Documentation Notes

Giuseppe Cervone
giuseppe.cervone@edu.unife.it

1 Introduction

The aim of the project is to develop what in cybersecurity and IoT is called a data diode, a unidirectional communication device for data exchange. The importance behind this project is that industrial data diodes are expensive devices, and implementing a high amount of them like for this specific use case is a very costly move, but with software implementations (firewalls) or hardware implementations (serial or optical communications) it is possible to have cheaper data diodes that are almost as functional for most use case scenario. The initial setup will include two Raspberry Pi model 3 with a specific configuration which we will highlight later, and a TTL-232R-3V3 cable without the RX pin being plugged in one of the raspberries.

2 Devices

2.1 Raspberry Pi

As far as the Raspberry Pi models, I have chosen to use the Raspberry Pi 3 as it's a nice midway point between performance and cost (Includere comparativa con raspberry 2 e 4). This particular machine should be able to handle the data size we are planning to share (around 1Gb/day), at a cost that is much lower than that of a Raspberry Pi 4. The machines are configured as follows:

- Raspberry Pi 3 Model B Ver1.2:
  - Quad Core 1.2GHz Broadcom BCM2837 64bit CPU
  - 1GB RAM
  - BCM43438 wireless LAN and Bluetooth Low Energy (BLE) on board
  - 100 Base Ethernet
  - 40-pin extended GPIO
  - 4 USB 2 ports
  - 4 Pole stereo output and composite video port
  - Full size HDMI
  - CSI camera port for connecting a Raspberry Pi camera
  - DSI display port for connecting a Raspberry Pi touchscreen display
  - Micro SD port for loading your operating system and storing data
  - Upgraded switched Micro USB power source up to 2.5A
  - Raspberry Pi OS Lite (latest version)

  - 16GB microSD card

Using the RPI-Imager tool I installed the latest Raspberry PI OS image. Upon first boot up, remember that raspberry pi login is username: pi, password: raspberry. As Raspberry Pi OS Lite is an OS without a desktop environment, we don't have a usual system setting manager, luckily we can use raspi-config to enable a series of features useful for testing software. We initialize raspi-config by using the command sudo raspi-config and then changing these specific settings:

- In the system options:
  - Enable WIFI for ssh purposes.
  - Change username and password if needed.
- In the interface options:
  - Enable SSH.
  - Enable Serial interface, remembering to disable login shell but enable serial interface.
- Run sudo apt-get upgrade to make sure all packages are up to date before using raspi-config.
- Disable bluetooth:
  - We won't be using bluetooth for the project, so we disable it, making the final product more secure.
  - We disable bluetooth by adding 'dtoverlay=disable-bt' in /boot/config.txt.
- Troubleshooting
  - Check that UART is enabled in /boot/config.txt by adding 'enable_UART=1'.
  - Check that serial console is disabled by removing "console=serial0,115200" (or "console=ttyS0,115200") in /boot/cmdline.txt.
  - Check for permissions in the dialout group.
  - Check that /dev/serial0 doesn't have a getty console running on it. In case it does, it can be disabled by using the commands: sudo systemctl stop serial-getty@ttyS0.service and sudo systemctl disable serial-getty@ttyS0.service.

2.2 UART Cable

As mentioned earlier, the serial communication will go through a TTL-232R-3V3 cable, which is USB to UART, with +3.3V TTL levels UART signals. The cable has 3-pins on one end, and USB on the other. Using GPIO14 pin on the RPi we can trasmit data, using GPIO15 we receive data. As previously mentioned, GPIO15 can be removed after the initial tests are over, as the communication will be one-way. In appendix 1, we can see an extract from the the documentation of the cable, showing the correct way to plug in the cable in the correct pins, make sure that GND pin is in any of the ground pins available on the raspberry GPIO, and most of all remember that TX and RX are from the machine's point of view, thus they have to be plugged in the opposite way (GPIO14 with yellow and GPIO15 orange). With the cable plugged in we run the command ls /dev/tty* on both raspberry machines so we have an idea of what port we are using on each raspberry in the serial communication (For the port that's sending data it should be /dev/ttyS0, while for the other port it should be USB0). Raspian will always map the serial port to /dev/serial0, so it is suggested to use the alias in programming.

In Appendix 2 is an image of how the two machines should look like once plugged in. This image also gives us further information on which machine will be exposed to a future outside network, and which machine won't have communications coming in.

3 First test

Having the machines setup, you can test if the machines are working correctly by running the two programs found in appendix 1. In case this program won't run correctly, refer back to the troubleshooting section earlier. The code sndtest.py has to be run on the machine where the GPIO pins are being used, and recvtest.py has to be run on the machine where the USB is plugged in. It's suggested you start by running recvtest.py using the command python3 recvtest.py, and then run python3 sndtest.py. Below is an image of what it should look like. (Aggiungero' in futuro, sono sulle raspberry)

4 Data diode 0.1

The first implementation of a data diode I will try out is a project based on a github repo that is not mantained anymore. https://github.com/thephez/data-diode. This project tries to accomplish a very similar feat to what I've been trying to do, with the exception that it also wants to load files on dropbox. The project also uses CTS as a handshake system between the two machines, an implementation which I think I will need in the future.

5 Appendix

5.1 Appendix 1: Pinout uart

5.2 Appendix 2: Machines plugged in

5.3 Appendix 3: Example code and output

4.2 TTL-232R-5V and TTL-232R-3V3 Cable Signal Descriptions

Header Pin Number	Name	Type	Colour	Description
1	GND	GND	Black	Device ground supply pin.
2	CTS#	Input	Brown	Clear to Send Control input / Handshake signal.
3	VCC	Output	Red	+5V output,

Header Pin Number	Name	Type	Colour	Description
4	TXD	Output	Orange	Transmit Asynchronous Data output.
5	RXD	Input	Yellow	Receive Asynchronous Data input.
6	RTS#	Output	Green	Request To Send Control Output / Handshake signal.

Table 4.1 TTL-232R-5V and TTL-232R-3V3 Cable Signal Descriptions

Figure 1: Pinout del cavo seriale

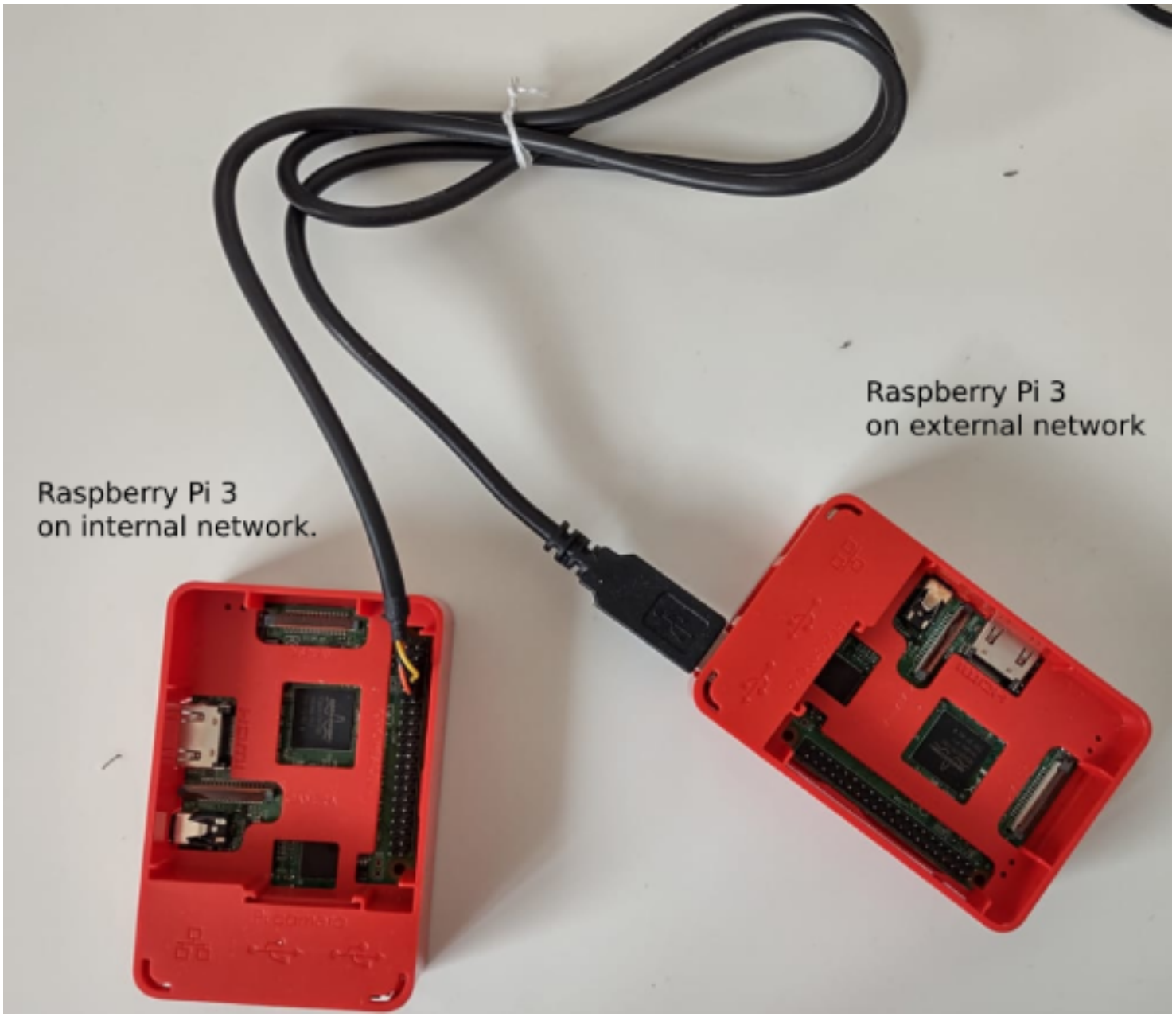

Figure 2: Example 1