different if using different J-Link version. In Fedora, all of the commands except `arm-none-eabi-gdb` needs to be `sudo`.

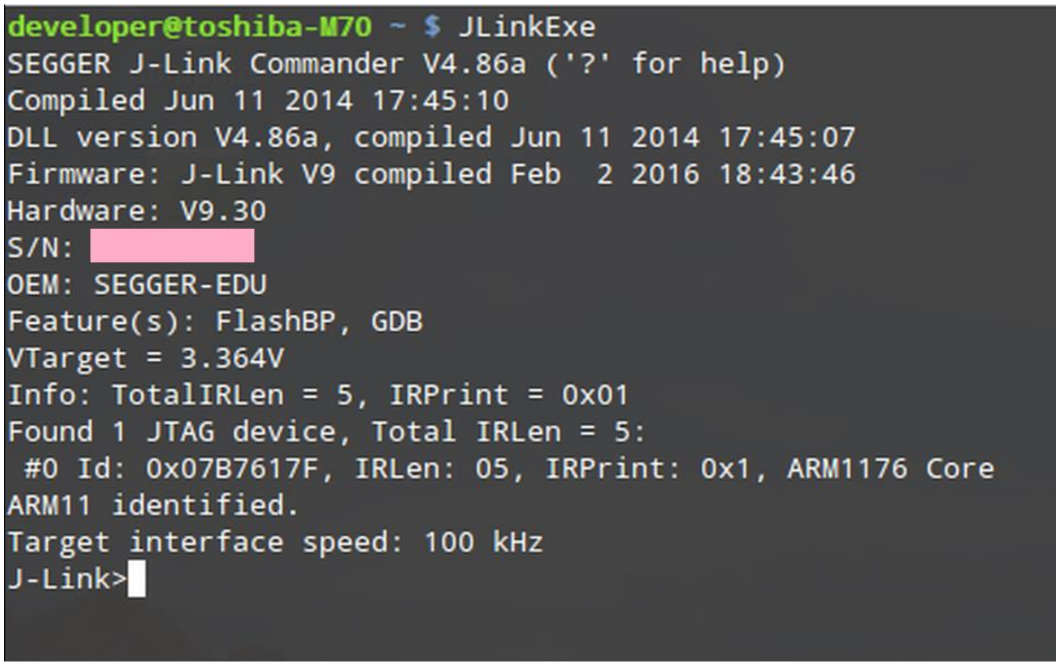
Plug the Raspberry Pi to the GPIO cable. <u>**DO NOT**</u> plug anything else especially the micro USB cable.



Check to see if the Raspberry Pi is powered on when plugging in the UART USB cable.
Note that the ACT LED will blink once when the unit is plugged in.



Plug in the J-Link to USB port. Check to see if the J-Link unit is detected.



```
developer@toshiba-M70 ~ $ JLinkExe
SEGGER J-Link Commander V4.86a ('?' for help)
Compiled Jun 11 2014 17:45:10
DLL version V4.86a, compiled Jun 11 2014 17:45:07
Firmware: J-Link V9 compiled Feb  2 2016 18:43:46
Hardware: V9.30
S/N: [redacted]
OEM: SEGGER-EDU
Feature(s): FlashBP, GDB
VTarget = 3.364V
Info: TotalIRLen = 5, IRPrint = 0x01
Found 1 JTAG device, Total IRLen = 5:
 #0 Id: 0x07B7617F, IRLen: 05, IRPrint: 0x1, ARM1176 Core
ARM11 identified.
Target interface speed: 100 kHz
J-Link>
```

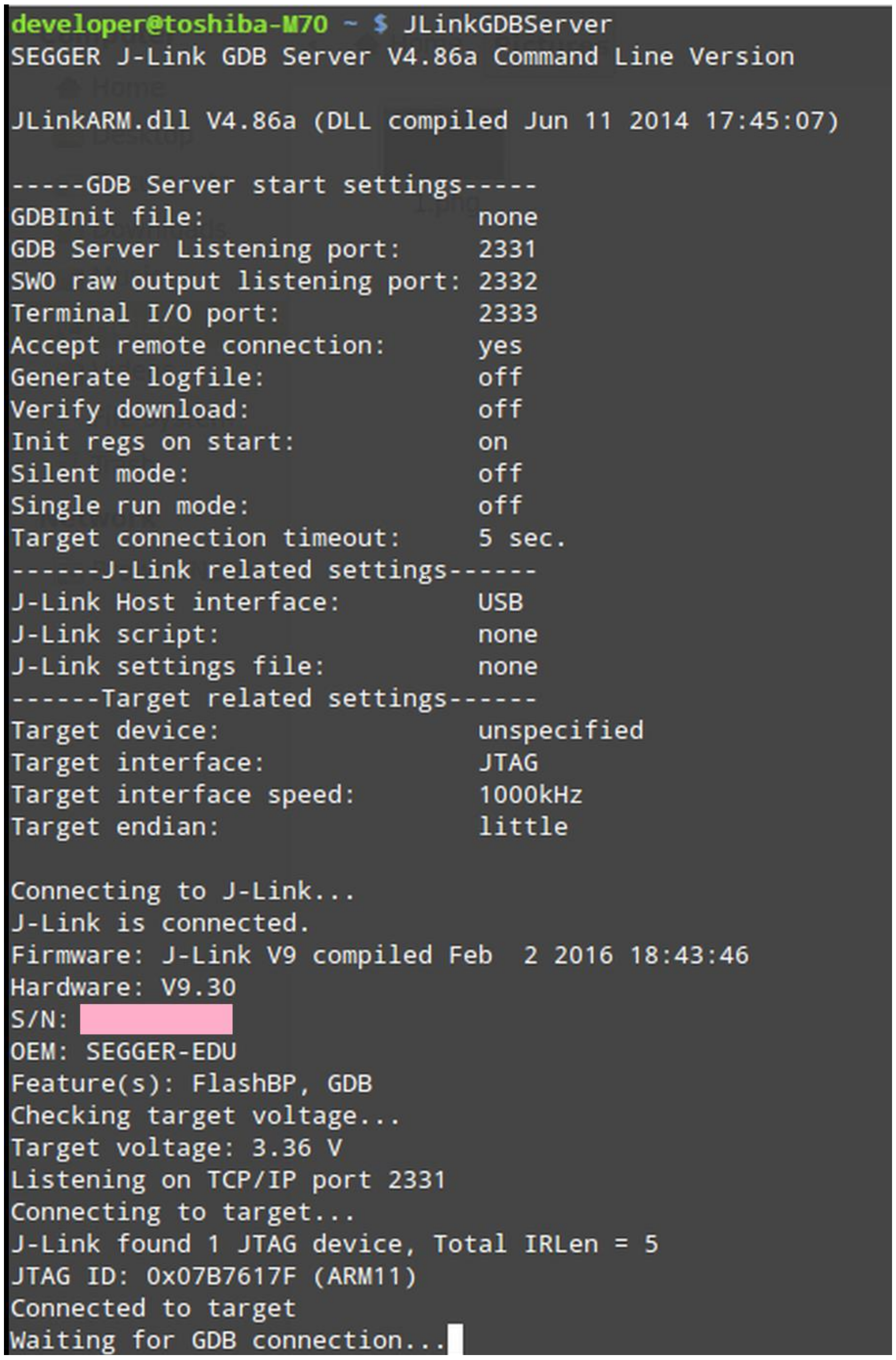
Note: For SEGGER J-Link V6.00 and above, the command would be:

```
JLinkExe -device <RaspberryPiVer>
```

where <RaspberryPiVer> would be:

ARM11	when using a Raspberry Pi 1
Cortex-A7	when using a Raspberry Pi 2

Check to see if the J-Link GDB Server can be connected.



```
developer@toshiba-M70 ~ $ JLinkGDBServer
SEGGER J-Link GDB Server V4.86a Command Line Version

JLinkARM.dll V4.86a (DLL compiled Jun 11 2014 17:45:07)

-----GDB Server start settings-----
GDBInit file:                  none
GDB Server Listening port:     2331
SWO raw output listening port: 2332
Terminal I/O port:             2333
Accept remote connection:      yes
Generate logfile:              off
Verify download:               off
Init regs on start:            on
Silent mode:                   off
Single run mode:               off
Target connection timeout:     5 sec.
------J-Link related settings------
J-Link Host interface:         USB
J-Link script:                 none
J-Link settings file:          none
------Target related settings------
Target device:                 unspecified
Target interface:              JTAG
Target interface speed:        1000kHz
Target endian:                 little

Connecting to J-Link...
J-Link is connected.
Firmware: J-Link V9 compiled Feb  2 2016 18:43:46
Hardware: V9.30
S/N: [redacted]
OEM: SEGGER-EDU
Feature(s): FlashBP, GDB
Checking target voltage...
Target voltage: 3.36 V
Listening on TCP/IP port 2331
Connecting to target...
J-Link found 1 JTAG device, Total IRLen = 5
JTAG ID: 0x07B7617F (ARM11)
Connected to target
Waiting for GDB connection...
```

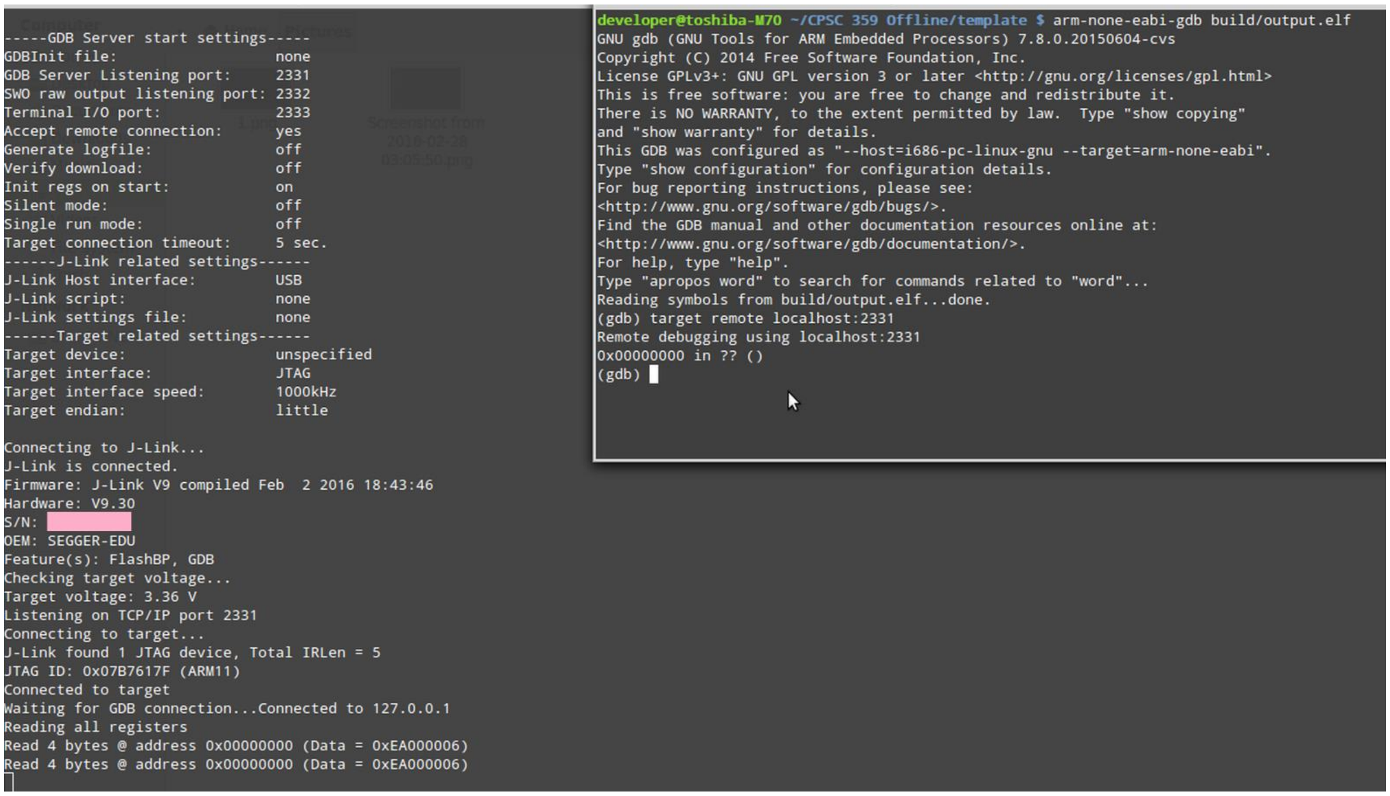
Note: For SEGGER J-Link V6.00 and above, the command would be:

```
JLinkGDBServer -device <RaspberryPiVer>
```

where <RaspberryPiVer> would be:

ARM11	when using a Raspberry Pi 1
Cortex-A7	when using a Raspberry Pi 2

Check to see if arm-none-eabi-gdb connects.



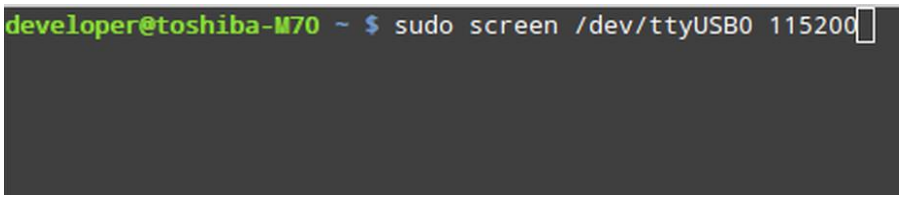
```
-----GDB Server start settings-----
GDBInit file:                  none
GDB Server Listening port:     2331
SWO raw output listening port: 2332
Terminal I/O port:             2333
Accept remote connection:      yes
Generate logfile:              off
Verify download:               off
Init regs on start:            on
Silent mode:                   off
Single run mode:               off
Target connection timeout:     5 sec.
------J-Link related settings------
J-Link Host interface:         USB
J-Link script:                 none
J-Link settings file:          none
------Target related settings------
Target device:                 unspecified
Target interface:              JTAG
Target interface speed:        1000kHz
Target endian:                 little

Connecting to J-Link...
J-Link is connected.
Firmware: J-Link V9 compiled Feb  2 2016 18:43:46
Hardware: V9.30
S/N: [redacted]
OEM: SEGGER-EDU
Feature(s): FlashBP, GDB
Checking target voltage...
Target voltage: 3.36 V
Listening on TCP/IP port 2331
Connecting to target...
J-Link found 1 JTAG device, Total IRLen = 5
JTAG ID: 0x07B7617F (ARM11)
Connected to target
Waiting for GDB connection...Connected to 127.0.0.1
Reading all registers
Read 4 bytes @ address 0x00000000 (Data = 0xEA000006)
Read 4 bytes @ address 0x00000000 (Data = 0xEA000006)
```

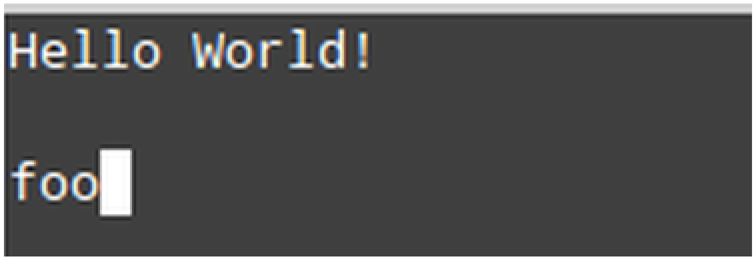
```
developer@toshiba-M70 ~/CPSC 359 Offline/template $ arm-none-eabi-gdb build/output.elf
GNU gdb (GNU Tools for ARM Embedded Processors) 7.8.0.20150604-cvs
Copyright (C) 2014 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.  Type "show copying"
and "show warranty" for details.
This GDB was configured as "--host=i686-pc-linux-gnu --target=arm-none-eabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from build/output.elf...done.
(gdb) target remote localhost:2331
Remote debugging using localhost:2331
0x00000000 in ?? ()
(gdb)
```

Command above is: arm-none-eabi-gdb build/output.elf (assuming you are in the project root folder)

Check to see if the UART interface works.



```
developer@toshiba-M70 ~ $ sudo screen /dev/ttyUSB0 115200
```



```
Hello World!

foo
```

Note: code was executed at this stage.

After executing the screen command, you should have a black screen without errors. If errors occur, try unplugging the UART USB cable then try again.

That's all folks! Let the development begin! Best of luck!