Neousys Technology Inc.

IGT-20/ IGT-21 Industrial Gateway

IGT-20 – Unboxing and configuration
Embedded systems for IIoT, Industry 4.0 and Automation

Table of contents

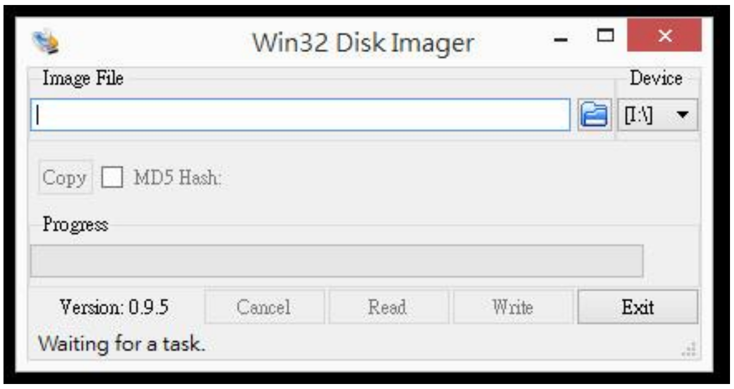
1. Make IGT-20 SD with Win32 Disk Imager 3
2. Using devmem2 on IGT-20 4
3. IGT-20 Persistent Tethering of WiFi 5
4. Access Auto Flow Control RS-485 on IGT-20 in CLI 5
   4.1. Auto Flow Control RS-485 on IGT-20 5
   4.2. Send Message over RS-485 on IGT-20 in CLI 5
   4.3. Receive Message over RS-485 on IGT-20 in CLI 6
5. Switching Modes of Serial Ports on IGT-20 to RS-232, RS-422 and RS-485 7
   5.1. Identify the Serial Port on IGT-20 7
   5.2. Switch the Mode of Serial Ports on IGT-20 7
6. IGT-20 OS Image and Disk Usage 8
7. IGT-20 microSD Access Speed 8
   7.1. Reading the microSD 9
   7.2. Writing the microSD 9
8. Configure IGT-20 to DHCP from Factory Setting 9
9. IGT-20 with Modbus for Environmental Monitoring 10
   9.1. Abstract 10
   9.2. System Overview 10
   9.3. Modbus RTU on IGT-20 10
   9.4. D3.js on IGT-20 11
   9.5. Conclusion 12
10. How to Modify Static IP Address of IGT-20 12
11. Connect to IGT-20 via Console Port 13
   11.1. Preparation 13
   11.2. Wiring 13
   11.3. Login with PuTTY 14
   11.4. Note 15

1. Make IGT-20 SD with Win32 Disk Imager

It's very common for users of embedded system, such as Raspberry Pi, Beaglebone and IGT-20, to use or upgrade to different OS. Of course, the first task is to get or build an OS image file, which is not covered in this article. Assuming that OS image is ready somewhere in your laptop or desktop PC running Windows, here's the way I make the SD with the OS image file.

The title has revealed that my favorite tool is Win32 Disk Imager. It can be downloaded at sourceforge. I've used it in Windows 7/8/10 without problems. After installation, an icon was created on your Desktop. If not, searching in the "Program Files" or "Program Files (x86)" might help.

Double-click the icon of Win32 Disk Imager to start it. If Windows UAC (User Account Control) pops up asking for authority, just click "Yes". And then the main window will appear soon.



Win32 Disk Imager can be used to write an OS image file of Raspberry Pi, Beaglebone and IGT-20 to an SD card, as well as backup the existing OS to a file. However, I have never saved the OS image in real cases when dealing with Beaglebone and IGT-20. In this article, IGT-20 is taken as an example. It's exactly the same procedure for Raspberry Pi, Beaglebone and others. The following figure shows steps to write IGT-20 image to an SD, located in disk "I" in my case.

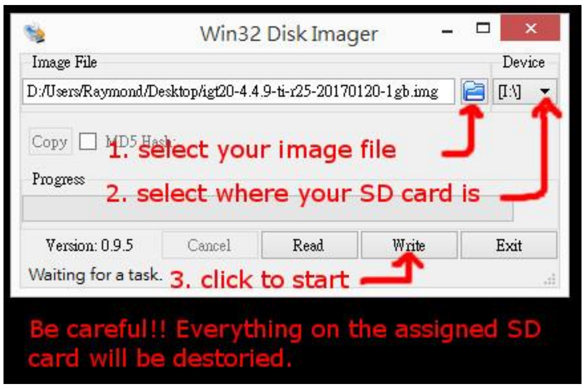


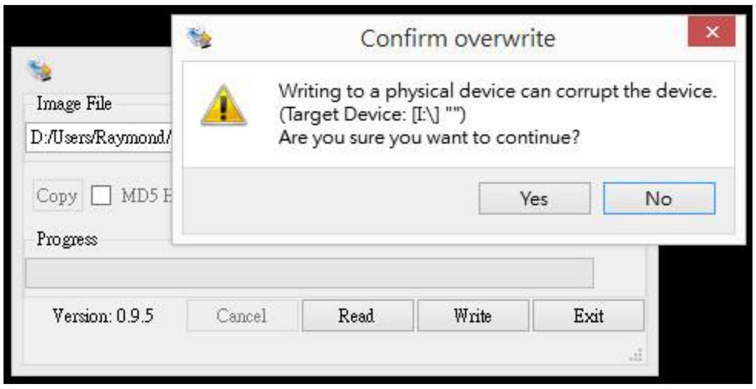
Image File: D:/Users/Raymond/Desktop/igt20-4.4.9-ti-r25-20170120-1gb.img

Device: [I:\]

1. select your image file
2. select where your SD card is
3. click to start

Be careful!! Everything on the assigned SD card will be destoried.

After the image file of IGT-20 and and SD card are selected, just click Write to start the writing process. Because all the data on the SD card will be destroyed, a confirm window will pop up, click Yes if you're sure the right SD is selected.


Confirm overwrite

Writing to a physical device can corrupt the device.
(Target Device: [I:\] "")
Are you sure you want to continue?

After Yes clicked, Win32 Disk Imager starts to write the OS image file of IGT-20 to the SD card. During the writing process, the green progress bar extends. Also the writing speed is shown at the bottom left corner.

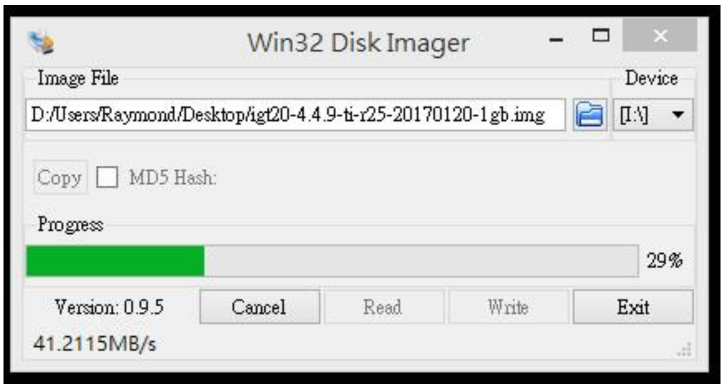

Win32 Disk Imager

Image File: D:/Users/Raymond/Desktop/igt20-4.4.9-ti-r25-20170120-1gb.img

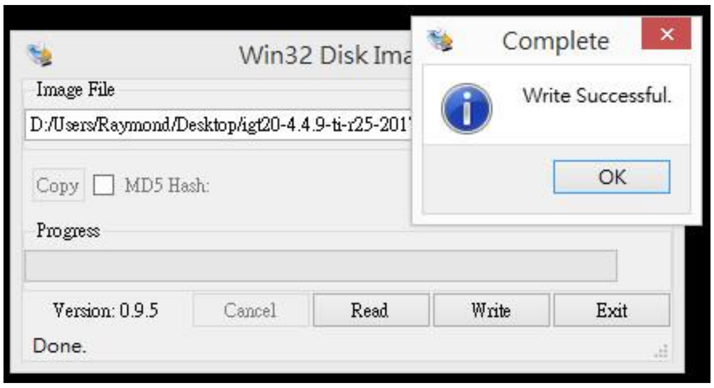
Device: [I:\]

Progress: 29%

Version: 0.9.5

41.2115MB/s

When the progress bar comes to 100%, the process finishes, and a prompt window pops up, saying Write Successful.


Complete

Write Successful.

Done.

Upon seeing this, you're almost done. Just click OK in the prompt window and the Exit in the main window to close Win32 Disk Imager. After that, remove the SD card from your laptop or desktop PC. Insert the SD card to IGT-20, turn on the power of IGT-20, and you're all set.

2. Using devmem2 on IGT-20

In many cases using an embedded system like the IoT Gateway IGT-20 from Neousys, one might need to read and write memory directly. There is a simple tool named devmem2, which origins from Jan-Derk Bakker. Since, the original link of the source file seems broken recently, a copy is attached here.

Just download the source file devmem2.c onto IGT-20, and compile it using the following command:

```
gcc devmem2.c -o devmem2
```

```
gcc devmem2.c -o devmem2
```

the output file will be an executable file, devmem2. And the syntax will be shown when it's executed without any parameters. Some example will be shown later when I'm free to write more.

3. IGT-20 Persistent Tethering of WiFi

By default, IGT-20 manages networking with connmanctl. One can use the following command, assuming that WiFi has been enabled, to enable tethering of WiFi on IGT-20.

```
connmanctl tether wifi on <SSID> <PassPhrase>
```

However, the tethering won't be enabled by default automatically after IGT-20 reboots or connman service restarts, for connman disables tethering by default after its service restarts. To make IGT-20 persistent tethering WiFi, please create a configuration file named main.conf with the following content:

```
[General]
PersistentTetheringMode=true
```

And put this main.conf in /etc/connman. Then it works.

4. Access Auto Flow Control RS-485 on IGT-20 in CLI

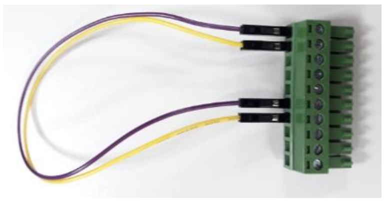
4.1. Auto Flow Control RS-485 on IGT-20

Due to the fact that RS-485 is half-duplexed, it can either send or receive messages in any single moment. This fact results in some cumberson work to switch the direction of messages of the RS-485. When this RS-485 plays as the role of a master in a communication scenario, it sometimes becomes an issue if the RS-485 consumes too much time to switch from sending back to receiving messages. In that case, this master might lose the response message from the slave it had sent the request.

The RS-485 on IGT-20 features its auto flow control. This functionality enabled users to use RS-485 on IGT-20 without manually controlling the direction of message. That is, the program for RS-232 is also applicable on RS-485 on IGT-20. It's very convenient and friendly since the serial ports on IGT-20 is configurable among RS-232, RS-422 and RS-485. Once different sensors or devices are required in different applications, users need not to modify their existing program dramatically. Switching the mode of serial port on IGT-20 is described in another article.

4.2. Send Message over RS-485 on IGT-20 in CLI

Before we start the evaluation, we need a proper fixture. And the most easy one is a simple loopback. What you need is only two wires, and here's what I did:



Data+ of ttyS1 and ttyS2 are connected (purple wire), and Data- of ttyS1 and ttyS2 are connected (yellow wire). Plug this fixture into CN3 of IGT-20, and preparation is done.

In this configuration, messages sent from ttyS1 is received by ttyS2, while message sent from ttyS2 is received by ttyS1. To perform the quick loopback test of serial ports on IGT-20, it's very easy to do it in the CLI. Just type the following command in CLI:

```
echo "message from ttyS1" > /dev/ttyS1
```

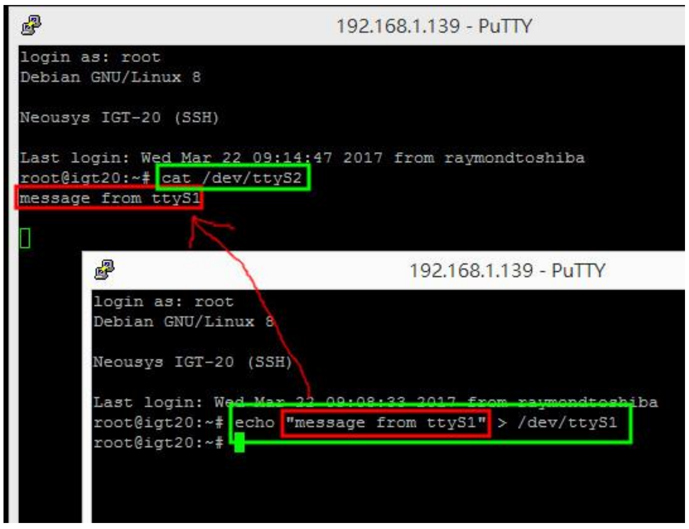
And the message is sent. However, you can see nothing about it. If you'd like to see to believe, let's now try to catch the messages on ttyS2.

4.3. Receive Message over RS-485 on IGT-20 in CLI

To make it simple for the test, just create another SSH connection to the same IGT-20. And type the following command in CLI:

```
cat /dev/ttyS2
```

After that, once you send message from ttyS1 as mentioned above, you can see the message shown in this SSH client. Here's my screen snap-shot.

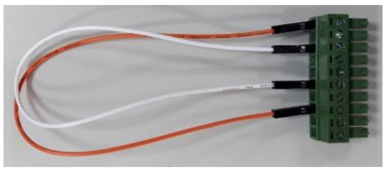


```
login as: root
Debian GNU/Linux 8

Neousys IGT-20 (SSH)

Last login: Wed Mar 22 09:14:47 2017 from raymondtoshiba
root@igt20:~# cat /dev/ttyS2
message from ttyS1
```

```
login as: root
Debian GNU/Linux 8

Neousys IGT-20 (SSH)

Last login: Wed Mar 22 09:08:33 2017 from raymondtoshiba
root@igt20:~# echo "message from ttyS1" > /dev/ttyS1
root@igt20:~#
```

In this figure, text in green rectangular is the command, while the red ones are the messages. the "message from ttyS1" in the bottom window is sent to the top one.

During the test, nothing has been done to change the direction of RS-485, thanks to the auto flow control feature of IGT-20. And actually, this can be duplicated with RS-232 and RS-422 on IGT-20. What you need is another fixture. Here's, by the way, my fixture of RS-232.



Rx of ttyS1 and Tx of ttyS2 are connected (orange wire), while Tx of ttyS1 and Rx of ttyS2 are connected (white wire). With this wiring, messages from ttyS1 is sent to ttyS2 over RS-232, while message from ttyS2 is sent to ttyS1.

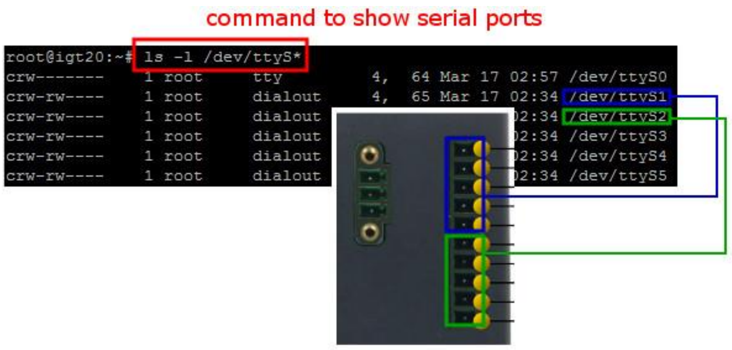
The most important thing is that don't forget to switch the modes of the serial ports to that matches your fixture.

5. Switching Modes of Serial Ports on IGT-20 to RS-232, RS-422 and RS-485

Designed as an industrial gateway for IoT, IIoT, Industry 4.0 and automation applications, IGT-20 has two serial ports enabling the connection to various sensors and devices. And among the sensors and devices, serial communication, i.e. RS-232, RS-422 and RS-485, is a very common interface. This article shows you how to switch modes of serial ports on IGT-20 to RS-232, RS-422 and RS-485.

5.1. Identify the Serial Port on IGT-20

IGT-20 supports two serial ports, located on the top side of it. It's the first step to ensure that wiring is correct. The following figure shows the mapping between devices in OS and devices on the connector.

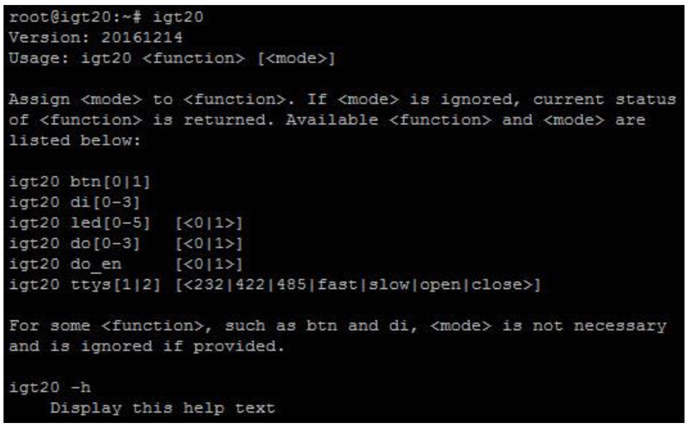


```
root@igt20:~# ls -l /dev/ttyS*
crw-------    1 root     tty        4,  64 Mar 17 02:57 /dev/ttyS0
crw-rw----    1 root     dialout    4,  65 Mar 17 02:34 /dev/ttyS1
crw-rw----    1 root     dialout            02:34 /dev/ttyS2
crw-rw----    1 root     dialout            02:34 /dev/ttyS3
crw-rw----    1 root     dialout            02:34 /dev/ttyS4
crw-rw----    1 root     dialout            02:34 /dev/ttyS5
```

The figure shows only the whole groups of pins. The detail pin-out definition is revealed in the manual of IGT-20.

5.2. Switch the Mode of Serial Ports on IGT-20

The two serial ports on IGT-20 are configurable. It's possible to switch the mode among RS-232, RS-422 and RS-485. A sample script can be found in /usr/bin/igt20 to accomplish several on-board functionalities, such as access DI/O's, user buttons, user LEDs. Simply typing igt20 in CLI will show you the usage of this script. And of course, you can read the source code in detail if you'd like to.



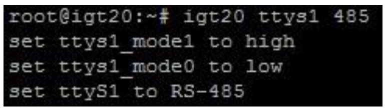
```
root@igt20:~# igt20
Version: 20161214
Usage: igt20 <function> [<mode>]

Assign <mode> to <function>. If <mode> is ignored, current status
of <function> is returned. Available <function> and <mode> are
listed below:

igt20 btn[0|1]
igt20 di[0-3]
igt20 led[0-5]   [<0|1>]
igt20 do[0-3]    [<0|1>]
igt20 do_en      [<0|1>]
igt20 ttys[1|2]  [<232|422|485|fast|slow|open|close>]

For some <function>, such as btn and di, <mode> is not necessary
and is ignored if provided.

igt20 -h
    Display this help text
```

As the help message shows, it supports also switching modes of serial ports. We can switch ttyS1 to RS-485 the using the following command:

```
igt20 ttys1 485
```

```
1    igt20 ttys1 485
```

Here’s the snap shot of screen.

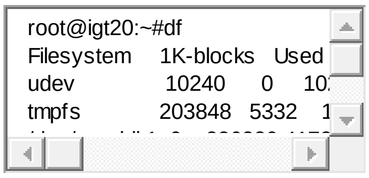


```
root@igt20:~# igt20 ttys1 485
set ttys1_mode1 to high
set ttys1_mode0 to low
set ttyS1 to RS-485
```

Upon you seeing this, IGT-20 is ready to run RS-485 over ttyS1. By using the same approach, it’s also possible to switch the mode of serial ports on IGT-20 to RS-232 and RS-422. Have fun!

6. IGT-20 OS Image and Disk Usage

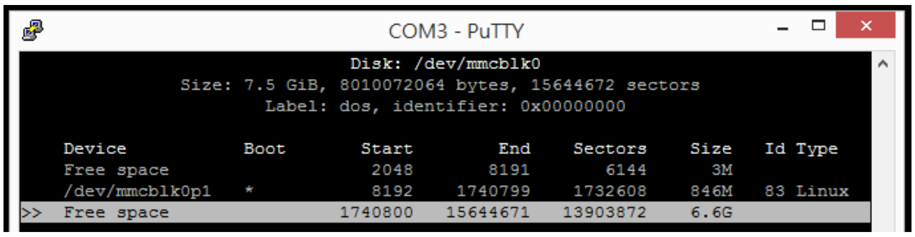
Currently the pre-installed OS of IGT-20 is Debian 8, and the OS image is less than 1GB in a 8GB microSD. By using df command, one can observe the real disk usage as below:



```
root@igt20:~#df
Filesystem    1K-blocks   Used
udev            10240      0    10
tmpfs          203848   5332    1
```

```
root@igt20:~#df
Filesystem     1K-blocks   Used Available Use% Mounted on
udev              10240      0     10240   0% /dev
tmpfs            203848   5332    198516   3% /run
/dev/mmcblk1p0   836280 417880    358704  54% /
tmpfs            509616      0    509616   0% /dev/shm
tmpfs              5120      0      5120   0% /run/lock
tmpfs            509616      0    509616   0% /sys/fs/cgroup
```

The image occupies around 850MB while 54% is still not used. With another command cfdisk, one can find the total microSD capacity and the free space



COM3 - PuTTY

```
                              Disk: /dev/mmcblk0
            Size: 7.5 GiB, 8010072064 bytes, 15644672 sectors
                   Label: dos, identifier: 0x00000000

    Device          Boot      Start        End     Sectors    Size   Id Type
    Free space                 2048       8191        6144      3M
    /dev/mmcblk0p1  *          8192    1740799     1732608    846M   83 Linux
>>  Free space              1740800   15644671    13903872    6.6G
```

As shown in the picture, the real capacity is 7.5GB (8,010,072,064 bytes) with 6.6GB not located. Since IGT-20 provides the other external microSD slot, it’s better to store the user data in the external microSD. However, in some cases, if you’d not like to have another SD, it’s better to create a new partition using the 6.6G free space instead of extending the original partition.

7. IGT-20 microSD Access Speed

IGT-20 provides dual microSD slots and supports up to SDHC. Read/Write test has been performed with the default 8GB MLC microSD. Here’s the methodology and test result:

7.1. Reading the microSD

```
root@igt20:~# hdparm -tT /dev/
/dev/mmcblk0:
Timing cached reads: 474 MB ir
Timing buffered disk reads: 62 
```

```
root@igt20:~# hdparm -tT /dev/mmcblk0
/dev/mmcblk0:
Timing cached reads: 474 MB in 2.00 seconds = 237.00 MB/sec
Timing buffered disk reads: 62 MB in 3.07 seconds = 20.21 MB/sec
```

7.2. Writing the microSD

```
root@igt20:~# dd if=/dev/zero c
1+0 records in
1+0 records out
536870912 bytes (537 MB) cop
```

```
root@igt20:~# dd if=/dev/zero of=/media/512M bs=512M count=1
1+0 records in
1+0 records out
536870912 bytes (537 MB) copied, 48.1249 s, 11.2 MB/s
```

8. Configure IGT-20 to DHCP from Factory Setting

The LAN of IGT-20 is set to static IP, 192.168.8.2, for users' convenience when it's shipped out of factory. This facilitates users to connect to your IGT-20 by SSH via a common CAT-5e cable as soon as you get them. However, if you'd like to connect your IGT-20 to Ethernet for some reasons, such as upgrade, installation of new software packages and so on, the factory setting doesn't work well.

It's strongly recommended the console port is used if you'd like to change IGT-20 to DHCP because the new IP assigned by DHCP server to IGT-20 might be unknown to you. By using console port, it's possible for you to know the new IP. Here's a very simple way, as root, to make it done:

```
rm -rf /var/lib/connman/*cable
systemctl restart connman<cod
```

```
rm -rf /var/lib/connman/*cable
systemctl restart connman<code>
```

Please note that this approach will delete ALL configured wired networks on the IGT-20. Although there's only one LAN on IGT-20, please still take a little more care. In case that you'd like to specifically configure the network setting, please refer to ConnMan man page.

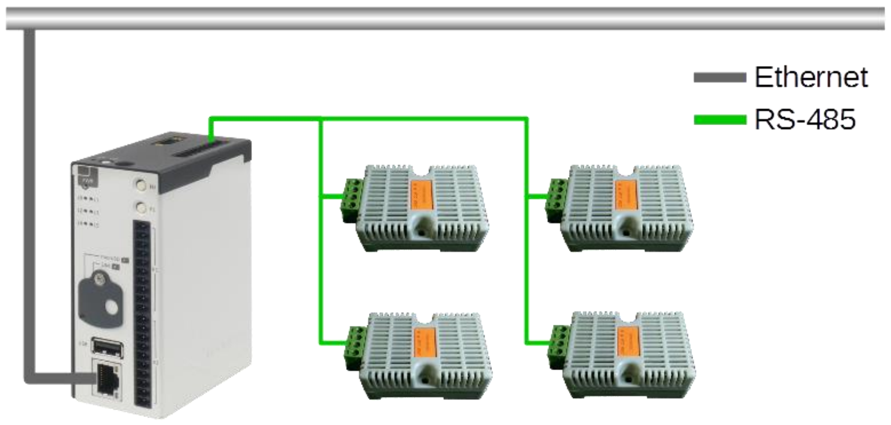
9. IGT-20 with Modbus for Environmental Monitoring

9.1. Abstract

This article describes a real installation of IGT-20 that collects information of temperature and humidity in an electric component warehouse. In this installation, IGT-20 accesses to four sensors which run a very popular Modbus RTU protocol over the RS-485 interface. IGT-20 keeps monitoring if the values go beyond the expected limits. Historic data are stored in IGT-20 and visualized with D3.js. And also the stored data can be uploaded to cloud for further analysis.

9.2. System Overview

The following figure reveals the overview of the described system.



In order to know the temperature and humidity at different locations, it's very common to use multiple sensors. And these location might be far away from one another. Thus the multi-drop wiring and long distance features of RS-485 make it a good choice. There totally four sensors in this installation at the four corner of this warehouse.

IGT-20 acts as a gateway to inquiry data from these sensors via its built-in RS-485 and upload it to private and public clouds via its built-in LAN. Beside of the transparent transmission, IGT-20 can also monitors, stores and visualizes the data.

With totally two SDHC slots, IGT-20 makes it possible to keep user data up to around 60GB. Considering some applications, such as discrete manufacturing, environmental monitoring, health care and so on, this is a sufficient storage and enables advanced features.

IGT-20 monitors the local and real time data in two ways. The first is to check if each single datum is in a expected range. Besides, the locally kept historic data enable IGT-20 to check the trend and stability of data. This ensures the electronic components staying in a good condition. Further complicated off-line analysis is also possible after these data is uploaded.

Apache Server is installed in IGT-20 to provide cross-platform visualization of the locally stored data. Managers and warehouse keepers can use their desktop, handy phone and tablet to inquiry the data or to see the whole trend chart just with those commonly used web browsers.

The built-in mPCie slot has been reserved and not used yet. By installing a 3G or LTE module, IGT-20 can use SMS to send short message to warehouse keepers once abnormal situation occurs. However, this is not covered in this article.

9.3. Modbus RTU on IGT-20

Modbus is a de-facto standard for industrial devices and uses the master-slave architecture. That is, the master asks, and then the assigned slave reacts. The frame format is quite simple for implementation. And the hardware requirement is low. Thus Modbus has been a popular protocol and and still widely used around the world.

As an industrial gateway, IGT-20 provides two built-in configurable RS-232/422/485 serial ports to connect to devices with different interfaces. The modes, i.e. RS-232, RS-422 and RS-485, is switched by setting two internal GPIOs. This is covered by another article.

The auto flow control feature of RS-485 of IGT-20 makes it easier to program for half-duplex interface. It's a wise choice to use sensors with Modbus RTU over RS-485 with IGT-20. According to the manual of the sensor, the default Device ID is 0x01, and temperature and humidity are available at Holding Register 0x02 and 0x03 respectively. What we need to do is open the serial port and send the following hex out.

```
0x01 0x03 0x00 0x02 0x02 0x65 0xCB
```

If everything is fine, the response can be read

```
0x01 0x03 0x04 0xTT 0xtt 0xHH 0xhh 0xRR 0xrr
```

where

```
temperature = (TT*256+tt)/10.0
humidity = (HH*256+hh)/10.0
```

RR and rr are the CRC. It's not difficult to complete this implementation. However, a Modbus library has been developed by libmodbus.org, and has been put into Debian repositories. Although libmodbus is not included in the factory OS image, it can be installed by the following command:

```
apt-get update && apt-get upgrade
apt-get install libmodbus-dev
```

Using libmodbus make it much easier to read temperature and humidity with the following pseudo code:

```
ctx=modbus_new_rtu(TTY_DEVICE, BAUDRATE, PARITY_CHECK, DATA_BITS, STOP_BITS);
modbus_connect(ctx);
modbus_set_slave(ctx, DEVICE_ID);
modbus_read_registers(ctx, 2 /*start addr*/, 2 /*# of registers*/, readData);
temperature=readData[0];
humidity=readData[1];
modbus_close(ctx);
modbus_free(ctx);
```

The CRC field and error handling of communication are done by libmodbus. Focus can be put on the data itself.

9.4. D3.js on IGT-20

Using the ready library libmodbus is only part of IGT-20 benefit. The pre-installed Debian comes with more than 50,000 packages. Apache is another commonly used software package. Apache is a very popular web server. Around 50% of the on-line web server is based on Apache. Apache can also be installed by the following command:

```
apt-get update && apt-get upgrade
apt-get install apache2
```

Installing Apache turns IGT-20 into a small web server and connects IGT-20 with modern web technologies, such as AJAX and HTML5. One of the possibilities is to make IGT-20 accessible on different platform, such as desktops, handy phones and tablets. In this installation, visualization of the stored data is done with D3.js.

D3.js is a JavaScript library and helps to bring data to life using HTML, SVG and CSS. With a few lines of code of D3.js and JavaScript, an interactable and clear real-time chart can be implemented. This provides a good choice of UI for local data. Again once it's done, it's cross-platform accessible.

Detail of implementation is not covered here. Following is the screen shot from a handy phone.

Though there's no display output of IGT-20, there is still much possibility to create a better user interface if necessary.

9.5. Conclusion

This picture was taken where this installation deployed. IGT-20 is at the bottom of the picture, two sensors locates at right side of this picture, while another two are outside of the picture. In this installation, IGT-20 acquires temperature and humidity data via RS-485 with Modbus RTU protocol. By using lib

modbus, a free software library to send/receive data according to the Modbus protocol, it's very easy to turn IGT-20 into a Modbus master and get data from slaves. And there are many other Modbus sensors, such as air quality and air pressure, for environmental monitoring.

IGT-20 equips with Cortex-A8, 1GB RAM and pre-installed OS, Debian, which comes with around 50,000 software packages to facilitate your work. Besides, around 60GB storage of user data is provided by the dual SDHC slots. This makes it possible to store the data and make a local analysis.

The raw data, analysis result and alarm if any can be uploaded to private or public cloud via the built-in LAN, or optional WiFi/3G/LTE modules depending on your application. The real time trend chart, or any other visualization, is also possible and made easy by off-the-shelf packages, such as Apache and D3.js.

If your work needs only transparent transmission, there are many other lower level products which meet the needs well. However, if local storage, on-line analysis and visualization of data are considered, IGT-20 can be a better solution.

10.How to Modify Static IP Address of IGT-20

The pre-installed Debian of IGT-20 is provided with ConnMan and already configured with a static IP 192.168.8.2 by default. With the factory configuration, i.e. no other networking modification being done, the Static IP Address can be changed by editing ConnMan configuration file. The configuration file can be opened with the following command:

```
root@igt20:~# nano /var/lib/connman/*cable/settings
```

The content of this configuration file looks like

```
[ethernet_############_cable]
Name=Wired
AutoConnect=true
IPv4.method=manual
IPv4.netmask_prefixlen=24
IPv4.local_address=192.168.8.2
IPv4.gateway=192.168.8.1
```

where ############ is the MAC address of each IGT-20. If something similar is not shown, probably the default configuration has been modified already. Another approach will be better for this situation, please press ctrl-x and then enter "No" to exit the editor without saving. Assume that everything is identical except the mentioned MAC address, just modify the red numbers shown above according to your network environment.

An valid IP address configuration is composed of 3 parts, the IP address, the subnet mask and the gateway address. If you have difficulties getting these three parameters, please consult your IT people. Once the parameters are clear, assign the IP address to IPv4.local_address, the subnet mask to IPv4.netmask_prefixlen and gateway IP address to IPv4.gateway. After that, press ctrl-x and then enter "Yes" to save your modification, exit the editor and go back to command line prompt.

Be sure that your networking configuration is correct because the improper settings will keep you away from connecting to IGT-20 with SSH again after ConnMan service is restarted. To restart the ConnMan service, enter the following command via the command line:

```
root@igt20:~# systemctl restart connman
```

If you connect to IGT-20 with SSH, you'll need to reconnect to IGT-20 according to new configuration including the host with proper networking configuration. If the new SSH can be established, congratulation you're all set. If not, you'll need to login via the console port to check your network configuration.

11.Connect to IGT-20 via Console Port

This article describes how to connect IGT-20 via console port

11.1. Preparation

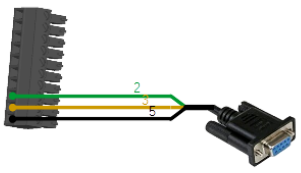
To connect and log in to IGT-20, we'll need to prepare the following items:

- IGT-20
- 10-pin terminal
- Power supply unit, PSU
- Open end to female DB-9 cable
- A console client with RS-232 port

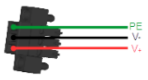
Of course, we will need a IGT-20 and a proper PSU. IGT-20 accepts a wide DC input range from 8 to 25 VDC. A PSU with output between this range usually works fine. The console port of IGT-20 runs via RS-232, and locates at the connector X1 with a 10-pin terminal block which is shipped with IGT-20. And thus we need an open-end DB-9 cable to connect from X1 to a COM port on the console client, which is usually a PC or laptop.

11.2. Wiring

Connect the open end of the DB-9 cable to the 10-pin terminal block as the following figure. And push this terminal block into X1 of IGT-20, while the DB-9 into the COM port of the host.

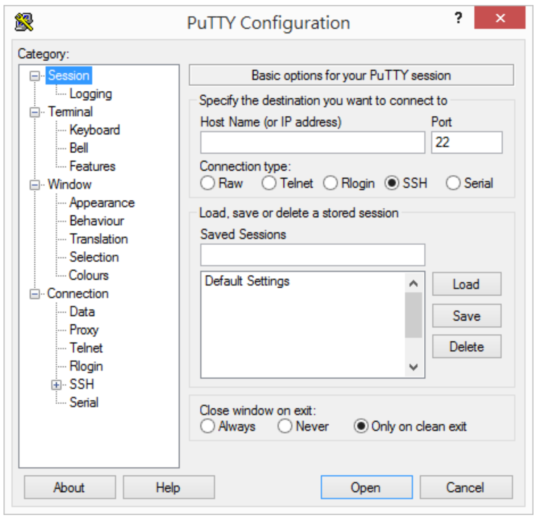


The DC input connector is a 3-pin terminal block. Connect it to PSU as the following figure.



11.3. Login with PuTTY

PuTTY is commonly used console client software running on Windows and Linux. Information can be found here. Assume that PuTTY is well installed on the client PC with COM port connected to IGT-20 as described earlier, open this PuTTY application.



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

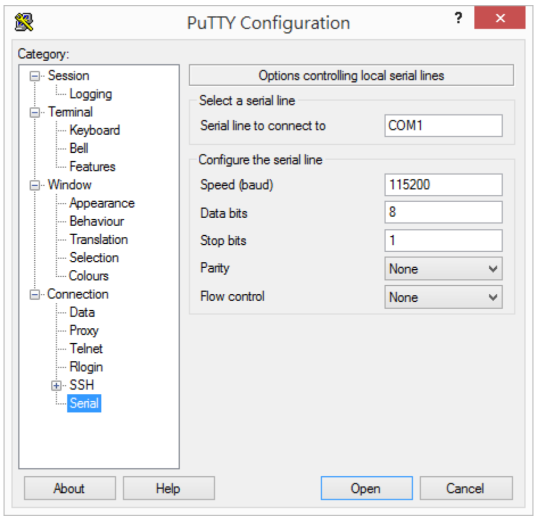
Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: Always Never [✓ Only on clean exit]

Since we are going to connect to IGT-20 via COM, click "Serial" in the Category box to the left of the window.



Options controlling local serial lines

Select a serial line

Serial line to connect to: COM1

Configure the serial line

Speed (baud): 115200

Data bits: 8

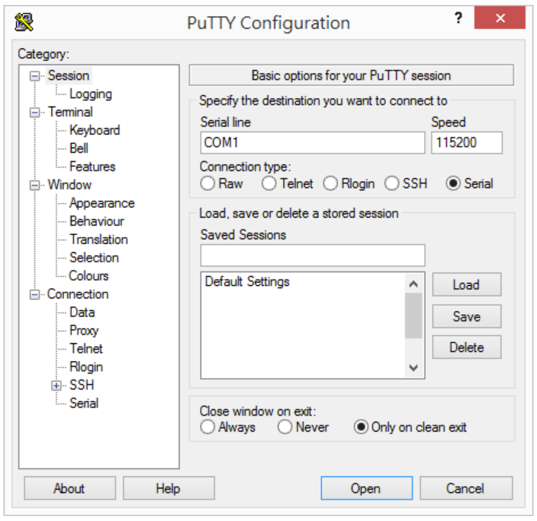
Stop bits: 1

Parity: None

Flow control: None

Assign the communication parameters, i.e. Speed, Data bits, Stop bits, Parity and Flow control, as shown in the figure right above. The box "Serial line to connect to" has to been fitted with the correct COM port notation. This figure was taken in Windows. The value might be something like "/dev/ttyS1" in Linux.

After communication parameters are assigned, click “Session” to back to the previous window, and then choose the Connection type to “Serial”. At this moment, PuTTY looks like the following figure except that the Serial line might be different depending on which COM port is used.



Specify the destination you want to connect to

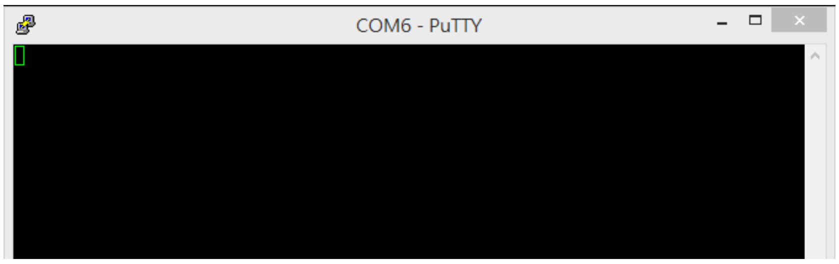
Serial line: COM1

Speed: 115200

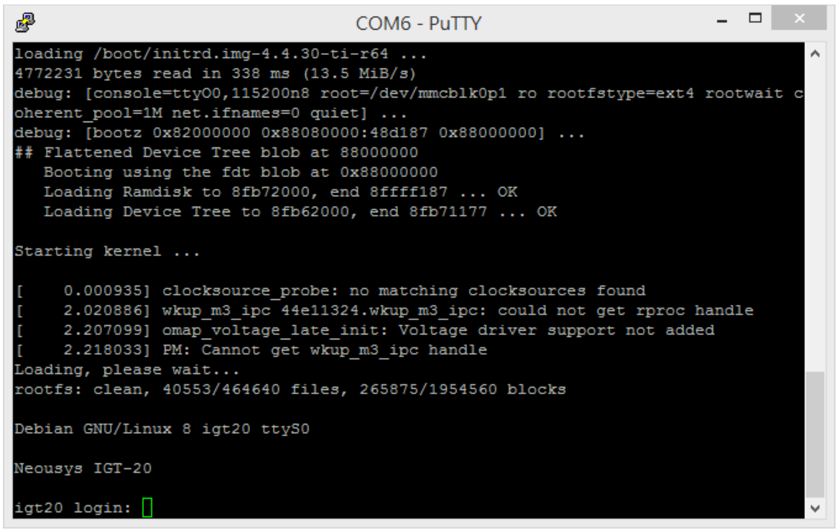
Connection type: Raw Telnet Rlogin SSH [✓ Serial]

Close window on exit: Always Never [✓ Only on clean exit]

Now click Open to start the communication. If everything is fine, a console window will pop up.



There is no any content shown because IGT-20 has not yet powered up. After turn it on, we can see messages passing by for a while, and come to the login prompt eventually.



```
loading /boot/initrd.img-4.4.30-ti-r64 ...
4772231 bytes read in 338 ms (13.5 MiB/s)
debug: [console=ttyO0,115200n8 root=/dev/mmcblk0p1 ro rootfstype=ext4 rootwait c
oherent_pool=1M net.ifnames=0 quiet] ...
debug: [bootz 0x82000000 0x88080000:48d187 0x88000000] ...
## Flattened Device Tree blob at 88000000
   Booting using the fdt blob at 0x88000000
   Loading Ramdisk to 8fb72000, end 8ffff187 ... OK
   Loading Device Tree to 8fb62000, end 8fb71177 ... OK

Starting kernel ...

[    0.000935] clocksource_probe: no matching clocksources found
[    2.020886] wkup_m3_ipc 44e11324.wkup_m3_ipc: could not get rproc handle
[    2.207099] omap_voltage_late_init: Voltage driver support not added
[    2.218033] PM: Cannot get wkup_m3_ipc handle
Loading, please wait...
rootfs: clean, 40553/464640 files, 265875/1954560 blocks

Debian GNU/Linux 8 igt20 ttyS0

Neousys IGT-20

igt20 login:
```

Type “root” here and then hit Enter. You’ll get into the default shell of IGT-20 and enjoy.

11.4. Note

It’s strongly advised to change the password after initial login for the sake of system security.