Appendix D: Using RPi5 (January 2024)

This is a free on-line supplement to my book "Projects Guide for ROBOTIS Engineer: Combined Edition" (https://www.amazon.com/Projects-Guide-ROBOTIS-ENGINEER-Combined/dp/0999391879). It is designed to help the reader set up an RPi5 to be used with this book's materials.

The RPi5 became commercially available in Fall 2023. The RPi5 is roughly 2-3 times faster than the RPi4B used by the author back in 2021 (see Appendix B). The RPi5 comes with either 4 or 8 GB of RAM for developing projects with the ENGINEER Kits 1 and 2 (https://www.raspberrypi.com/documentation/computers/raspberry-pi-5.html).

There is a plethora of suitable cases for the RPi5, the author used the RPi case (left in Fig. D.1) and the GeekPi case (right in Fig. D.1).

Fig. D.1 RPi and GeekPi cases for the RPi5.

D.1 Hardware and Mechanical Design Considerations

D.1.1 Power Options

The RPi5 power requirements are **5V/5A** DC which are higher than the **5V/3A** required for the RPi4B (see Appendix B). The power issues and solutions described in Appendix B for the RPi4B are still applicable to the RPi5, so they are not repeated here.

D.1.2 Memory Storage Options

Robotics users most likely would use the RPi5's microSD memory card to store its RPi OS and application software packages. The author suggests a minimum of 64 GB capacity, with fast memory read/write capabilities that would fit the reader's budget. The author has used "SanDisk Extreme PRO" and "Samsung EVO" products. At the minimum, the user should buy at least **2 units of the same brand and model type** to facilitate the backing up of the RPi OS system – a very important step to keep up with! Unlike in 2021, there are now many options to back up the user's unique RPi OS setup, the author uses a combination of RonR's Image File Utilities (https://forums.raspberrypi.com/viewtopic.php?t=332000) and RPi Disk Imager, and also Win32DiskImager (https://forums.raspberrypi.com/viewtopic.php?t=361820) if ones prefer to stay with graphical interfaces as much as possible. "raspiBackup" (https://raspibackup.linux-tips-and-tricks.de/en/home/) and "rpi-clone" (https://github.com/billw2/rpi-clone) are also worth looking into. The RPi Forum is the primary place to keep track of these developments.

The reader can potentially use USB 3.0/3.1 portable drives that incorporate faster memory technologies such as SSD (Solid State Drive) hardware to have the RPi boot up faster and perform better disk-intensive jobs such as ones for a Web Server. However, our robotics applications rely more on numerical computations and access to external devices such as sensors and communications hardware, and as these tasks are executed inside the 4/8 GB RAM area, these faster USB memory technologies are of no benefits to robotics projects described in this book. PCIe drives and other peripherals should also be available in the near future for the RPi5.

D.1.3 Compatible Peripherals

For **Video Camera** devices, the reader can use the Pi Camera V2 that comes with the ENGINEER Kit 2, but the reader will need to purchase a "standard-to-mini" FFC cable to hook it up to the RPi5 MIPI port (https://www.amazon.com/Makerfocus-Raspberry-Camera-Cable-Ribbon/dp/B0716TB6X3). The author also had successfully used USB webcams such as Logitech C270/C310/C910/C920, which can provide better quality video than the Pi Camera V2, but they are not as "compact" as the Pi Camera with their much longer USB cabling (which can be "hacked" shorter by "willing and able" users). Other more capable Pi Cameras are also available (https://www.raspberrypi.com/documentation/accessories/camera.html). The author suggests readers to look into the V3 camera if they need HDR images and autofocus capability.

The RPi5 cannot detect/work with the **BT-210** nor the **BT-410** receivers from ROBOTIS.

The ROBOTIS **LN-101** module works great as a "wired" serial communications device between an RPi5 USB port (as **ttyUSBx**) and the UART port on the CM-550 (see video at https://www.youtube.com/watch?v=i3ubdCkla6k).

For users handy with making custom cables using Dupont connectors, they can start with ONE 4-pin/3-wire cable that comes with the LN-101 (https://www.robotis.us/robot-cable-3p-1100mm-ln-101/) and obtain TWO custom wired-communications cables that can be used to connect the CM-550's UART port to the RPi5's GPIO pins that correspond to the **mini-UART** (i.e. **ttyS0** or **serial0**) device on the RPi (see Fig. D.2). This is the "lightest" wired-serial configuration.

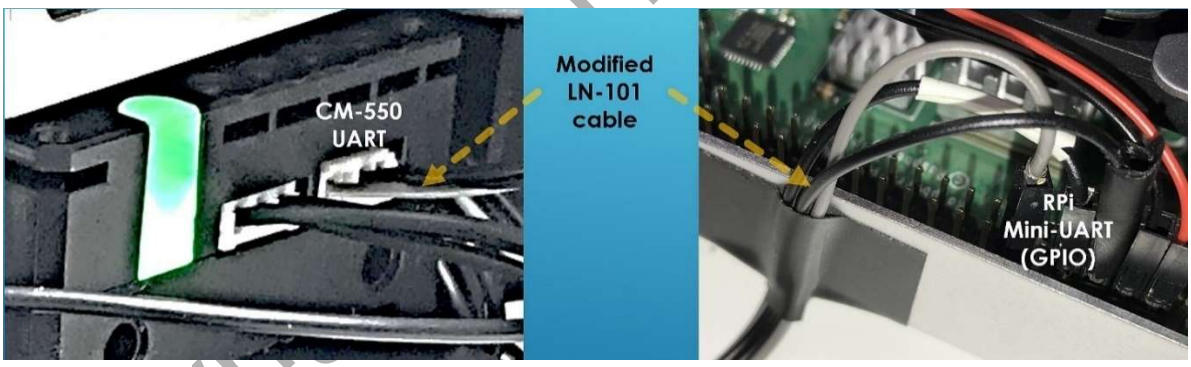

Fig. D.2 "Wired" Serial Communications between CM-550 and RPi's GPIO pins.

There are two important web links that the reader must check out before creating this special UART cable:

a) The CM-550's UART Pinout at https://emanual.robotis.com/docs/en/parts/controller/cm-550/#communication-port.
b) The RPi5's GPIO Pinout is documented at https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#gpio.

The author's UART cable connects the CM-550's UART connector to specific GPIO pins on the RPi5's GPIO Header in the following manner:

- CM-550's Pin 1 (GND) connects to RPi4B's GPIO Header Pin 6 (GND). This establishes a common ground to both CM-550 and RPi4B.
- CM-550's Pin 2 (Power) is not connected to anything as the RPi powers its own UART port.

- CM-550's Pin 3 (RXD) connects to RPi4B's GPIO Header Pin 8 (i.e. GPIO 14 or TXD).
- CM-550's Pin 4 (TXD) connects to RPi4B's GPIO Header Pin 10 (i.e. GPIO 15 or RXD).

For other details, please see video at https://www.youtube.com/watch?v=JL5CssdrCBs and contact the author for a list of components and tools needed if interested.

By far, the easiest way to connect serially between the RPi5 and the CM-550 is via a USB cable **rated for data transfer** (USB A male to micro-B male). The RPi5 would then recognize the CM-550's USB port as **ttyACM0** (on the RPi5, the reader can use the "**dmesg**" command to list out this information).

D.2 RPi OS Software Considerations

The Raspberry Pi Foundation provides great support for RPi users to get started at this web site (https://www.raspberrypi.org/documentation/). If the reader is a first-time user of the RPi5, this is the first place to start learning (and return to) regarding the RPi ecology. There is lots of web-based information "in the wild" for the RPi, and the user will need to sort (and try) out to ascertain which information is current and which is not. That is why backing up one's RPi OS microSD card is so important before attempting any big upgrade!

The current RPi OS is named **Bookworm** using 64-bits and it is optimized for the RPi5 and its RP1 module. For robotics application, the **libcamera** library and the **rpicam-apps** tools are well integrated into **GStreamer** and **OpenCV** (https://forums.raspberrypi.com/viewtopic.php?t=361778).

Commercial books abound for RPi beginners, and the reader needs to find the one(s) that would fit the reader's skill levels best. The author recommends a few "keeper" references:

- "The Official Raspberry Pi Beginner's Guide", 5th Edition by Gareth Halfacree.
- "Exploring Raspberry Pi" by Derek Molloy.
- "Raspberry Pi Cookbook" by Simon Monk.
- "Linux Cookbook" by Carla Schroder.

The Raspberry Pi Imager tool is available for Windows, macOS, Ubuntu and on the RPi itself at https://www.raspberrypi.org/software/. This is a great tool to use in preparing your microSD card and to install on it the latest RPi OS of your choice (and later to back up your current RPi OS into another SD). For the first-time installation, the author recommends connecting the RPi to physical keyboard, mouse, HDMI display and wired Ethernet to ease troubleshooting if needed.

On the RPi, the author recommends the installation of the Desktop and other software for C/C++/Python Programming and Remote Access (https://www.raspberrypi.com/documentation/computers/remote-access.html#introduction-to-remote-access) (i.e. the Full 64-bit version of Bookworm OS).

Among the Desktop utilities, the author has often relied on:

1. Under "Accessories", the "SD Card Copier" tool can be used to back up the booting microSD card into another USB microSD card reader for archiving purposes. The author prefers "Imager" as it is faster and verifies the backed-up SD card. **Imager** also allows handy OS customizations such as "chosen" Username and Password, as well as the user's local WiFi SSID and password, see documentations at (https://www.raspberrypi.com/documentation/computers/getting-started.html#raspberry-pi-imager).
2. Under "Preferences", the "Raspberry Pi Configuration" tool is key to setting up important features (see Fig. D.3) such as:
    a. Booting up into the Desktop and auto-logging in as the custom user.
    b. Enabling the Serial Port (**ttyS0** or **serial0**) and Disabling the Serial Console. These settings are important as they enable the access of **ttyS0** through GPIO Pins described earlier in Section D.1.

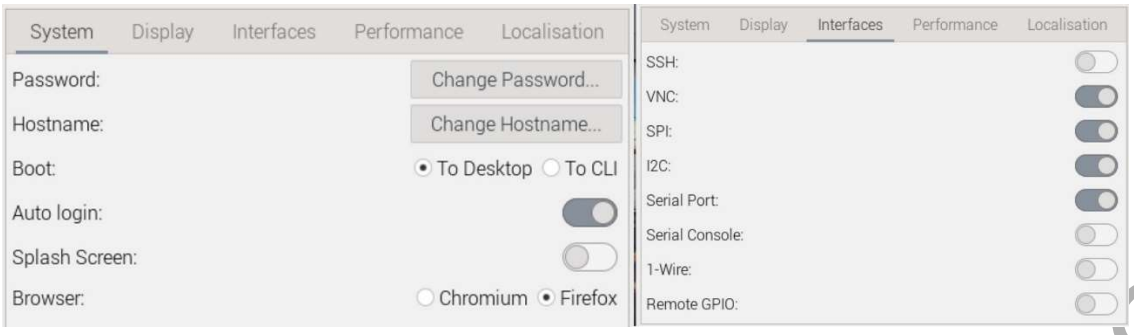

Fig. D.3 Author's setup on his "Raspberry Pi Configuration" tool.

3. Also, under "Preferences", the "Screen Configuration" tool (a.k.a. "Screen Layout Editor") is used by the author to set a "fixed-size" display screen to be used when the RPi is in headless mode (see Fig. D.4).

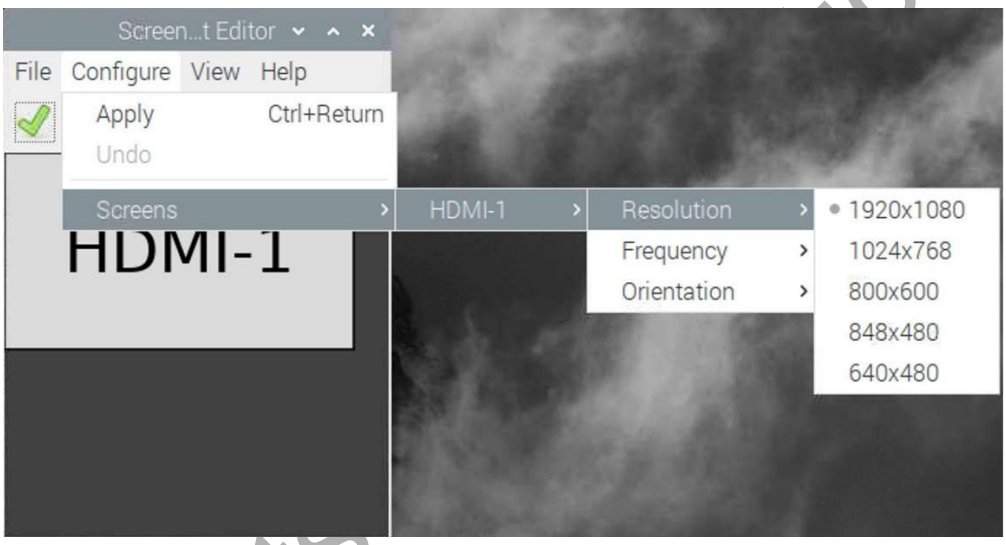

Fig. D.4 Author's setup on his "Screen Configuration" tool.

Currently, the RPi's VNC Server is enabled on Wayland and it used to be on X11, this is creating some problems for some folks in using VNC Viewer https://www.realvnc.com/en/connect/download/viewer/ on the Remote PC. Please review this post for some current work-around (https://forums.raspberrypi.com/viewtopic.php?t=357582), furthermore RealVNC and RPi folks are working on resolving these issues due in First Quarter 2024. Personally, the author has no problem using the VNC Viewer within his local network, however the File Transfer component is not currently available (so back to File Transfer via USB drives for a while).

This Viewer tool allows keyboard and mouse access into the RPi's Desktop (i.e. RPi in headless mode) – please see Fig. D.5.

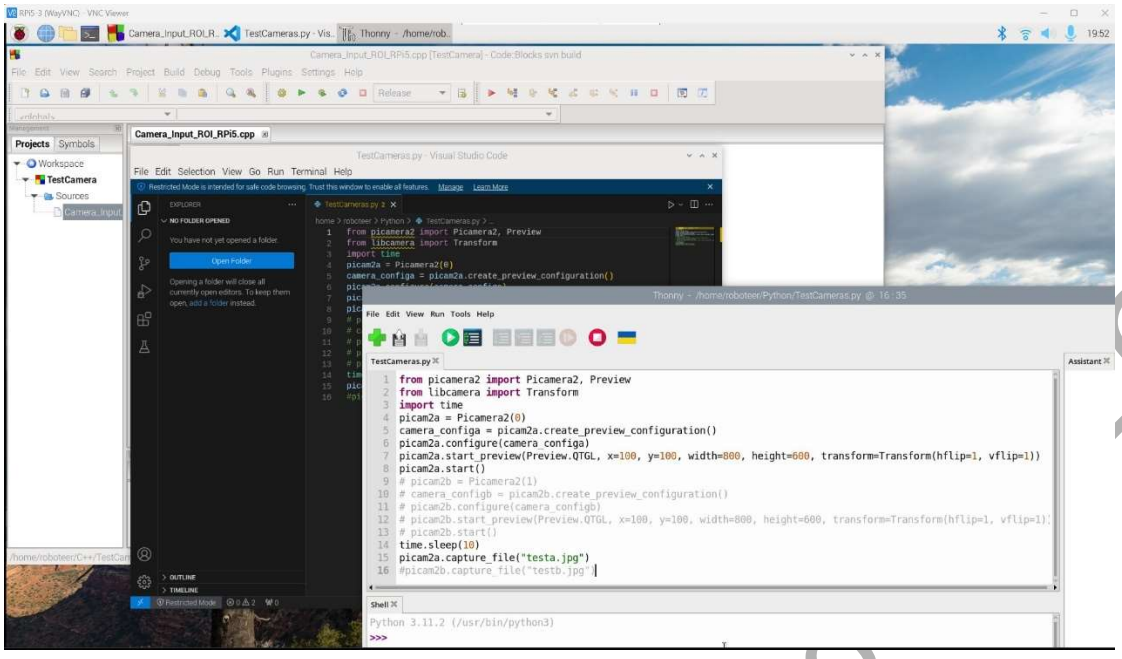

Fig. D.5 WiFi Remote Access to RPi's Desktop via VNC Viewer.

The author suggests two more "handy" software tools to be used at the PC Desktop level:

1. "AOMEI Partition Assistant" in case your microSD card got "mangled bad" format-wise, (https://www.aomeitech.com/aomei-partition-assistant.html) – the Standard Edition is free.
2. "Win32 Disk Imager" to back up the RPi OS images that need archiving "somewhere safe"! (https://sourceforge.net/projects/win32diskimager/)

D.3 Setting up BlueTooth Services between RPi and Windows PC

Section D.1.3 showed how to set up a "wired" serial connection between an RPi5 and a CM-550 as these systems are usually in close physical proximity to each other on any practical robot. But for some projects, we may need an additional "wireless" connection between an RPi and a Desktop PC, fortunately the RPi5 has both WiFi and BlueTooth server hardware. However, the BT Serial Port Profile (SPP) is not set up by default on the RPi OS as normally installed.

This web link https://scribles.net/setting-up-bluetooth-serial-port-profile-on-raspberry-pi/ provided detailed steps allowing the RPi to communicate via a BT Serial Port to a phone. The author adapted Steps 2, 3, 4 and 5 to his Windows 11 environment. Assuming that a VNC Viewer Session had been set up successfully with the RPi from the Windows Desktop (see Section D.2), the next steps are as followed:

a) Assuming the user has logged on to the RPi as the custom user with administrative privileges and has opened a **bash** Terminal where the tasks laid out in Step-2 "Enabling SPP on RPi" from the above web link are next implemented. After SubSteps 2-1, 2-2 and 2-3 are performed, the reader can just do a "reboot" of the OS to restart the BT service or perform the SubSteps 2-4 and 2-5 without having to reboot the RPi.
b) Once the reader is back into a **bash** Terminal, please issue the following command at the prompt line:
    - **sudo sdptool browse local**
    - Then lots of information will be scrolling down the **Bash** Terminal. The reader can then scroll up and look for a specific group of information as the one given below:

      Service Name: Serial Port
      Service Description: COM Port
      Service Provider: BlueZ

Service RecHandle: 0x10006
Service Class ID List:
"Serial Port" (0x1101)
Protocol Descriptor List:
"L2CAP" (0x0100)
"RFCOMM" (0x0003)
Channel: 1
Language Base Attr List:
code_ISO639: 0x656e
encoding: 0x6a
base_offset: 0x100
Profile Descriptor List:
"Serial Port" (0x1101)
Version: 0x0100

- If the "Serial Port" Service is shown as above, the **sdptool** has done its job!

c) Step 3 from the above web link can be performed by issuing the listed commands on the RPi's Terminal, or the reader can perform more interactive steps as shown below:

- The "new" Step 3-1 is for the reader to go the RPi Desktop and left-click on the BlueTooth Manager icon on the right of the Main Menu Bar. A pull-down menu will be shown (see Fig. D.6). Next, enable the "Make Discoverable" option for the RPi which will make the BlueTooth icon blink RED.

Fig. D.6 Pairing RPi and Windows PC via BlueTooth.

- The "new" Step 3-2 is to get back to the Windows PC and run the usual "Add BlueTooth Device" App from "Settings" (assuming the reader has made the PC "discoverable" – such as "LEGION" in the author's setup). It will take some time for the PC to find the RPi which will show up with the Host Name given to the RPi5 (see Fig. D.6 where the author's RPi5 was named "RPi5-3").
- A "new" Step 3-3 may be needed if Step 3-2 is having a hard time detecting the RPi5, whereas the reader can go back to the RPi BT Tool, left-click on the "Add Device" option and start of looking for and pairing with the reader's PC (i.e. "LEGION" for the author's case).
- If all goes well, the user will see that his/her PC is listed as a Connected Device in the RPi BT Manager tool (top right picture in Fig. D.6). The RPi5 will keep this pairing for subsequent boot-up and login from then on.

- If the user then goes back to the PC and uses "Control Panel" to "View Devices and Printers", the RPi5 will be shown as a multi-role Audio Device, although we only need to use its SPP service (see "COM19" in the bottom right picture in Fig. D.6).

d) Step 4 from the above web link can be performed to establish Connection between RPi4B and PC by using an RPi's **bash** Terminal to issue the following command on the prompt line:
   - **sudo rfcomm watch hci0**
   - This Terminal will then show the following response.
     Waiting for connection on channel 1
     (Please note that this Terminal should not be "**closed**" unless the BT connection between the RPi5 and the PC is no longer needed).

e) Step 5 is best shown in this web video available at https://www.youtube.com/watch?v=nqzhT7ENaxQ, whereas the PC was connected to two RPis via BlueTooth, using a specific procedure that goes back and forth between the RPis and the PC, using Python and Visual Studio Code (see Sub-Sections 4.6.2, 5.6.3 and 5.6.4 also).

D.4 Setting up Python 3 and Visual Studio Code

In February 2021, Microsoft Visual Studio Code (VS Code) becomes available for both 32-bit and 64-bit RPi OS (https://www.raspberrypi.org/blog/visual-studio-code-comes-to-raspberry-pi/). General setup information is available at https://code.visualstudio.com/docs/setup/raspberry-pi, and Python specific setup information is at https://code.visualstudio.com/docs/setup/raspberry-pi#python-articles. The author had followed these instructions and they worked great on his 64-bit RPi OS: see tutorial on making the PC as a Supervisor Controller of two RPi+CM550 robots at https://www.youtube.com/watch?v=nqzhT7ENaxQ. For C/C++ projects in this book the author uses Visual Studio 2019 at the Windows PC level and Code::Blocks at the RPi level. For Python projects, the author had found that Visual Studio Code works better than Thonny especially for ROBOTIS Dynamixel SDK Library (https://forum.robotis.com/t/using-rpi5-with-robotis-systems-and-gstreamer-opencv/5183).

D.5 Setting up Code::Blocks

For the RPi5 with the December 5 2023 64-bit OS installed, the "standard" Code::Blocks IDE that can be installed via the shell command "**sudo apt install codeblocks**" is Version 13046 which was found to be adequate for RPi projects developed in this book.

D.6 Setting up ZigBee SDK

Volume 1 (Thai, 2020) shows how to use the ZigBee (i.e., ZIG2Serial) SDK in a Windows 10 environment and Visual Studio 2019 (see Section 2.4 of Volume 1). Fortunately, there is a Linux version for the ZigBee SDK that works with Raspian 64-bit and it is available for download at this link http://support.robotis.com/en/baggage_files/zigbee_sdk/zigbee_sdk_linux_v1_00.zip. The instructions for how to build a binary library for it in Linux is available at https://emanual.robotis.com/docs/en/software/embedded_sdk/zigbee_sdk/#for-linux (essentially just use the MAKE utility).

However, for the RPi5 (using Bookworm OS), the **gcc** compiler now requires **-fcommon** as an extra argument for its CFLAGS parameter for the MAKEFILE to work properly. Please see this post for more details (https://forum.robotis.com/t/cant-use-dynamixel-sdk-on-raspberry-pi/1236/8).

The author has made a tutorial video with more details and it is available at this link https://www.youtube.com/watch?v=i3ubdCkla6k, as an application of this SDK on the RPi4B and Code::Blocks (but it applies for the RPi5 too).

Please note that the author had found that the ZigBee SDK only works with communication devices such as **ttyS0** or **ttyUSB0**, but not with **ttyACM0** at run-time (although **ttyACM0** codes would compile OK). The user would have to use Boost ASIO Serial Port with **ttyACM0** (see more details in Sub-Section 4.4.1).

D.7 Setting up Dynamixel SDK

For installing and using the ROBOTIS Dynamixel SDK, please refer to these web links for "quickstart" in Python or C applications (https://community.robotis.us/t/dynamixel-sdk-quick-start-python/260, https://community.robotis.us/t/raspberry-pi-dynamixel-quick-start-c-language/257, https://forum.robotis.com/t/using-rpi5-with-robotis-systems-and-gstreamer-opencv/5183/2). As the **gcc** compiler has been upgraded to a version higher than **5** (as shown in those videos), please make that you add the flag **-fcomm** in the appropriate MakeFile.

With the U2D2, there is a way to increase DXL data throughput by changing the USB Latency Timer setting from 16 ms (default value for Linux) to 1 ms (see https://emanual.robotis.com/docs/en/software/dynamixel/dynamixel_sdk/faq/#how-to-change-an-usb-latency-in-dynamixel-sdk).

D.8 Setting up BOOST libraries

Some projects in this book would require the use of 2 serial communication ports for each programming application whether on the PC or on the RPi. The author's previous ENGINEER book (i.e. Vol 1) showed how to set up and use the BOOST libraries to achieve this requirement at the PC level.

With the December 5th 2023 version of Bookworm OS, if the reader uses the standard command "**sudo apt install libboost-all-dev**", version 1.74.0.3 would be installed and is adequate for this book projects. The Boost Include files can be found in the standard folder **/user/include/boost**.

But if the user really wants a more recent version, the procedure for the RPi is unfortunately more involved. To create the setup procedure used in this book (**using V. 1.7.5 as an example and for an RPi4B running on Buster 64-bit OS – the reader is welcomed to reproduce these steps on an RPi5 with Bookworm OS if needed**), the author combined the setup information gathered from two web links:

- https://www.domoticz.com/wiki/Build_Domoticz_from_source
- https://www.boost.org/doc/libs/1_75_0/more/getting_started/unix-variants.html

The following procedure installs the BOOST Libraries V. 1.75.0 from source codes onto an RPi4B on 64-bit RPi OS using a **bash** Terminal:

1. On the author's RPi4B, the **cmake** tool is V. 3.16.3 which turns out to be adequate for this job.
2. We need to check that the OS is **clean of previous BOOST libraries** (which are called "**libboost**" on the RPi. Issue the following command line in a **Bash** Terminal:
   - **sudo apt search libboost**
   - The system would respond with a long list of packages named "libboost-XXXX". On a "clean" system, all libboost-XXXX entries would have no information enclosed in parentheses or only with "(default version)".
   - On the author's RPi, there was some libboost libraries shown as "(installed)". For example "libboost-regex-dev", so the author has to remove this library with this command: **sudo apt remove --purge --auto-remove libboost-regex-dev** The reader can also issue a "blanket" command as shown in the "domoticz.com" link, any libboost library found "not installed" will just be "ignored".
3. We need to download the source code package for BOOST V. 1.75.0 to the RPi, the author happened to choose the "Downloads" folder inside the "/home/pi" folder so the following commands were used:
   - **cd Downloads**
   - **wget https://dl.bintray.com/boostorg/release/1.75.0/source/boost_1_75_0.tar.gz**
   - **tar xfz boost_1_75_0.tar.gz**

     (this task takes about 2 minutes to get done and it creates a folder named "boost_1_75_0" where there are a "multitude" of files and folders needed for the upcoming build/install process)
4. The next steps are to build and install the BOOST libraries:

- **cd boost_1_75_0**
- **ls**

  (to list all the files and folder in "boost_1_75_0", you will see a file named "bootstrap.sh" which will be invoked next)

- **./bootstrap.sh**

  (this task takes about 1 minute to get done and its goal is to build the "b2" tool which has an on-line manual at https://boostorg.github.io/build/manual/develop/index.html#_introduction that explains the various options used in the next two command lines)

- **./b2 --build-dir=/tmp/build-boost toolset=gcc variant=release link=shared threading=multi runtime-link=shared stage >BuildBoost.log 2>&1**

  (this command line builds "shared" libraries only and this task runs for about 12 minutes and it creates a big file named "BuildBoost.log" inside the "current" folder, i.e. "boost_1_75_0", that would hopefully show no errors!)

  (to build **"static" and "shared"** libraries, use the following command line:

  ./b2 --build-dir=/tmp/build-boost toolset=gcc stage >BuildBoost.log 2>&1 ,

  this task needs ~26 minutes to complete)

- **sudo ./b2 install --build-dir=/tmp/install-boost toolset=gcc variant=release link=shared threading=multi runtime-link=shared >InstallBoost.log 2>&1**

  (this command line installs "shared" libraries only and this task takes about 15 minutes and creates the file named "InstallBoost.log" inside the "boost_1_75_0" folder, that would show hopefully no errors!)

  (to install "static" and "shared" libraries, use the following command line:

  sudo ./b2 install --build-dir=/tmp/install-boost toolset=gcc >InstallBoost.log 2>&1

  This task needs ~29 minutes to complete)

5. The **include** files for the BOOST libraries can be found in the folder **/usr/local/include/boost** and the BOOST **static** and **shared libraries** can be found in the folder **/usr/local/lib**. These folders would need to be referred to in the configuration files for Code::Blocks (see Section D.8).

D.9 Setting up OpenCV + GStreamer + Libcamera

At the time of writing of this book (Summer 2021), the Pi Camera utilities such as **raspistill**, **raspivid** and the Python **picamera** module were not working with the 64-bit Beta Buster OS (https://www.raspberrypi.org/forums/viewtopic.php?f=29&t=213435&p=1839305&hilit=picamera+64+bit#p1839305). Thus, the author settled on OpenCV, all the while realizing that not all possible hardware performances of the Pi Camera V2 (at the time) would be achievable with this option.

At present (January 2024), the **libcamera** library has matured along with the OS shift to the more stable **Bookworm** OS integrating with the **GStreamer** framework, and **Python** has moved on to the **picamera2** module. Hardware wise, we also have the V3 and Global Shutter cameras coming into service. So, there had been lots of hardware and software development since 2021. Fortunately for the author and for the readers, these changes affect the OS backend much more than the application programming aspects of the projects used in this book.

The only change necessary is about how the camera is initially connected to the new OS backend. For a long time, the traditional OpenCV procedure for connecting and using a camera was via a camera ID (=0, 1 and so on, depending on how many cameras were being used). This procedure can be illustrated with the following programming statements:

```
VideoCapture vid_cam(cam_id);      // cam_id is an integer = 0 for first camera
...
vid_cam >> frame;      //to acquire a new image frame
```

At present, the new procedure requires the assignment of a GStreamer **pipeline** parameter which is a C/C++ string that describes the camera source in terms of its detailed operations such color space, pixel resolutions and frame rate:

```
string p_line1 = "libcamerasrc camera-name=/base/soc/i2c0mux/i2c@1/imx708@1a ";
        // PiCamera V3 identification in the OS backend
        string p_line2 = "! video/x-raw, format=BGR, width=640, height=480, framerate=100/1
        ";      // frame formatting parameters
        string p_line3 = "! appsink";  // corresponds to OpenCV program at runtime
        string pipeline = p_line1 + p_line2 + p_line3;
        VideoCapture vid_cam(pipeline, cv::CAP_GSTREAMER);  // connect camera
        ...
        vid_cam >> frame; //to acquire a new image frame - same as before.
```

And fortunately, the rest of a typical OpenCV application code remains unchanged!

However, there is lots of work to be done to get the OS backend properly set up to handle such a GStreamer pipeline. Please see this RPi Forum post for more details (https://forums.raspberrypi.com/viewtopic.php?t=361778). Essentially, the software libraries **OpenCV, GStreamer** and **libcamera** need to be rebuilt with the proper settings so that they can work together for our ROBOTIS projects.