www.iter-systems.com
support@iter-systems.com

Reference	ETD-2020
Classification	Unclassified
Version	2.11
Date	17/07/21
Copy number *(if applicable)*	N/A

SwathRT Manual

This page is left blank intentionally

Voids

10		
9		
8		
7		
6		
5		
4		
3		
2		
1		
	Section	**Notes**

List of modifications

Version	**Date**	**Modifications**	**Pages**	**Writer**	**Checker**
2.11	17/07/21	Updates from use with Omega box		MFG	
2.10	15/04/21	Adding instructions for compiling on Windows		MFG	
2.09	19/02/21	Correcting TEM input-output port numbers		MFG	
2.08	28/08/20	Review and various small corrections		MFG	
2.07	18/08/20	Updating deploy instructions		MFG	
2.06	22/06/20	Added cross-compile instructions		MFG	
2.05	19/03/20	Technical review		MFG	
2.04	16/03/20	New Graphic chart	28	FBY	
2.03	04/12/19	Notes on filter settings, more on PPS	28	MFG	
2.02	07/11/19	Added Interfacing section		MFG	
2.01	27/09/19	More notes on compiling code		MFG	
2.00	22/07/19	Merging with RPi setup manual		MFG	
1.01	24/09/18	Review and update		MFG	
1.00	01/11/17	Initial document		MFG	

Table of contents

1 INTRODUCTION

1.1 REFERENCES

Ref 1 Getting Started, “Bathyswath Getting Started.docx”: introductory manual for Bathyswath systems
Ref 2 Bathyswath file formats manual, “ETD_2010_Bathyswath_File formats.pdf”
Ref 3 Bathyswath Parsed file format manual, “ETD_2011_Bathyswath_Parsed file format.pdf”
Ref 4 Bathyswath-2 OEM integration manual, “ETD_2006_Bathyswath-2_OEM integration manual.pdf”
Ref 5 ITER Systems website, at http://iter-systems.com
Ref 6 ITER Systems swathRT GitHub, at https://github.com/iter-systems/swathRT
Ref 7 Qt IDE web page, at https://www.qt.io/ ,
Ref 8 GitHub repository help, at https://docs.github.com/en/github/creating-cloning-and-archiving-repositories/about-repositories

1.2 GLOSSARY & ACRONYMS

ACRONYMS	DEFINITION
GUI	Graphical user interface
swathDisplay	A program that receives Bathyswath raw data over UDP and displays a summary of it on a screen
swathRT	A program that interfaces to a Bathyswath TEM to control it, acquire its data and store it to disk or pass it on by UDP messages
swathRT_cmd	A program that controls a Bathyswath system via swathRT, by sending UDP command messages to swathRT
TEM	Transducer electronics module: interfaces with Bathyswath sonar transducers; generates transmit pulses and amplifies, digitises and transmits received data

1.3 PREFACE

The Bathyswath User Documentation covers all models in the series. The documentation is divided into these main parts:

Getting Started - the paper part of the documentation, which introduces Bathyswath and covers aspects such as software and hardware installation and deployment. [Ref 1]

Online User Guide - covering all aspects of using Bathyswath as a hydrographic surveying tool. The guide is accessed from the Swath Processor **Help menu.**

Grid Processor manual – This provides instructions for using the Bathyswath Grid Processor program.

SwathRT manual – ***This document***: this provides instructions for installing, configuring and using the swathRT suite of programs for controlling Bathyswath in real time, particularly on non-Windows operating systems.

Installation instructions – the software and other components are supplied with specific installation instructions, which build in the installation guides provided in this manual.

Auxiliary equipment manuals – Bathyswath is usually supplied with auxiliary equipment, such as attitude sensors, positioning systems and compasses. These will have been supplied with their own manuals and/or online guides, and the operator should also read these before using the system.

The ITER Systems website, iter-systems.com – [Ref 5]

1.4 OPERATING SYSTEMS

Most Bathyswath systems use the Bathyswath software running on a computer running the Windows operating system, connected directly to the Bathyswath electronics. But designers of remotely operated and autonomous systems prefer to use computers that run the Linux operating system, as these tend to be smaller, lighter, and consume less power. Bathyswath can be used with such Linux computers, using the swathRT software, which is described in this manual.

The swathRT software is developed in the Qt development platform, which is designed to produce code that can be compiled on a wide range of operating systems. ITER Systems can supply swathRT compiled for Raspberry Pi Linux and Windows, and most other Linux variants and other operating systems can be compiled for by clients or by ITER Systems.

1.5 PARTS OF A BATHYSWATH SYSTEM

The Bathyswath electronics hardware is called the Transducer Electronics Module (TEM).

The Bathyswath Windows software is called Swath Processor.

Most Bathyswath remote systems have three parts, connected to the same network, with some links using Wi-Fi or similar wireless connections:

- Bathyswath Transducer Electronics Module,
- Linux computer, running SwathRT,
- Windows computer, running Swath Processor.

For general information on setting Bathyswath systems, see the Getting Started manual [Ref 1].

1.6 RASPBERRY PI

The specific instructions in this manual have been developed and tested on the Raspberry Pi microcomputer (“RPi”), which is an excellent small and cheap but powerful microcomputer. It works well with Bathyswath systems, and is good for use on small platforms such as small USVs.

RPi has been used with Bathyswath-2 systems, and is included in the Bathyswath-2-Omega system and the Bathyswath-3 system, which is being developed at the time of writing.

Standalone RPi computers are convenient to use; at the time of writing the best version to use is the Raspberry Pi 4 Model B (“4B”). Bathyswath-2-Omega and Bathyswath-3 use Raspberry Pi Compute Modules. The instructions in this document apply to the 4B; there may be differences in the Compute Module configuration.

The examples given and the compiled software are for the default “Raspbian” operating system, which is a version of Linux.

Most of the examples given in this manual should also work on other Linux systems.

Executable software that has been compiled on and for Raspbian work on some, but possibly not all, other Linux versions.

2 THE SWATHRT SOFTWARE

2.1 PROGRAMS IN THE SWATHRT SUITE

The swathRT programs are:

- **swathRT** A program that interfaces to a Bathyswath TEM to control it, acquire its data, and store it to disk or pass it on by UDP messages.
- **swathRT_cmd** A program that controls a Bathyswath system via swathRT, by sending UDP command messages to swathRT.
- **swathDisplay** A program that receives Bathyswath raw data over UDP and displays a summary of it on a screen. However, most systems use a separate Windows computer running **Swath Processor**, which provides more features for viewing data from Bathyswath and associated sensors, and so is used instead of swathDisplay. This program is useful when ping data sent from swathRT needs to be viewed in Linux.
- **SimBSW** A software simulator for the Bathyswath hardware, which can be used for testing software.

2.2 THE PURPOSE OF SWATHRT

swathRT was originally developed to allow Bathyswath systems to run on autonomous vehicles that use small Linux-based embedded computer systems. The main Bathyswath software package runs on Windows and requires a relatively powerful modern computer to run its full set of functions and displays. In contrast, swathRT is lightweight, can be compiled for any operating system, and has the minimum of functions that are required for running Bathyswath in real time.

In Bathyswath-3, swathRT is the generic interface for Bathyswath sonars as standard, using a small computer platform built into the Bathyswath hardware, thus avoiding the need to run the Swath Processor application when working with third-party processing software systems.

2.3 BATHYSWATH SYSTEM SOFTWARE NETWORK

The Bathyswath electronics hardware is called the Transducer Electronics Module (**TEM**).

The Bathyswath Windows software is called **Swath Processor**.

Most Bathyswath remote systems have three parts, connected to the same network, with some links using Wi-Fi or similar wireless connections:

- Bathyswath **TEM** (Transducer Electronics Module).
- **Linux** computer, running **swathRT.**
- Where there is a connection to shore, a **Windows** computer, running **Swath Processor.**

Several variations are possible; for example, swathRT can run on the Windows computer, or swathRT can run stand-alone with no connection to an external computer.

3 INSTALLATION

3.1 INSTALLATION SOURCES

Installation packs are available for Windows and Raspberry Pi Raspbian Linux. The swathRT software suite can be built for most other Linux flavours and operating systems. The simplest way to do this is for the client to install Qt Creator on their own system and to compile the code locally. Contact ITER Systems to request this.
swathRT can also be cross-compiled on a Windows computer to run on a separate Linux computer. See Annex F.

3.2 CONFIGURING LINUX SYSTEMS FOR SWATHRT

The instructions below are for installing swathRT on the Raspberry Pi Linux system, Raspbian. Most of these instructions should also be valid for other Linux platforms.

3.2.1 Operating System

Install the default Raspbian operating system, using the SD card provided, or by installing an operating system image, as described here. Operating system downloads are available from here. It is usually simplest to install the NOOBS operating system installer first, and then to run that to install Raspbian. See 0 for instructions on setting up a Raspberry Pi Compute Module.

3.2.2 Wi-Fi and Internet Access

If you are using the Ethernet network for a fixed, local addressing link (see section 3.2.4 below), then you can't configure it for Internet access. So, you need to use Wi-Fi to access the Internet, which you need to update the operating system and get software resources.

- Left-click on the network icon in the top-right of the screen.
- A list of the available networks should be shown; select the one that you want to use.
- Enter the wireless key in the "Pre-shared key" box.
- Testing: try the Chromium web browser. If you set up the Wi-Fi after connecting an Ethernet cable to a laptop, the Chromium web browser might not be able to access the Internet, because it will have defaulted to try to use the Ethernet link. Try disconnecting the Ethernet and try again.

It may be simpler to use Wi-Fi for configuration, and leave the Ethernet cable disconnected.

To use VNC and SSH without the Ethernet:

- In VNC, use "raspberrypi" as the VNC Server address, ***or***
- Find the IP address allocated to the RPi by the DHCP server:
  - In a Raspbian shell, type: `ifconfig`,
  - Note the IP address given for wlan0, and use that for VNC and SSH logins.

Systems that use the ITER Systems NCPB board, such as Bathyswath-2 Omega and Bathyswath-3, might not have Wi-Fi included as standard. If so, simply connect a Wi-Fi dongle to a USB port in the system.

Alternatively, to access the Internet over the RPi's Ethernet connection:

- Connect VNC using Wi-Fi as shown above.

- Connect the Ethernet port of the RPi to a DHCP server (for example, the Ethernet ports on the back of the Wi-Fi hub that connects to the Internet over a phone line). Connect directly, not through an Ethernet switch (that might not work).
- Set the Ethernet port to DHCP by clearing all the entries in the IP settings, as shown in 3.2.4 below. (You will need to restore those settings later).

Or, to share the computer's Wi-Fi connection to the Internet with the RPi:

- Connect the RPi to PC's ethernet port using an ethernet cable
- Go to "Network Connections" on the Windows PC and select the "Wireless Network Connection"
- Right-click and select properties. In the "Sharing" tab under "Internet Connection Sharing" enable both checkboxes.
- This step will warn you that the IP address of the PC's Ethernet connection will change, e.g. to 192.168.137.1. So you will need to change the IP address of the RPi too, to an address in that domain, e.g. 192.168.137.80.
- Restart the PC.
- Now the RPi will obtain an IP address from the PC and can access internet through your PC
- If need to find the IP address of the RPi to SSH or to remote login from the PC, run "ping raspberrypi" command; this should show the address of the RPi.
- (*This option doesn't seem to work when we tested it*)

3.2.3 Update the Operating System

In a terminal:

```
sudo apt-get update
```

Then

```
sudo apt-get upgrade
```

and

```
sudo apt-get dist-upgrade
```

3.2.4 IP Address

Most systems use a fixed IP address to connect to the TEM. However, both the TEM and RPi can also be set to use DHCP addressing, where addresses are allocated by a server on the network. The instructions in this document assume fixed addressing.

The TEM IP address defaults to 192.168.0.240, but can be changed using Swath Processor.

- Right-click on the network icon in the top-right of the screen,
- Select "… Network Settings",
- Select "Interface" and "eth0",
- Enter:
  - IP Address: 192.168.0.80 (or whatever IP address works best),
  - Router: 192.168.0.1,
  - DNS Severs: 192.168.0.1.
- Click Apply,
- Reboot,
- Test by:
  - Connect an Ethernet cable between a laptop and the RPi,
  - "ping 192.168.0.80" from a Windows command shell on the laptop.
- If a second Ethernet port is used in the RPi to connect to the TEM, then the RPi external address will need to be on a different domain, for example 192.168.2.80.

swathRT systems generally use fixed IP addressing without a DHCP server. It may be necessary to force the RPi to not use DHCP, as follows:

- In a terminal window; edit /etc/dhcpcd.conf
- sudo nano /etc/dhcpcd.conf
- At the end, add:
  # fallback to static profile on eth0
  interface eth0
  fallback static_eth0
  static ip_address=192.168.0.80/24
  static domain_name_servers=
- Save and exit
- Reboot

3.2.5 Ethernet Ports

The Raspberry Pi 4 Model B has one Ethernet port. To connect it to both the Bathyswath TEM and a cable to connect to the rest of the system, an Ethernet switch is needed. However, an alternative is to fit a USB-to-Ethernet converter to one of the RPi's USB ports, and so have two different Ethernets, which will have different IP domains. It works best if the Ethernet-to-USB is used to connect to the TEM,

3.2.6 Typical IP address map

At typical IP address map for a Bathyswath RPi system is:

- RPi: 192.168.0.80
- PC: 192.168.0.70
- Motion & position sensor: 192.168.0.60
- TEM: 192.168.0.240

3.2.7 Typical IP address map, two Ethernet ports on RPi

If two Ethernet ports are used (see above), then the IP address map could be:

- TEM: 192.168.2.240
- RPi (to TEM): 192.168.2.80
- RPi (to PC): 192.168.0.80
- PC: 192.168.0.70
- Motion & position sensor: 192.168.0.60

3.2.8 SSH

You can run commands on the RPi using an SSH connection from a Windows laptop. Windows 10 provides ssh via cmd windows.

To enable ssh on the Raspberry Pi:

- On the Raspberry Pi screen, start a Terminal (AKA shell, command prompt)
- Enter "sudo raspi-config"
- Go down to Interfacing Options
- Go to SSH
- "Would you like the SSH server to be enabled?" > Yes
- <Finish>
- Reboot
- Follow instructions to set a new "pi" user password, if you want.
  - The default password for user "pi" is "raspberry"
  - To change it, open a Terminal window and enter "passwd"
  - Set a new password, e.g. "swath01"
- You can now transfer files using FileZilla (see section 3.4.2), or log in remotely using ssh or PuTTY (see section 3.4.3).

3.2.9 VNC

You can use VNC to log in remotely using a copy of the RPi's screen on the laptop:

- On the RPi:
  - Enter `sudo raspi-config`
  - Go down to **Interfacing Options**
  - Go to **VNC**

- "Would you like the VNC server to be enabled?" > Yes
- <Finish>
- Reboot

- On the Windows PC: See section 3.4.6.

3.2.10 Qt Libraries

swathRT is compiled using the Qt system, and so requires the Qt libraries to run. Install them using

```
sudo apt install qt5-default libqt5serialport5 libqt5serialport5-dev libatomic1
```

Alternatively, install the Qt Creator program; see section 5.9.

3.2.11 Screen Resolution in Headless Mode

When there is no HDMI monitor connected, we can connect to the RPi GUI using VNC (see below). But with no monitor, the RPi defaults to 640x640 pixels, which is too small to do anything sensible. So, we can change the default setting. Some systems don't show anything on VNC unless either this is set, or a monitor is plugged in to the Raspberry Pi.

- In a terminal window, use `sudo raspi-config`.
- Go to the **Advanced Options** setting, then **Resolution.**
- Select a suitable resolution for the screen that you are using.
- Allow the reboot.

This page gives instructions:

https://support.realvnc.com/knowledgebase/article/View/523/2/troubleshooting-vnc-server-on-the-raspberry-pi

You can also set it in the config file, as follows.
In a Terminal window:

```
sudo nano /boot/config.txt
```

After the `hdmi_` settings section, add:

```
# Force a larger display size; otherwise we only get 640x640 pixels in VNC
hdmi_ignore_edid=0xa5000080
hdmi_group=2          # specifies groups of settings for hdmi_mode
hdmi_mode=23          # 1280x768
```

or

```
hdmi_mode=16          # 1024x768
```

or

```
hdmi_mode=35          # 1280x1024, 60Hz
```

see https://www.raspberrypi.org/documentation/configuration/config-txt/video.md

More instructions here.

3.2.12 USB Serial Port

USB-to-serial adaptors can be used with RPi. Connect the adaptor to the RPi, and use a terminal to check that it is seen by Raspbian. `ls /dev/tty*` should include `/dev/ttyUSB0` (etc.)

3.2.13 Terminal Emulator

You can use a terminal emulator to see the inputs to the serial port and to type outputs to it.

Emulators suggested include **screen** and **minicom**.

Install with `sudo apt-get install screen`

To use, type

```
sudo screen /dev/ttyUSB0 38400
```

… for the first USB-serial converter running at 38400 baud.

Use **ctrl-A d** keys to stop.

3.2.14 Bridging the RPi Ethernet Ports for Testing

If a separate Ethernet port is used for the TEM, using a USB-Ethernet adaptor, it can sometimes be useful to bridge the two RPi Ethernet ports, so that the TEM Ethernet port is connected directly to the external PC, instead of the RPi second port. This allows you to run Windows software that connects to the TEM, from the ITER Systems Bathyswath software suite.

You can do that as follows:

- Install the `bridge-utils` package:
  - sudo apt-get install bridge-utils
- Add a bridge
  - `brctl addbr br0`
- Add the two Ethernet ports to the bridge
  - `sudo brctl addif br0 eth0`
  - `sudo brctl addif br0 eth1`
- Configure the bridge to an IP address (the example below uses 192.168.16.29)
  - `sudo ifconfig br0 netmask 255.255.255.0 192.168.16.29 up`

This setting does not persist over a reboot, so rebooting or power-cycling returns to the original setting.

More instructions here.

3.2.15 Setting up a GSM dongle

GSM dongles are a good way of connecting to the internet over GSM (mobile phone) signal.

Devices that work with Raspberry Pi include Huawei 3372 and Huawei 303.

For our tests, a Huawei Unlocked E3372- LTE/4G 150 Mbps USB Dongle (the dongle) was used, fitted with an EE PAYG Triple SIM Card Preloaded with 6 GB of 4GEE Data.

Plug the dongle into the RPi. After a few seconds it should enumerate as an ethernet adapter with the interface name `eth1`.

It is prioritised over Wi-Fi as the default interface, as is the case with the interfaces provided by the GSM Hat (above). Therefore, no further configuration is needed.

However, if the dongle doesn't seem to be the default interface, then you can use the following commands to edit the routing table:

```
sudo route del default
sudo route add default eth1
```

If you then run `route -n` ; you should see a list of interfaces, with `eth1` having the lowest value under the `metric` column, meaning it has the highest priority when trying to route data to the internet.

The device is usually recognised by the RPi when you plug it in. To check that, enter
`lsusb`

You should see something like

```
Bus 001 Device 006: ID 12d1:14dc Huawei Technologies Co., Ltd. E33372 LTE/UMTS/GSM HiLink Modem/Networkcard
```

… amongst other entries

If not, you might need to install the driver:
`sudo apt-get install usb-modeswitch –yes`

then reboot.

You should be able to see the settings for the GSM dongle by finding the IP address that it has been given, using ipconfig:

```
eth1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
      inet 192.168.8.100  netmask 255.255.255.0  broadcast 192.168.8.255
```

and then using the web browser to go to the root of that address, in this case 192.168.8.1. This will open (for our dongle) a Huawei browser page.

3.2.16 Setting up the Data SIM Card

The SIM card used in the dongle or in the GSM HAT runs out when the data is all used up or if the time period has expired.

The test system is using an EE data SIM card for use in the UK. For other countries, it makes sense to purchase a local data SIM card.

With EE, to top up the data allowance

- Set up a “My EE” account associated with the SIM card at https://myaccount.ee.co.uk/app
- Log in to the account associated with the SIM card
- Top up the amount of money on the card, by at least the price of the data pack required
- Then, purchase a data pack

It is theoretically possible to set up and manage multiple SIM cards through the same My EE account.

More instructions here.

3.3 INSTALLING SWATHRT ON LINUX

3.3.1 Directory structure

The swathRT programs do not enforce any particular directory structure, but these instructions use the following. swathRT_cmd also defaults to an assumption of this structure.

- Home directory for user **pi**:
  `/home/pi`
- Top level for swathRT apps:
  `/home/pi/swathRT`
- Deployment of executable files:
  `/home/pi/swathRT/swathRT_deploy`

3.3.2 Copying and deploying executable files

The swathRT executable binary is available from ITER Systems, and from the swathRT GitHub repository [Ref 6]; contact support@iter-systems.com to request access.

To deploy swathRT on Linux systems:

- Copy the release executable to a suitable folder for deployment (e.g. `/home/pi/swathRT/swathRT_deploy`)
- It might be necessary to make the swathRT file executable, using
  `sudo chmod +x swathRT`
- Copy RTsettings.ini to the same location (see Annex A). Alternatively, swathRT will use default user settings and create a new ini file the first time that it closes.
- In a terminal window, simply call the exe to run it, e.g.
  `/home/pi/swathRT/swathRT_deploy/swathRT` (if you run from the same directory, you can call `./swathRT;` note the leading "`./`")

3.3.3 Start swathRT at boot

You might want swathRT to start automatically when Linux boots up. It is also useful to run it in a terminal window, so that you can see the messages sent to the terminal (console) by the program. The standard method of using rc.local doesn't work for that, because the X Windows environment that the terminal application uses hasn't started up at that point. Instead, do the following:

```
mkdir /home/pi/.config/autostart
nano /home/pi/.config/autostart/swathRT.desktop
```

The create swathRT.desktop as:

[Desktop Entry]

Version=1.0

Type=Application

Encoding=UTF-8

Name=swathRT

Comment=run swathRT

Exec=xterm -fs 15 -hold -e /home/pi/swathRT/swathRT_deploy/swathRT

Terminal=true

Categories=None

- Edit `/home/pi/.config/lxsession/LXDE-pi/autostart`
- Add, at the end of the file:
  `@sudo lxterminal -e /home/pi/swathRT/swathRT_deploy/swathRT`
- Save the file, then reboot to test.
- However, this doesn't seem to apply to later versions of Raspbian

Alternatively, to use rc.local and so not use the XTerminal console window:

- Edit rc.local
  `sudo nano /etc/rc.local`
- Towards the end, but before "exit 0", add
  `/home/pi/Swath/swathRT_deploy/swathRT &`
  Note the '&' at the end; without it, the boot sequence won't end until swathRT ends.

3.3.4 Create a shortcut to run swathRT on the desktop

To create a shortcut on the RPi desktop that can be used to run swathRT with a double-click, create a .desktop file on the desktop.

cd to /home/pi/Desktop, and use a text editor called "swathRT.desktop", containing:

```
[Desktop Entry]
Version=1.0
Type=Application
Encoding=UTF-8
Name=swathRT
Comment=run swathRT
Exec=/home/pi/swathRT/swathRT_deploy/swathRT
Terminal=true
Categories=None
```

3.3.5 Running swathRT_cmd on the Linux system

It can also be useful to run the control dialog, swathRT_cmd, on the Linux system. Use the instructions in 3.3.2 and 3.3.4, substituting "swathRT_cmd" for "swathRT" where it appears.

3.4 CONFIGURE A WINDOWS LAPTOP TO CONNECT TO RPI

3.4.1 IP Address

Set the Windows laptop to use fixed IP addressing. The default address is 192.168.0.70. Use **Start**, then select **Settings** > **Network & Internet** > **Ethernet**.

3.4.2 FileZilla

Install and set up FileZilla (or a similar FTP client) to transfer files between the Windows and Linux computers:

- Use Site Manager to create a new site:
  - Host: 192.168.0.80
  - Port: can leave blank
  - Protocol: SFTP
  - Logon: normal
  - User: pi
  - Password: swath01

3.4.3 ssh and PuTTY

You can log in remotely to the Linux system using **ssh** from a Windows **cmd** window, using, for example:

```
ssh pi@192.168.0.80,
```

or

```
ssh pi@raspberrypi
```

then enter the password.

Windows 10 systems include ssh; earlier versions might not, in which case you can use the PuTTY SSH client, which can be downloaded from www.putty.org. PuTTY is also needed for some swathRT_cmd functions: see below.

It might also be necessary to run PuTTY once on a newly-connected system to allow it to update its cached connection credentials. Start PuTTY, and enter the RPi IP address (e.g. “192.168.0.80”) in Host Name. The default port of 22 works. Select SSH, then Open. If the system warns that configuration parameters have changed, accept the new ones. RT command actions such as setting the RPi time should now work.

3.4.4 Controlling the Linux Operating System from swathRT_cmd

Some of the command buttons provided by swathRT_cmd make changes to the remote Linux that need superuser permission. These need PuTTY to be installed, from www.putty.org, to use the “plink” tool.

The user name and password for a user in the Linux system that has superuser rights must be entered correctly in swathRT_cmd.

3.4.5 ssh key Regeneration

If you are using more than one RPi with the same IP address, you will need to regenerate the ssh key on the Windows computer, otherwise you will get an error of the form `WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!`
in a Windows cmd window, use
`ssh-keygen -R 192.168.0.80` (or the IP address that you are using for the RPi).

You need to do this for VNC to work, too. It is also needed for some of the command buttons in swathRT_cmd.

3.4.6 VNC

You can use VNC to log in remotely using a copy of the Linux screen on the laptop; see 3.2.9 for the Linux setup.

- Download a VNC viewer, e.g. https://www.realvnc.com/download/vnc/windows/, and install it.
- Start the VNC Viewer (the Server needs a license, the Viewer does not):
  - Log in as 192.168.0.80, user "pi", password "raspberry" (or "swath01"),
  - That should give you a copy of the RPi screen.

3.4.7 Setting the Raspberry Pi clock from Windows

Raspberry Pi does not include a battery-backed real-time clock, so its clock needs to be set each time the RPi status up. If it is connected to the Internet, it does that automatically from Internet time servers. But if there is no Internet connection, then the time must be set another way. Time is needed in swathRT to help synchronise data inputs, when using "PC clock" mode in Swath Processor (although all inputs are time-stamped with the same clock), and for setting the names of data files.

Time can be set by one of several methods:

- Use the **Set remote time** button in **swathRT_cmd.**
  Note that this needs the correct **Target user** and **password** to be set in the swathRT_cmd window.
- Manually on the RPi desktop using VNC
  - Open a terminal window
  - Use the date command. This accepts a wide range of date formats, for example "YYMMDD HH:MM". Use -s to set the date and time; it must be run as superuser:
    `sudo date -s "200319 10:44"`
- Manually through ssh in a Windows cmd window
  - Open a Windows cmd window
  - Enter `ssh pi@192.168.0.80` or `ssh pi@raspberrypi`
  - Enter the password for pi
  - Enter the `date` command as above
- Set up an NTP server on the Windows computer, and set up the RPi to receive times from it.

3.5 INSTALLING SOFTWARE ON WINDOWS

3.5.1 Installing the swathRT software on Windows

An installation application, **swathRT_Windows.msi**, is available to install the swathRT software (swathRT, simBSW, SwathDisplay and swathRT_cmd) on a Windows system. Simply double-click on swathRT_Windows.msi and follow the instructions. Icons for the programs will appear on your desktop: double-click on them to run the programs.

3.5.2 Installing Swath Processor

The Bathyswath Swath Processor application is used to control and monitor Bathyswath systems when they are connected directly to Windows computers. It can also be used to monitor swathRT running on a remote Linux computer.

To install it, simply double-click on the bathyswath.msi installation application, which is available from ITER Systems.

To configure Swath Processor to receive data from swathRT, see 5.2.4.

4 INTERFACING TO SWATHRT

4.1 INTERFACES

swathRT can be configured to interface with external systems using UDP packets over Ethernet and/or serial ports.

4.2 INPUTS

swathRT receives command messages to control it and data from motion, position, sound velocity and other auxiliary sensors.

4.3 OUTPUTS

swathRT can output data as UDP packets, as well as saving data to file on the local computer disk. These packets can be downsampled for use with low-capacity communications links.

The Bathyswath sonar data and auxiliary sensor data can be saved to file or output as either Bathyswath raw data (as in .sxr files) [Ref 2] or parsed data (.sxi) format [Ref 2 and Ref 3]. The parsed data format sonar data is in range-angle-amplitude form, and so is more useful if a vehicle needs the data for onboard processing.

4.4 CONFIGURATION

The communications ports are configured using the `RTSettings.ini` file. See Annex A.

4.4.1 External UDP communications configuration

Configure the UDP communications port for connection to the operator's computer using the following ini settings:

```
[GeneralSettings]
commandsPort=52763              UDP port that command packets are sent to
receiverIP=192.168.0.70         IP address that sonar data from swathRT is sent to
receiverPort=52764              UDP port that sonar data from swathRT is sent to
auxSendDecimation=1             Send 1 in N attitude and position samples over UDP link
pingSendDecimation=1            Send 1 in N pings over UDP link (use odd number)
reducePingSizeForUDP=1          Decimate pings sent over UDP to 1 in N sonar samples
UDPPingDelay_ms=20              Delay between sending pings over UDP, ms
```

The external control system, e.g. the vehicle's onboard control system or control over a radio or Wi-Fi link, should be set up to send to the port specified in `commandsPort` and the IP address of the computer module that swathRT runs on.

If data is sent to another program for further processing (for example Swath Processor, see 3.5.2), then `receiverIP` should be set to the IP address of the computer that is running it, or to `127.0.0.1` ("localhost") if it is running on the same computer as swathRT.

The external program should be set to receive UDP data from the port specified in `receiverPort`.

If the communications channel has limited bandwidth, then sonar and auxiliary sensor data can be sub-sampled for sending using `pingSendDecimation` and `auxSendDecimation` respectively. `pingSendDecimation` should usually be *set to an odd number*: if you have two sonar channels enabled (port and starboard), then an even number will only send one side (port or starboard). You can also reduce the size of the sonar pings that are sent using `reducePingSizeForUDP`. Pings written to file on the Linux system are not affected by this setting.

4.4.2 Auxiliary sensor data input and output

Auxiliary sensor data can be received from the commands port (`commandsPort`), the auxiliary sensor ports (`auxUDP_port_*`) and the serial ports (`serialPortName_*`).

Auxiliary sensor data packets are recognised by unique headers in each packet. When a packet is received and classified as a sensor data packet, then the following things happen:

- The data is packaged into a Bathyswath raw data attitude string data block (MRUT_DATA) or position string data block (GPST_DATA).
- The data bock is written to file, output over the UDP port, or both, according to the current settings and commands.
- In addition, $GPZDA messages are decoded and the time is sent to the Bathyswath electronics to set its internal clock. That clock is further refined by PPS timing pulses received directly on the TEM (electronics module) PPS input connector. See below.

Sensor data packet types currently recognised are listed below. New ones can be added using `attitudeStrings` and `positionStrings`.

Packet starts with	Data type
$GPZDA	Timing message, NMEA 0183 format
$GPGGA	Position data, NMEA 0183 format
$PASHR	Motion data, NMEA 0183 format
$GPHDT	Heading data, NMEA 0183 format
$PSAT	Attitude data, NMEA 0183 format
Binary bytes 0xFF, 0x5A	SBG-systems INS data, SBG binary format
Text strings, char 1 ':', char 8 ' '	TSS1 attitude messages

4.4.3 Auxiliary data UDP communications configuration

The onboard sensors, e.g. the vehicle's motion and position sensors, should be set up to send to the port specified in `auxUDP_port_1`, (or one of the others, _2, _3) and the IP address of the computer that swathRT runs on. swathRT will accept data from any of the three UDP ports. Set unused ports to 0, otherwise an error message will be shown when swathRT starts or the ini file is re-read.

```
[AuxDataSettings]
auxUDP_port_1=4000          Receiving on port 4000
auxUDP_port_2=5000          Receiving on port 5000
auxUDP_port_3=0             The third port is disabled (port set to 0)
```

When data is received on any of these ports, it is inspected and marked as either attitude data or position data by looking for data strings from comma-separated lists, as follows:

```
attitudeStrings="$PASHR,$PSAT,:,\xfffdZ\xfffd"        Headers of attitude data packets
positionStrings="$GPGGA,$GPHDT,$GNGGA,$GNHDT"         Headers of position data packets
```

The "\xfffdZ\xfffd" sequence specifies two binary bytes, in this case the header of the SBG attitude data packets.

4.4.4 Auxiliary data Serial Port configuration

Auxiliary data can also be received on serial ports. As with the UDP ports, data is received on any of the specified ports, and assigned to attitude or position data streams using the header strings.

Configure the input serial ports using the following ini settings:

```
useSerialPort_1=true              Enable serial port 1
useSerialPort_2=false             Enable serial port 2
useSerialPort_3=false             Enable serial port 3
serialPortName_1=/dev/ttyUSB0     Device name of serial port 1
serialPortName_2=/dev/ttyUSB1     Device name of serial port 2
serialPortName_3=/dev/ttyUSB2     Device name of serial port 3
serialPortBaudRate_1=19200        Baud rate of serial port 1
serialPortBaudRate_2=19200        Baud rate of serial port 2
serialPortBaudRate_3=19200        Baud rate of serial port 3
```

4.4.5 Reporting auxiliary data to the terminal

The auxiliary data strings can be reported to the swathRT terminal window, if required for test and diagnosis:

```
reportAuxData=false               Report the raw auxiliary data strings to terminal
reportAttitude=false              Report the parsed attitude strings to terminal
```

Generally, the auxiliary data strings are not parsed, and the raw strings are stored to file and/or forwarded to the external UDP port.

4.4.6 Calculating range and angle

Use

```
[ProcessingSettings]
computeRangeAngle=true            calculate range and angle from phase data
```

4.4.7 Filtering range and angle

Range and angle data can be filtered, to remove items that are outside acceptable limits:

```
doAmplitudeFilter=true            Filter data by amplitude
minAmplitude=500                  Remove data items less than this amplitude
doRangeFilter=false               Filter by range
maxRange=40                       Remove items more than this range (metres)
minRange=5                        Remove items less than this range (metres)
```

```
doPhaseConfidenceFilter=false     Compare angle from various phase calcs.
phaseConfidenceLevel=60           Remove items that disagree more; percentage
```

4.4.8 Output data format

To control the format of the sonar data being output from swathRT, use:

`outputType=SONAR_DATA4` *For raw data (sxr), or*

`outputType=PARSED_PING_DATA` `PARSED_PING_DATA` *for parsed (sxi)*

to control the format of the sonar data being output from swathRT. The `PARSED_PING_DATA` data needs `computeRangeAngle=true` to be set.

4.4.9 Storing RINEX data files

RINEX files are used for postprocessing GNSS data. They are binary format files that are stored in real-time, and uploaded to postprocessing tools to obtain better-quality GNSS position data.

Some GNSS systems (e.g. Trimble) allow RINEX files to be stored in the device or a connected computer using the system's own configuration software. However, for example, the Hemisphere configuration software (PocketMax) does not support saving RINEX files on Linux systems. In this case, these files can be collected using swathRT.

RINEX files can be stored from swathRT by configuring the RTsettings.ini file as follows:

- writeRinexFile=true
- rinexFileFreq=1 *(1 Hz collection rate: this is the default for RINEX)*
- hsRinexIDs="Bin65,Bin66,Bin76,Bin94,Bin95,Bin96," *(these are the packet types used for collecting RINEX files from Hemisphere GNSS systems: other GNSS systems may use different packet types)*

The RINEX files are stored continuously as soon as the configurations above are detected; the start-stop controls used for reading sonar data are not used in this case. This is because a continuous record of raw GNSS data is needed for data processing.

4.4.10 Input command format

Commands can be received from the commands port (`commandsPort`), the auxiliary sensor port (`attPosPort`) and the serial port (`serialPortName`).

The following messages can be received:

Message	Bathyswath Actions
$PMISS,LINE_START	Start pinging and writing to file
$PMISS,LINE_END	Stop pinging and writing to file
$SWPCT,SNR,RNG, *range*	Sonar control: set ping range in metres
$SWPCT,SNR,PLS, *pulse*	Sonar control: set pulse length in cycles, 2 to 250
$SWPCT,FILE,WRITE,ON	Start writing data to file
$SWPCT,FILE,WRITE,OFF	Stop writing data to file
$SWPCT,UDP,SEND,ON	Start outputting data on the UDP port
$SWPCT,UDP,SEND,OFF	Stop outputting data on the UDP port
$SWPCT,SNR,TX,ON	Turn sonar transmit on

Message	Bathyswath Actions
$SWPCT,SNR,TX,OFF	Turn sonar transmit off
$SWPCT,FILE,FLDR,*folder_path*	Set the folder where raw data files are stored
$SWPCT,READ_SETTINGS	Re-read the .ini settings file
$SWPCT,FILE,POSTSC,*file_postscript*	Set a postscript for file names (text added to the end of each file name)
$SWPCT,SNR,RESET	Reset the swathRT connection to the sonar
$SWPCT,KILL	Shut down the application
$SWPCT,SHUTDOWN	Shut down the Linux operating system

4.4.11 Sonar data timing

Data packets from the Bathyswath TEM are time-stamped using the TEM's internal clock. In order that these data packets can be related to motion and position data for post-processing, the TEM's clock needs to be synchronised with an external time source. This is done using both of two inputs:

- $GPZDA NMEA 0183 messages sent to swathRT (see above)
- PPS timing pulses sent directly to the TEM.

Use of PPS signals is controlled using the following entries in RTsettings.ini:

`PPSOn=false`	*Enable PPS timing pulse input to the TEM*
`PPSRisingEdge=false`	*PPS pulses are valid on the rising edge (else falling edge)*
`PPSSource=0`	*Specifies the port used in the TEM for PPS pulse input (see 5.3.5)*
`PPSWindow=2`	*Window around 1s in microseconds for PPS pulses to be accepted*

See the OEM Integration Manual [Ref 4] for guidance on connecting PPS signals to the TEM. "TEM IO Connector" gives the pinout of the connector used, and "PPS" describes the options. There are four input channel options. The default is "GPI1", specified by "`PPSSource=0`" and using pin 23 on the TEM I/O connector.

5 USING SWATHRT

5.1 STARTING THE SOFTWARE

On Linux, simply call the exe from a terminal window, e.g. cd to the directory holding the exe files, and enter **`./swathRT.`**
(note the leading './')

On Windows, double-click on the appropriate .exe file (e.g. swathRT.exe) in the distribution directory, or use a shortcut on the desktop or Start menu from it.

swathRT can also be configured to start automatically when the Linux computer starts up; see section 3.3.3.

5.2 CONFIGURING THE SOFTWARE

5.2.1 swathRT

swathRT is configured and controlled in two ways:

- UDP packet commands, which may come from swathRT_cmd or from another controlling program, e.g. the control software of an autonomous vehicle.
- An ini file, which is read when the program starts to set up its user settings. A UDP command can be sent to swathRT to tell it to re-read the ini file, so swathRT can be controlled in real time by editing the ini file, and then sending the re-read-ini-file command with swathRT_cmd. See Annex A.
  Note that where settings are set from both (as for ping range), each method will over-write the setting of the other.

5.2.2 swathRT communications to external programs

The RTsettings.ini file can be configured to receive from swathRT_cmd and send to swathDisplay or Swath Processor on a UDP port. By default, these are:

```
receiverPort=52764
commandsPort=52763          UDP port that command packets are sent to
```

It must also be configured to send data to the correct IP address (see section 3.2.4 for a suggested IP addressing scheme):

```
receiverIP=192.168.0.70     IP address that sonar data from swathRT is sent to
```

When using swathRT on a remote vehicle, collecting the raw data onboard and monitoring it remotely over a radio link, it is usually necessary to down-sample the data that is sent over the link, in order to keep within the bandwidth of the link. To do this, for example to send 1 in 10 pings and 1 in 5 auxiliary data strings, set:

```
pingSendDecimation=10       Send 1 in N pings over UDP link (use odd number)
reducePingSizeForUDP=1      Decimate pings sent over UDP to 1 in N sonar samples
auxSendDecimation=5         Send 1 in N attitude and position samples over UDP link
```

5.2.3 swathRT_cmd and swathDisplay

These two programs have graphical user interfaces (GUIs), so the programs are configured using the buttons, edit windows, etc. in the GUIs.

5.2.4 Swath Processor

Use the Socket 1 button in the main dialog bar on the left of the screen to open the Socket 1 dialog, and set:

- Enable on
- Receive on
- UDP/IP on

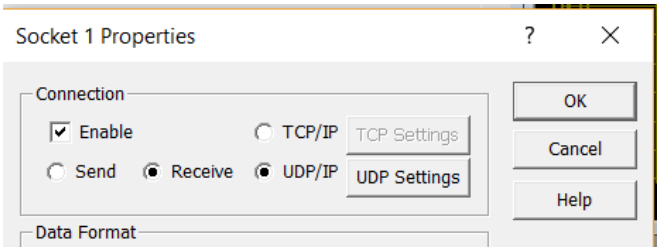

Figure 1 Socket settings for use with swathRT

In the UDP Settings dialog, set:

- Broadcast
- Port = 52764 (this should match the `receiverPort` setting in RTsettings.ini; see above)

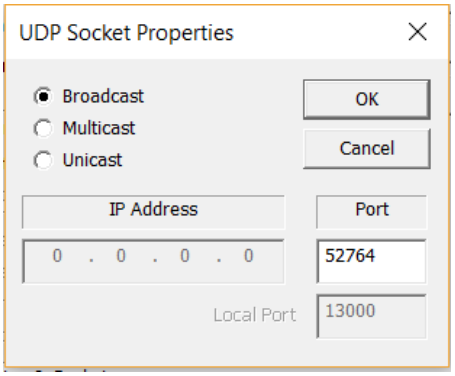

Figure 2 UDP Socket Properties for use with swathRT

5.2.5 Firewalls

UDP transfers will be blocked by firewall tools unless the applications are excluded in the firewall settings.

In Windows 10, use “Windows Defender with Advanced Security”

- Inbound Rules
- New Rule
- Protocols and ports
  - Protocol type: UDP
  - Local port
    - Specific ports
    - 52764
- General
  - Name, e.g. “swathRT incoming port”

- Action
  - Allow the connection

5.3 USING THE SOFTWARE

5.3.1 swathRT_cmd

swathRT_cmd sends commands to swathRT running on a remote computer over a UDP link.

To use swathRT_cmd:

- Enter the IP address of the computer running swathRT in **Target IP Addr**
  - To run on the same computer, use 127.0.0.1
  - Preset buttons are provided for common addresses
- **Start** to send a start pinging message to swathRT
- **Stop** to send a stop pinging message to swathRT
- **Reset** to send a reset sonar message: this causes swathRT to search for a TEM
- **Kill swathRT** to send a shutdown message to swathRT
- **Shutdown** to send a message to swathRT that makes it shut down the operating system on the remote computer
- **Re-read settings** to tell swathRT to re-read the ini file. Note that this will over-ride the settings in the swathRT_cmd window.
- **Ping Range** to set the range of sonar pings in metres. Enter a number or use the up-down arrows.
- **Tx On** to enable transmit pulses with swathRT
- **Stop** to send a stop pinging message to swathRT
- **Send UDP** to make swathRT send sonar data out by UDP
- **Write to File** to make swathRT write sonar data to file
- **New Line** starts the sonar and starts a new data file
- **File Folder** to set the name of the directory that swathRT writes data files to. File names are generated from date and time.
- **File Postfix** adds text to the end of every file name created. This can help you to identify sub-areas of your survey. E.g. if you enter "_area1", then files are written with names like "20190510_003038_area1.sxr".
- **Target password** specifies the password for the remote system. This defaults to "swath01". It is needed for functions that need superuser permission, such as **Start swathRT**, **Shutdown OS** and **Set remote time**.

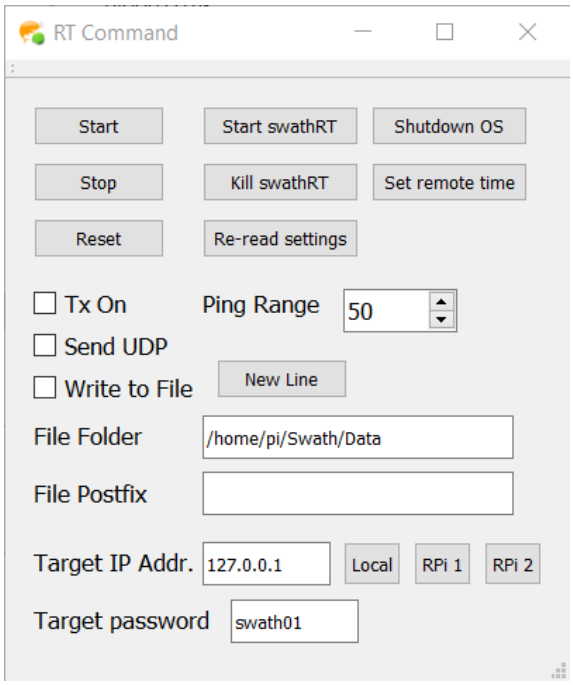

Figure 3 The swathRT_cmd screen

Some of the buttons use the **plink** command to send commands to the Raspberry Pi. This needs the ssh link to be fully configured. If you change the connection to the RPi, e.g. its IP address, you may need to update the ssh interface. See section 3.4.3.

swathRT_cmd has its own ini file, which is stored in the same location as the executable file. This allows settings entered into the swathRT_cmd window to persist between sessions. You can also edit the ini file directly if you wish.

5.3.2 SwathDisplay

swathDisplay receives data from swathRT over a UDP link, and shows parts of the data. It is mostly intended for test and diagnostic purposes. For example, it confirms that the sonar is running, and it can be used to “rub test” the sonar on a remote vehicle before deploying the vehicle.

For operational use on Windows, Swath Processor is generally more useful for monitoring the data collection by swathRT. See section 5.3.4.

swathDisplay is also available for Linux systems.

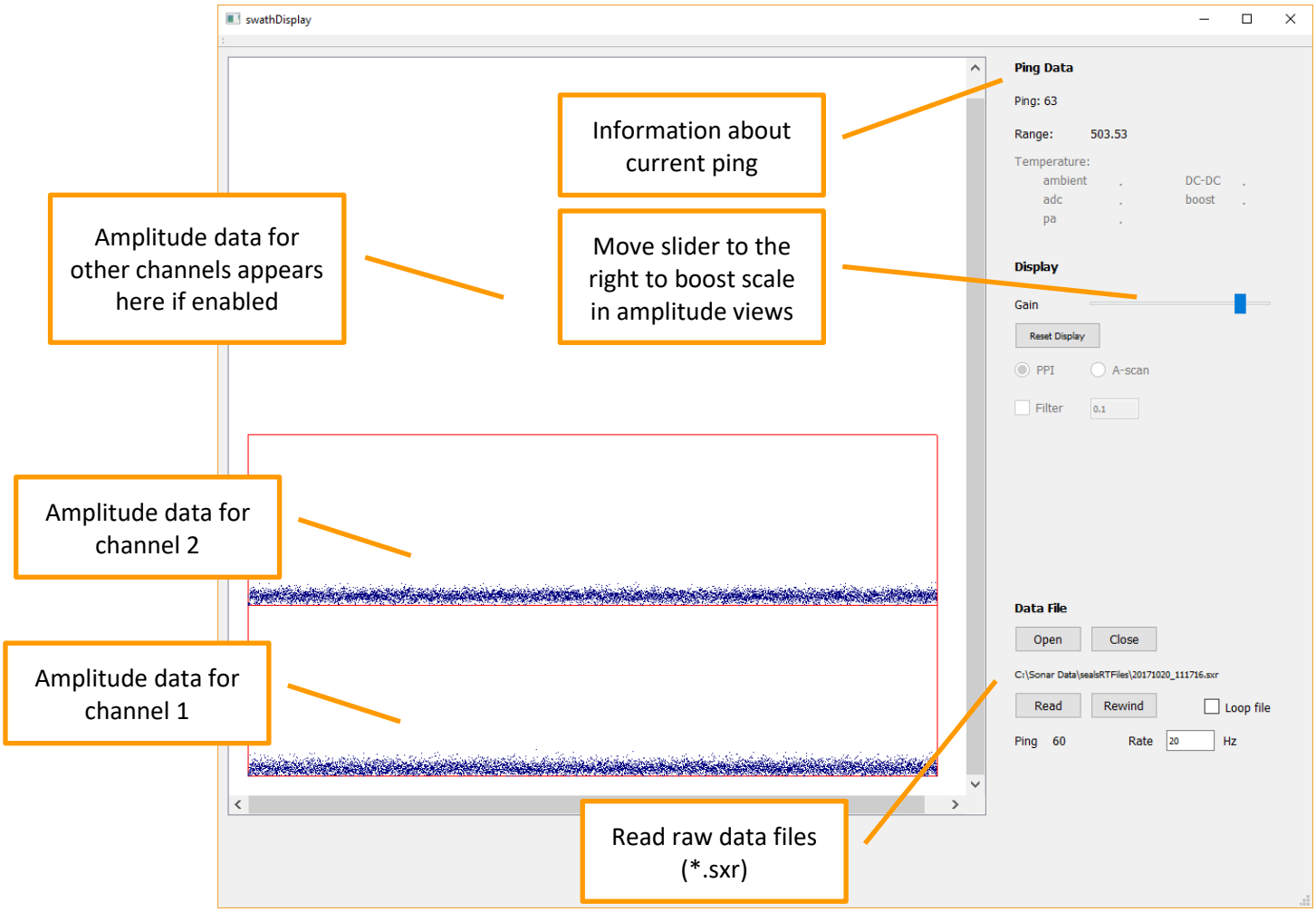

Figure 4 SwathDisplay screen

Enable UDP output from swathRT, using the swathRT_cmd tick-box, to send sonar data to swathDisplay.

5.3.3 Rub Test

As an example of the use of SwathDisplay, to check that the sonar is operating correctly on a remote vehicle before you deploy it:

- Start swathRT on the vehicle.
- Start swathRT_cmd and SwathDisplay on a Windows computer (usually a laptop) connected to the vehicle system with an IP connection (Ethernet, Wi-Fi, etc.).
- On the remote system, edit the swathRT.ini file to set the `receiverIP` to the IP address of the local Windows computer (your laptop), and use swathRT_cmd to get swathRT to re-read the ini file.
- Use swathRT_cmd to start the sonar system and to enable UDP messages (section 5.3)
- SwathDisplay should show the ping number changing, etc.
- Rub the port transducer with your hand: the amplitude data for channel 1 in the SwathDisplay window should show an increasing signal.

- Repeat for the starboard transducer: the channel 2 amplitudes should change.
- Use swathRT_cmd to turn on the sonar transmit signal (**Tx On**). The amplitude display should show “spikes” at the start of the displays (left-hand side).

5.3.4 Swath Processor

Use Swath Processor as you would for a “Windows-only” Bathyswath system. See the Getting Started manual [Ref 1] and the context-sensitive held system for Swath Processor; press the **F1 key** on the keyboard to see help information for the window that is currently highlighted.

Annex A SWATHRT INI FILE

The swathRT ini file is called `RTSettings.ini`. It is in the same directory as `swathRT.exe`.

swathRT creates a new ini file from its current set of settings when it shuts down. So, if the ini file is lost or corrupted, you can create a new one by running swathRT and then sending it a "Kill" message from swathRT_cmd.

The ini file is a simple text file. Each setting is on a separate line, with the form

`setting=value`

It is divided into sections, to make the settings easier to find, with each section header in square brackets, e.g.

`[GeneralSettings]`

Some of the settings in the file are not yet implemented in the program, and new settings may be added or existing ones removed in the future.

A typical RTSettings.ini file is shown below. Comments have been added here in italics: these are not present in the actual ini file. "On/off" settings are controlled by entering "false" or 6

`[GeneralSettings]`	*Section header: general settings for the program*
`commandsPort=52763`	*UDP port that command packets are sent to*
`receiverIP=192.168.0.70`	*IP address that sonar data from swathRT is sent to*[1]
`receiverPort=52764`	*UDP port that sonar data from swathRT is sent to*
`UDPPingDelay_ms=20`	*Delay between sending pings over UDP, ms*
`auxSendDecimation=1`	*Send 1 in N attitude and position samples over UDP link*
`pingSendDecimation=1`	*Send 1 in N pings over UDP link (use odd number)*
`reducePingSizeForUDP=1`	*Decimate pings sent over UDP to 1 in N sonar samples*
`rawFileFolder=/home/pi/Swath/Data`	*The folder on the system where raw data is stored*
`outputType=PARSED_PING_DATA`	*Output data format:* `PARSED_PING_DATA` *for parsed (sxi),* `SONAR_DATA4` *for raw (sxr)*
`startSonarAtStart=false`	*Start the sonar when the program starts*
`size=0`	*Terminator string for the section; do not edit*
`[AuxDataSettings]`	
`auxUDP_port_1=4000`	*Receiving on port 4000*
`auxUDP_port_2=5000`	*Receiving on port 5000*
`auxUDP_port_3=0`	*The third port is disabled (port 0)*
`useSerialPort_1=true`	*Enable serial port 1*
`useSerialPort_2=false`	*Enable serial port 2*
`useSerialPort_3=false`	*Enable serial port 3*

[1] 127.0.0.1 specifies the local computer ("localhost"), and is used with swathRT and the receiving application are on the same computer

```
serialPortName_1=/dev/ttyUSB0      Device name of serial port 1
serialPortName_2=/dev/ttyUSB1      Device name of serial port 2
serialPortName_3=/dev/ttyUSB2      Device name of serial port 3
serialPortBaudRate_1=19200         Baud rate of serial port 1
serialPortBaudRate_2=19200         Baud rate of serial port 2
serialPortBaudRate_3=19200         Baud rate of serial port 3

attitudeStrings="$PASHR,$PSAT,:,\xfffdZ\xfffd"        Headers of attitude data packets
positionStrings="$GPGGA,$GPHDT,$GNGGA,$GNHDT"         Headers of position data packets
reportAuxData=false                Report the raw auxiliary data strings to terminal
reportAttitude=false               Report the parsed attitude strings to terminal
writeRinexFile=false               Write GNSS data to a Rinex file

reportPingStats=false              If set true, swathRT reports ping stats to the console window
reportAttitude=false               Report attitude information to the console
reportPingStats=false              Print ping statistics to the console
serialPortBaudRate=38400           Baud rate of the serial port, if used

[SonarSettings]                    Section header: settings for the TEM
PPSOn=false                        Enable PPS timing pulse input to the TEM
PPSRisingEdge=false                PPS pulses are valid on the rising edge (else falling edge)
PPSSource=0                        Specifies the port used in the TEM for PPS pulse input (see 5.3.5)
PPSWindow=2                        Time window in us for PPS pulses around 1 s to be accepted
maxPingRate=5                      Maximum number of pings per second
overTemperatureSleep_ms=2000       App sleeps for this time if over-temperature detected
pingTimeout=500                    Pings time out after theoretical period plus this, ms
preampPowerOn=true                 Preamplifier power supplied from TEM to transducer
shutdownTemperature=8000           Sleep if temperature exceeds this, °C. See above.
size=0

[PingSettingsGeneral]              Section header: General settings for pings
alternate=false                    Alternate pings, else fire all chans simultaneously
autoSetPingSettings=true           Compute ping settings automatically, else use fixed
extTrigger=false                   Use an external source to trigger pings
extTriggerInvert=false             Pings triggered on falling edge of ext signal, else rising
extTriggerSource=3                 Specifies the port used for ext trigger input (see 5.3.5)
pingRange=200                      Ping range in metres
rxOn.0=true                        Sonar channel 1 is active if set true
rxOn.1=true                        Sonar channel 2 is active if set true
rxOn.2=false                       Sonar channel 3 is active if set true
sonarFrequency.0=117               Frequency of sonar channel 1 in kHz
sonarFrequency.1=117               Frequency of sonar channel 2 in kHz
sonarFrequency.2=0                 Frequency of sonar channel 3 in kHz
tdcrType.0=16                      Type of transducer connected to sonar channel 1 [5.3.6]
tdcrType.1=16                      Type of transducer connected to sonar channel 2 [5.3.6]
```

tdcrType.2=15	*Type of transducer connected to sonar channel 3 [5.3.6]*
triggerOut=false	*Output a ping trigger signal*
triggerOutInvert=false	*Send low-going trigger signal, else high*
triggerOutSource=3	*Specifies the port used for ext trigger output [5.3.5]*
txOn.0=true	*Output transmit pulse on sonar channel 1*
txOn.1=true	*Output transmit pulse on sonar channel 2*
txOn.2=false	*Output transmit pulse on sonar channel 3*
size=0	
[PingSettingsAuto]	*Section header: Auto settings for pings*
LNA18dBRange=50	*Switch LNA preamp to 18dB at this sonar range*
LNA24dBRange=100	*Switch LNA preamp to 24dB at this sonar range*
MaxTxLen=300	*Maximum number of cycles in the transmit pulse*
MaxTxPow=100	*Maximum transmit power, percent*
MaxTxRange=600	*Range at which max. transmit pulse length is used*
MinTxLen=2	*Minimum number of cycles in the transmit pulse*
MinTxRange=20	*Range at which min. transmit pulse length is used*
TxMaxPowerRange=20	*Range at which min. transmit power is used*
decimationFor100kFilter=1	*Range Decimation used when 100kHz bandpass filter used*
decimationFor10kFilter=6	*Range Decimation used when 10kHz bandpass filter used*
decimationFor30kFilter=4	*Range Decimation used when 30kHz bandpass filter used*
pulseLength100=10	*Use 100kHz bandpass filter at this pulse length and below*
pulseLength30=30	*Use 30kHz bandpass filter at this pulse length and below*
size=0	
[PingSettingsFixed]	*Section header: Fixed settings for pings*
autoAdjustSWGain=true	*Adjust the software gain to keep values within 16-bit limits*
bandwidth=100	*Bandpass filter width, kHz*
basegain=500	*Fixed gain value, dB*100 (500=5dB)*
decimation=1	*Decimation: remove 1 in N samples before sending*
lingain=0	*Linear gain value, dB*100, rising with range*
lnagain=1	*Setting of LNA preamp; 0, 1 or 2*
pgagain=0	*Setting of PGA preamp; 0 or 1*
sqgain=3000	*Square gain value, dB*100, rising with range squared*
transmitPowerAttenuation_dB=15	*Transmit power; 0 max, 15 min.*
txLength=2	*Transmit pulse length, cycles*
size=0	
[ProcessingSettings]	*Section header: Data processing*
computeAmpAltitude=false	*Calculate altitude from amplitude*
ampAltSmoothingWindow=0.01	*smoothing window for altitude calculator*
maxAltRange=100	*maximum range for altitude calculator*
minAltRange=5	*minimum range for altitude calculator*
computeRangeAngle=true	*calculate range and angle from phase data*
readFile=false	*read a raw data file (*.sxr)*

```
readFilename=                    name of the data file to read
size=0

[FilterSettings]                 Filters with range-angle calculation
doAmplitudeFilter=true           Filter data by amplitude
minAmplitude=500                 Remove data items less than this amplitude
doRangeFilter=false              Filter by range
maxRange=40                      Remove items more than this range (metres)
minRange=5                       Remove items less than this range (metres)
doPhaseConfidenceFilter=false    Compare angle from various phase calcs.
phaseConfidenceLevel=60          Remove items that disagree more; percentage
size=0
```

5.3.5 TEM Input and Output Ports

Value	Port
0	RS485 1
1	RS485 2
2	GPI 1
3	GPI 2

5.3.6 Sonar Transducer Types

Value of tdcrType	Transducer type
15	No transducer connected to TEM
10	SWATHplus/Bathyswath 1 117kHz
5	SWATHplus/Bathyswath 1 234kHz
13	SWATHplus/Bathyswath 1 468kHz
16	Bathyswath 2 117kHz
17	Bathyswath 2 234kHz
18	Bathyswath 2 468kHz
19	Seal detector system type 1

Annex B CONFIGURING A RASPBERRY Pi COMPUTE MODULE

5.4 INTRODUCTION

The Bathyswath Comms and PCB Board (CPB) uses an embedded Raspberry Pi Compute Module. This section describes how to set up swathRT on that hardware.

5.5 HARDWARE SETUP

The simplest way to set up a Compute Module is to use a Compute Module IO Board.

- Fit the Compute Module to the Compute Module IO Board.
- Fit the "USB slave boot enable" jumper to the enable "EN" position.
- Connect a USB port on your computer to the USB SLAVE port via a USB to micro-USB cable.
- Apply power through the POWER IN connector, using another USB cable.
- Connect a display to the HDMI port with an HDMI cable.

5.6 SOFTWARE RESOURCES

Go to the Raspberry Pi page "Flashing the Compute Module eMMC", at https://www.raspberrypi.org/documentation/hardware/computemodule/cm-emmc-flashing.md

You also need a program for writing images to SD cards, **balenaEtcher** or **Win32DiskImager**, as described in Installing operating system images using Windows.

Installing the operating system

Install and run the **RPiBoot.exe** Windows installer program as described under Windows installer in the web page above. This creates a drive letter on Windows where the operating system can be installed to.

5.7 INSTALL THE OPERATING SYSTEM

Download a Raspbian operating system from
https://www.raspberrypi.org/downloads/raspbian/

There are three choices:

- **Raspbian Buster Lite**: this is enough for operational systems.
- **Raspbian Buster with desktop**: this will be useful for development systems.
- **Raspbian Buster with desktop and recommended software**: this could be useful for fully-functional development systems, but is not recommended with versions of Compute Modules with small memory installed.

Use **balenaEtcher** or **Win32DiskImager** to flash the downloaded operating system (zip file) to the Raspberry Pi.

When the operating system flash has completed,

- Power down the Compute Module IO Board
- Fit the "USB slave boot enable" jumper to the disable "DIS" position, and remove the connection to the USB SLAVE port
- Connect keyboard and mouse via the main USB port, via a USB hub

- If a “desktop” version of Raspbian was installed, connect interfaces to it using:
  - Connect to the Internet using Ethernet to USB or WiFi to USB converter, using the USB hub.
  - Connect a monitor via the HDMI port.
- Re-apply the power: desktop systems will show the desktop on the monitor.
- Connect to non-desktop Raspbian systems using ssh.
- Continue with the system set-up as described in section 3.2.

Annex C CONFIGURING RASPBERRY Pi FOR DEVELOPMENT WORK

5.8 INTRODUCTION

If you plan to use the Linux system to develop the swathRT software, you will need to configure the QT compiler.

5.9 QT

Install Qt Creator, for software development on the RPi. This is only needed for development and test systems, not for production systems or for surveying.

- Open a Terminal window
- `sudo apt-get install qt5-default`
  - Allow that to finish
- `sudo apt-get install qtcreator`
  - Allow that to finish
- `sudo apt-get install qtdeclarative5-dev`
  - Not sure if this is essential, but it removes a warning “no qmlscene installed” in Qt
- You may also need to install the serialport module, using:
  - `sudo apt-get install libqt5serialport5`
  - `sudo apt-get install libqt5serialport5-dev`
- One addition that you need to make to the standard Qt Creator install: set the compiler:
  - Tools > Options > Build & Run > Kits > Desktop > Compiler > Manage > Add > GCC
  - Apply, OK
- Install cmake:
  - `sudo apt-get install cmake`

5.10 CONFIGURING QT CREATOR

The main task is to add the GCC compiler to the build “kits”. See above. GCC is installed as standard with the RPi Raspbian operating system.

The latest version of Qt Creator supported on Raspbian is considerably older than the latest on Windows.

To set up a debug build:

- In Manage Kits, set “Local PC” as the default
- Configure “Desktop”
  - Device type: Desktop
  - Device: Local PC
  - Compiler: GCC
  - Debugger: System GDB
  - Qt version: Qt 5.xx.yy (select from drop-down list; installed from qt5-default)
- In Build Settings
  - Add build configuration, and call the new one Debug
  - Change the build directory to “…-Debug”

- Click “Run CMake …”,
  - and add the arguments: “-DCMAKE_BUILD_TYPE=Debug”
- In the button in the bottom-left of Qt Creator, change to “swathRT Debug”
- Build and run debug

5.11 SOURCE CODE

Contact support@iter-systems.com to get a copy of the latest source code. This code is company-confidential, and a non-disclosure agreement (NDA) may be required.

Copy the code into the directory structure suggested in section 3.3.1 (or as appropriate for your own directory structure), with the following relative location of directories:

- Top level for swathRT apps:
  `/home/pi/swathRT`
  - swathRT project:
    `/home/pi/swathRT/swathRT`
    - swathRT code (.pro, .cpp, .h)
      `/home/pi/swathRT/swathRT/swathRT`
    - BNO055 handler code:
      `/home/pi/swathRT/BNO055`
  - swathRT_cmd project:
    `/home/pi/swathRT/swathRT_cmd`
    - swathRT_cmd code (.pro, .cpp, .h)
      `/home/pi/swathRT/swathRT_cmd/swathRT_cmd`
  - swathDisplay project:
    `/home/pi/swathRT/swathDisplay`
    - swathRT code (.pro, .cpp, .h)
      `/home/pi/swathRT/swathDisplay/swathDisplay`

Qt Creator will create folders for the output files and executables with a name that matches the build selections made, for example
`/home/pi/swathRT/swathRT/build-swathRT-Desktop-Release`

The “double layer” directory structure suggested above helps to manage the location of these intermediate directories.

Annex D CONFIGURING WINDOWS PC FOR DEVELOPMENT WORK

5.12 INTRODUCTION

Perhaps the simplest way to develop the swathRT software is to use the QT compiler on a Windows computer. When it compiles and runs in Windows, you can transfer it to a target system for final testing (see Annex C) or cross-compile it on Windows for the target (Annex F).

5.13 QT IDE

SwathRT is developed using Qt Creator [Ref 7], using the Qt classes and C++ Standard Template Library. One big advantage of Qt is that it supports compilation over a wide range of operating systems, including Windows and Linux.

5.14 GITHUB SOURCE CONTROL

The swathRT suite is maintained on GitHub [Ref 6]; contact support@iter-systems.com to request access. You will need a GitHub login to access the swathRT code project. An ITER Systems administrator can give you access on request.

When you have access to the GitHub repository, GitHub will prompt you to create a local repository for the code, see for example Ref 8.

5.15 SWATHRT APPLICATIONS

There are several separate programs in the swathRT suite. These are:

- simBSW: a software simulator for Bathyswath systems
- swathDisplay: a simple display program for Bathyswath data collected from swathRT. This is currently mostly useful for testing: in operation, Swath Processor is usually a better choice.
- swathRT: the main data collection program in the suite
- swathRT_cmd: a command tool, providing instructions to swathRT using UDP commands.

5.16 QT CREATOR

We use the free, open-source version of Qt Creator, available from https://www.qt.io/download-open-source.

Each of the applications listed has a .pro project file, which is used to define the set-up of the code for compilation, debug, etc. Open each one in Qt Creator to start a coding or debug session. You may need to configure Qt Creator to suit the setup of your development computer; see §5.10.

Annex E CLONE THE SD CARD

Having set up the RPi system, you can make a clone of the SD card. This is useful for:

- Creating new systems
- Making a backup in case the first SD card is corrupted, etc.

To copy the SD card:

1. On a Windows laptop, install and run Win32 Disk Imager from https://sourceforge.net/projects/win32diskimager/
2. Enter the location and name of the SD card image to save to in “Image File”; e.g. “<location>/170919_RPi_SD.img”
3. Insert the SD card into a reader on the laptop. Many laptops have SD card readers, but you will probably need an adaptor to use the microSD format that RPi uses.
4. Select the drive letter that Windows has allocated to the SD card in “Device”, e.g. “E:”
5. Wait for it to finish.

To write to a new SD card:

1. Run Win32 Disk Imager
2. Insert the SD card to write to (caution: this overwrites the contents of the SD card!)
3. Select an image file, created as above
4. Click “Write”

More instructions here.

Alternatively, use Win32 Disk Imager, in Read mode. Instructions here. You have to use this method for Compute Modules, which don’t have removable SD cards for the operating system.

However:

- The clone file is the same size as the whole SD card, which can fill a lot of disk space on your computer, and
- It isn’t that hard to set up a new system anyway.
- The SD card image could be stored on an SD card or USB drive.

Annex F Cross-Compiling Qt Software for Raspberry Pi

As an alternative to building the swathRT code on the target machine itself (Annex C), you can cross-compile the code on Windows. That is, you can create executable code that runs on another operating system using a Windows computer. This section describes how to do that. We have again used Raspberry Pi as an example, but this method should also work for any other operating system.

These instructions cover the installing the cross-compilation C/C++ toolchain for the RPi; configuring and cross-compiling the Qt libraries from source; configuring Qt Creator to recognise the cross-compilation toolchain and the cross-compiled Qt libraries; configuring individual programs for cross-compilation; and compiling and deploying programs such as swathRT and swathRT_cmd to development and operational RPis.

The end result of these instructions is a Windows host machine that can cross-compile C/C++ programs that will run on the RPi. The host machine will have a cross-compiled version of the Qt Libraries installed alongside the native version installed with Qt Creator. Programs in Qt Creator can be compiled using the cross-compilation toolchain and linked with the cross-compiled Qt Libraries to produce binaries that will run on the ARM processor on the RPi4. Qt Creator can deploy these binaries to a specific location on an RPi on the same network for testing, and the binaries can be deployed manually by placing them on an RPi and running them like any other program.

5.17 CROSS-COMPILE THE QT LIBRARIES FOR THE RASPBERRY PI 4

5.17.1 Download and unpack the Qt source code

Download the source code from https://download.qt.io/archive/qt/. Select the Qt version you want from the set of directories and subdirectories presented. Select **Single**, and choose either the .zip or .tar.gz files. These instructions have been tested on Qt 5.15 but they may work on other versions. The files should be extracted close to the system's root directory to minimise the length of the resulting file paths. *Note: unzipping the Qt zip file can take a very long time, and appear to be doing nothing at all at first; be patient*!

5.17.2 Download and install the necessary compilers and interpreters

For each of the following tools, get the newest versions

- Download the MinGW 64 native toolchain
  https://gnutoolchains.com/mingw64/
- Download the Raspberry Pi cross-compilation toolchain (or the one for your target)
  https://gnutoolchains.com/raspberry/
- Download and install the strawberry-perl Perl interpreter
  http://strawberryperl.com/
- Download the Python 3 interpreter
  https://www.python.org/downloads/

The rest of these instructions are written assuming that the default installation locations for these tools were selected.

5.17.3 Configure Qt for cross-compilation

Navigate to the root of the extracted Qt repository and run the following command:

```
.\configure.bat -skip qtwebengine -skip qtscript -skip qtlocation -de
vice linux-rasp-pi4-v3d-g++ -device-option CROSS_COMPILE=C:/SysGCC/ra
spberry/bin/arm-linux-gnueabihf- -sysroot C:/SysGCC/raspberry/arm-lin
ux-gnueabihf/sysroot -platform win32-g++ -nomake examples -opensource
 -confirm-license -no-opengl -prefix /out/RPi4/
```

The script will first build qmake then use it to check for the necessary conditions for compilation.

Below is a description of each of the options. This is not necessary for running this script but may help in adapting these instructions for versions of Qt for other platforms.

The modules `QWebEngine`, `QScript`, and `QLocation` do not support ARM so the following options stop them from being compiled

```
-skip qtwebengine -skip qtscript -skip qtlocation
```

The following option optimises the GCC/G++ flags for the hardware on the RPi4. Similar options for other versions of the RPi exist and can be found in the Qt Configure Command documentation. Use a different option for other platform targets:

```
-device linux-rasp-pi4-v3d-g++
```

The following option tells qmake to use the cross-compiler to compile Qt. If you selected a different install location when installing the cross-compiler this should be changed accordingly:

```
-device-option CROSS_COMPILE=C:/SysGCC/raspberry/bin/arm-linux-gnueab
ihf-
```

The following option tells the cross-compiler to use the standard library files and compilation utilities downloaded with the cross-compiler rather than ones built for Windows:

```
-sysroot C:/SysGCC/raspberry/arm-linux-gnueabihf/sysroot
```

The following option tells the repository to build qmake and other compilation utilities using the MinGW toolchain:

```
-platform win32-g++
```

The following option tells qmake not to build the examples as some of the examples do not support ARM.

```
-nomake examples
```

The following option bypasses some question asked at the beginning of the configuration process. The first states that the open source version of Qt is being configured, as opposed to the commercial version. The second agrees to the GNU GPL licensing.

```
-opensource -confirm-license
```

The following option tells qmake not to include any libraries requiring OpenGL. This prevents errors during the configuration and does not affect basic GUI applications such as swathRT_cmd.

```
-no-opengl
```

The following command sets a subdirectory in which to install the compiled Qt libraries. The root of this directory corresponds to the sysroot directory specified with the `-sysroot` option. In this case, Qt will be installed in `C:/SysGCC/raspberry/arm-linux-gnueabihf/sysroot/out/RPi4/`.

```
-prefix /out/RPi4/
```

5.17.4 Build Qt

After the configuration script has completed successfully, run the following command to build Qt:

```
mingw32-make -jN
```

Where N is replaced by the number of threads to use. A higher N means a faster compilation but there are diminishing returns once N grows greater than the number of logical processors present on the host system.

This process will take several hours, and will should not require intervention if all goes well.

Before continuing to the next step, run the command again without the -j option so make steps through each instruction sequentially. Errors can sometimes get buried when multiple threads are used for the make command. Rerunning it with a single thread means that any errors are obvious.

5.17.5 Install Qt

Once the build step has completed successfully, Qt can be installed using the following command:

```
mingw32-make install -jN
```

This a shorter process but the -j option is still used as before.

Once this step has completed successfully, this version of Qt can be used to cross-compile Qt projects in Qt Creator.

5.18 TROUBLESHOOTING

Below is a list of issues encountered during the investigation of this process along with the solutions found.

5.18.1 General Tips:

- If the configuration stage fails, the source code should be extracted from the archive file again as there is no way of removing the makefiles that qmake generates if it fails part way through.

- If the compilation stage fails, the command `mingw32-make clean` should be run to remove any code that was compiled, and the configuration stage should be repeated. If there is any suspicion that this has not restarted the compilation stage from scratch then the repository should be extracted again as something is likely very wrong with the configuration of the repository.

5.18.2 Problems Encountered

Configuration stage failing whilst building qmake:
This is likely an issue with the host system's native C/C++ toolchain. Having multiple downloads of MinGW installed can cause linking errors. Placing the installation of MinGW used in this process above other toolchains in the PATH variable or uninstalling other versions of MinGW will solve this issue.

Configuration stage failing after qmake is built:
Once qmake is built, the configuration script begins compiling a set of minimal programs using both the native and cross-compilation toolchains to determine whether all the necessary dependencies are installed. Most, but not all, of these dependencies are critical and a makefile will not be generated if the accompanying test fails. To identify the failed test add the option `-v` to the configuration command. This will print the results of compiling and running the test programs, allowing the identification of the missing dependency.

Qmake throws an error saying that it doesn't have a sysroot option:
This error seemed to have been *caused* by adding a slash to the end of the sysroot path so it looked as follows `C:/SysGCC/raspberry/arm-linux-gnueabihf/sysroot/` rather than `C:/SysGCC/raspberry/arm-linux-gnueabihf/sysroot`. This was likely compounded by the affects of the following issue.

'File does not exist' errors when headers from the standard library are imported:
This problem was caused by qmake replacing the path to sysroot with gibberish in hundreds of makefiles. This was caused by writing the path to sysroot using backslashes rather than forward slashes in the configuration command. Make sure all paths in the configuration script follow the unix convention for forward slashes in file paths.
This could also be caused by passing an incorrect path to the configuration script.

LibCore or possibly any other file complaining of "a syntax error in VERSION script":
These version scripts are generated using the Perl interpreter. However, the makefile produces an empty VERSION script and just doesn't fill it if there is no Perl interpreter installed. The lack of a Perl interpreter generates a small error the first time it is encountered but in each subsequent run of `mingw32-make` the makefile sees that the VERSION script already exists so it doesn't try to invoke the Perl interpreter again.
To solve this issue, delete the empty VERSION script that is generating the syntax error, ensure the terminal being used for compilation has access to a Perl interpreter, and run `ming32-make` again.

The compilation of a specific library fails but QtCore has been built successfully: This is likely due to incompatibility between that library or its dependencies, and the ARM architecture. To solve this issue, return to the configuration process and add the command `-skip <Library>` where `<Library>` is replaced with the incompatible library. QWebEngine, QScript, and QLocation have been skipped for this reason.

Include errors are generated when trying to include .cpp files:
This was solved by ensuring that configuration and compilation occur within the directory that Qt was extracted in. This was not a problem when including header files as their locations were denoted with `-I` compiler options. However, these did not work for the relative paths used to reference other .cpp files, so the relative paths were broken when the object code for each .cpp file was compiled away from the source directory.

5.19 SETTING UP QT CREATOR FOR CROSS COMPILATION

The following instructions describe the method for configuring Qt Creator to cross-compile programs using the toolchain and cross-compiled Qt Libraries installed in the previous section.

5.19.1 Assumptions

- Host PC OS: Windows 10
- Qt Creator Version: 5.12 or later
- Native C/C++ compilers are already installed on the Host PC.
- The instructions for the cross-compilation of the Qt Libraries have been completed.

5.19.2 Add the cross-compiled version of Qt to Qt Creator

On the menu bar, go to **Tools > Options**. In the window that pops up, click on Kits at the top of the side bar. On the set of tabs above the near the top of the window, click Qt Versions.

Click **Add** then select the `qmake` executable that can be found in the top-level `bin` directory of the unpacked Qt library.

This will allow Qt Creator to use this version of Qt during compilation.

5.19.3 Add the cross-compilation toolchain to Qt Creator.

On the set of tabs mentioned in the previous step, click **Compilers**.

If the toolchain is on PATH then Qt Creator will automatically detect it and add it to the list of compilers.

If this doesn't work, the compiler can be added manually using the **Add** button. Click **Add > GCC > C** and enter a unique name and the path to the C cross-compiler `C:\SysGCC\raspberry\bin\arm-linux-gnueabihf-gcc-8.exe` in the dialog that appears below. Click Add > GCC > C++ and repeat the instructions above for the C++ cross compiler `C:\SysGCC\raspberry\bin\arm-linux-gnueabihf-g++.exe`

5.19.4 Add the Raspberry Pi Debugger to Qt creator.

On the set of tabs mentioned in the previous step click on Debuggers.

Click **Add**. In the dialogue that appears below, enter a unique name and the path to the gdb debugger `C:\SysGCC\raspberry\bin\arm-linux-gnueabihf-gdb.exe`

5.19.5 Add the RPi used for development as device in Qt Creator

This will enable Qt to deploy and run programs on the RPi when the Build & Run Button (Green play button at the bottom left) is clicked. If the program has no GUI elements (e.g. swathRT) then it can will run over an SSH connection displayed in Qt Creator. If the program does have GUI elements (e.g. swathRT_cmd) then Qt Creator will deploy it but will not run it.

Click **Devices** in the side bar of the options window. Click **Add…**

In the window that pops up, select **Generic Linux Device** and click **Start Wizard**.

In the window that pops up, enter a unique name for this device, its LAN IP address, and the username the system should use to establish an SSH/SFTP connection to the device. Click **Next.**

If the user has already set up public/private keys for their version of Qt, click browse and find the location of the private key file. This is usually found in the directory `C:\Users\<username>\.ssh\` under the name `qtc_id`. The corresponding public key found in qtc_id.pub should be placed in the file `~/.ssh/authorized_keys` file on the Pi.

If a key pair does not exist, they can be generated using the Create New Key Pair button. On the window that pops up, press **Generate And Save Key Pair**. The key in `C:\Users\<username>\.ssh\qtc_id.pub` should be placed in `~/.ssh/authorized_keys` directory on the Pi.

This will allow Qt to deploy and run the compiled program assuming the projects .pro file has the required instructions.

5.19.6 Set up the cross-compilation kit in Qt Creator

On the set of tabs mentioned in the previous step, click Compilers

Click the add button next to the list of kits. Set the following options in the dialogue that appears below:

- **Device Type:** Generic Linux Device
- **Device:** <Name of the device set up in the previous step>
- **Sysroot:** C:-linux-gnueabihf
- **Compiler > C:** GCC (C, arm 32bit in C:)
- **Compiler > C++:** GCC (C++, arm 32bit in C:)
- **Debugger:** *<The name given for the Raspberry Pi Debugger in the previous step>*
- **Qt Version:** Qt 5.15.0 (RPi4)

The entry for Qt version may change if the Qt libraries are recompiled.

5.20 CROSS-COMPILING AND DEPLOYING EXISTING SOFTWARE

The instructions below describe the process and results of cross-compiling existing pieces of software in the environment produced in the previous two sections.

After the completing the previous two sets of instructions, the host PC should be ready for cross-compilation. This final section describes the configuration of individual projects for cross-compilation; the configuration required for a RPi to run the compiled programs; and compiling, deploying, and executing a program.

5.20.1 Project Configuration

1. Configure an existing code repository in Qt Creator to use the kit set up in the previous section

If the code is being loaded for the first time, a setup dialogue will allow a kit to be selected.

Otherwise, with the project open, click on Projects on the side bar of the main window. Right click on the name of the cross-compilation kit underneath Build & Run. Click Enable Kit “<Kit Name>” for Project “<Project Name>”.

2. Add instructions to the project’s .pro file to allow Qt Creator to deploy the code on the RPi.

Add the following text to the bottom of the project’s .pro file:

```
#Rules added for deploying to a Raspberry pi
qnx: target.path = /tmp/$${TARGET}/bin
else: unix:!android: target.path = </path/to/deployment/dir/>
!isEmpty(target.path): INSTALLS += target
```

This tells Qt creator where to deploy the code to. Replace `</path/to/deployment/dir/>` with the path on the Pi at in which to deploy the project. This should be within the home directory of the user specified in step 6 to avoid “Permission Denied” errors during the SFTP transfer.

5.20.2 RPi Configuration

If the RPi in question can natively compile Qt projects, then it already has the libraries required to run Qt. If this is not the case, follow in the instructions listed in Annex C Section 5.9 on installing Qt 5 onto the RPi.

5.20.3 Compiling, Deploying, and Executing Programs

Qt Creator and the program in question should now be configured for cross-compilation. The existing buttons for building and running the program found in the bottom left of Qt Creator will behave as they did when compiling natively. The Build button (The hammer) will build the project using the cross-compilation toolchain and place the resulting executable in the project’s build directory. This executable can be installed manually on a RPi4 and run like any other program.

Pressing the build and run button (The green play button) will produce different results depending on whether the program in question is GUI or CLI based. In both cases, the executable will be deployed onto the RPi at the location specified in instruction two in replacing `</path/to/deployment/dir/>`. Qt Creator will then try to run the program over an SSH connection. The CLI program will run successfully in this environment. However the GUI program will not be able to produce an GUI elements over an SSH connection and it will fail.

The GUI program can be run and tested on the RPi directly by connecting a keyboard and mouse. It can also be run over a virtual desktop connection using the VNC server that comes loaded with Raspbian/RPi OS. Instructions for setting this up can be found here: https://www.raspberrypi.org/documentation/remote-access/vnc/