Secure Communication with Master and Outstation over DNP3.0 Protocol

Introduction:

OpenDNP3 was released as open-source library. Originally OpenDNP3 was written in C++ which was a reasonable choice to write a platform independent library that needed to run efficiently on embedded Linux. Over time, .NET and Java bindings were added to integrate with more products and platforms. For our project we will be using the original C++ written OpenDNP3, the reason is that C++ is very old, complex, and error prone. Writing asynchronous C++ has always been extremely error prone. However, an extra layer of security with developer's involvement is always needed, so we use the TLS for this added layer of security.

Firstly, we will explain how the implementation of DNP3 open protocol from GitHub and TCP communication. Secondly, configuration changes according to the testbed which will be used will be explained, and in the third, the implementation of TLS – the creation of keys and certificate will be explained using the OpenSSL. In the fourth, the implementation of verifying the entity using a certificate and private key will be explained. Lastly, we will show the design and results of the implementation of the certificate and key. The evaluation of the design will show the screenshots of every step of outputs in the execution phase.

Test Environment:

We have made our environment using LINUX based operating system – Raspbian, running on Raspberry PI 3 with Wireless Area Network (WAN). Our hierarchical model consists of multiple master/outstations like architecture in SCADA systems, and Raspberry PI can run as an embedded system. The Raspberry Pi is a low cost, credit-card sized computer that can be plugged into a computer monitor or connected over SSL and Putty and uses a standard keyboard and mouse. It is compatible to manipulate the Pi using multiple libraries which are application specific and scripting languages like Python.

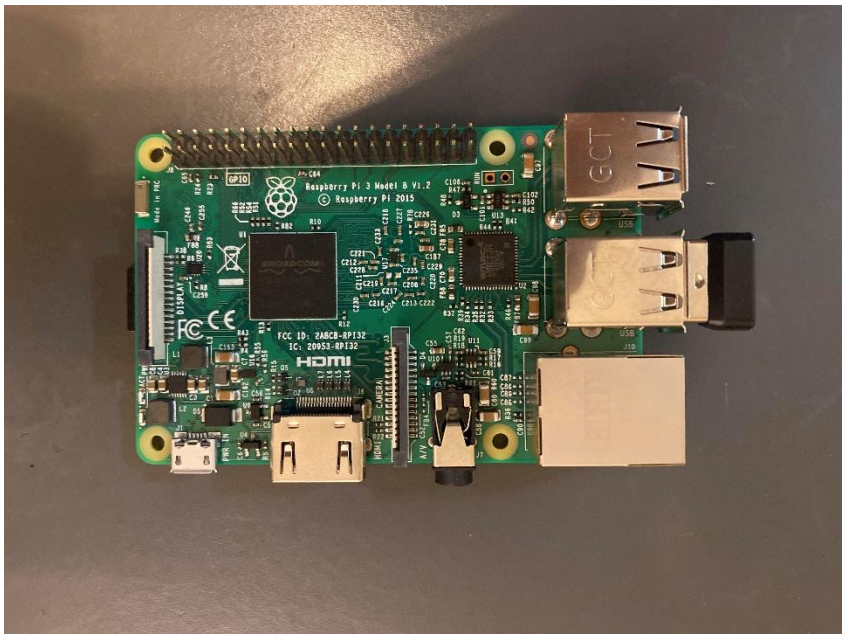
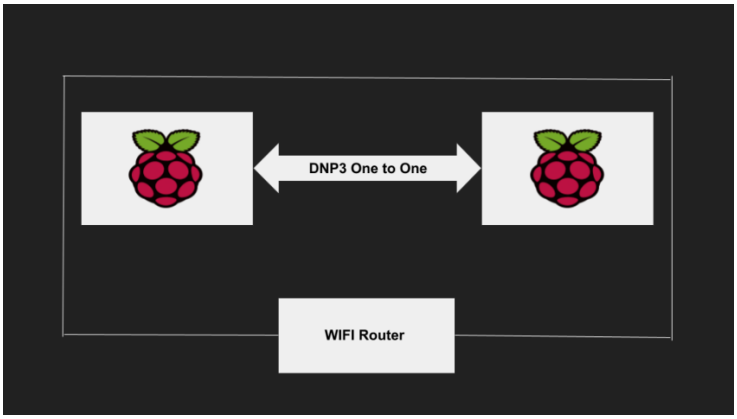
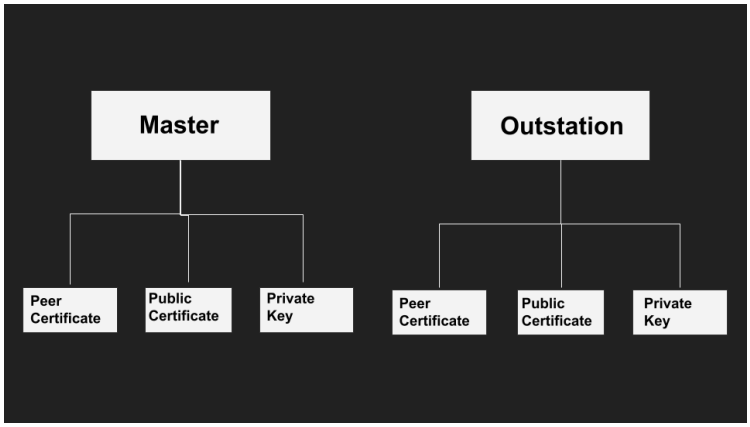


Fig 1.1: Raspberry Pi

For our implementation we have opted Opendnp3 which is a reference to DNP3 (IEEE-1815) protocol, and it is an open source from GitHub “”. It uses Automatak from the open source DNP3. Firstly, we have used C++ version of DNP3 with GCC and G++ compilers and to build the DNP3 in the system we have used CMAKE. The next step after this is to use ASIO as a cross-platform for input and output communication for the Linux. First thing we did is the TCP architecture that are already available in te examples provided by the open source DNP3. TCP/IP connection is shown below:



For the next part in the architecture our plan is to generate the keys and digital certificates over TLS communication. PKI infrastructure is widely used on the internet for many applications. In our architecture we have a digital certificate and private key generated at master and slave. Each entity has a public certificate, private key, and a peer certificate to verify the file transfer.



Configuration:

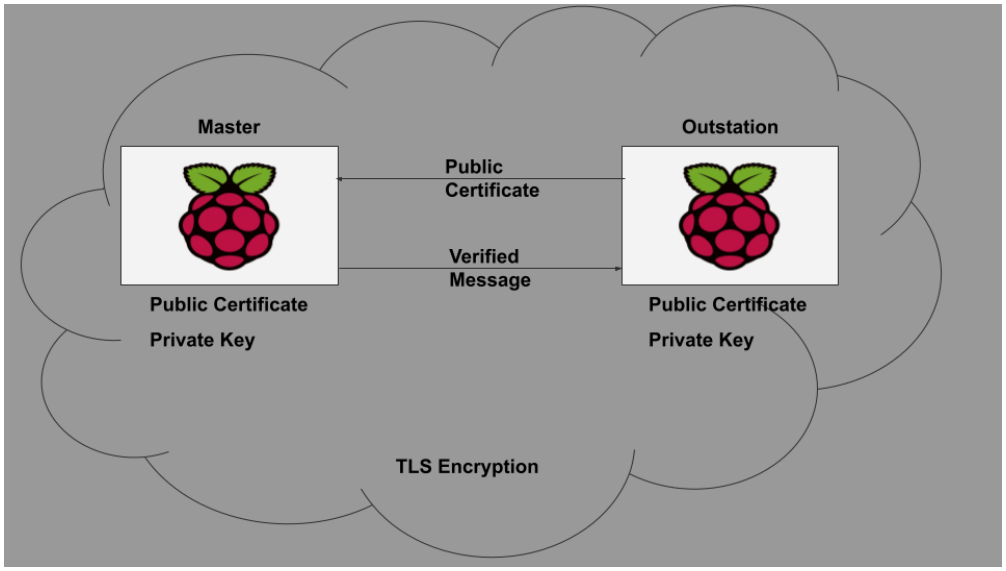
Configuration of DNP3 is the important process of changing the source code according to our implementation. Both the raspberry pi’s will be connected over the same network. Firstly, the main.cpp program is manipulated by changing the IP addresses. Master’s main.cpp consists of outstation’s IP address and the port number. Outstation’s main.cpp consists of master’s IP address as the peer and outstation’s IP address as the local device. The next configuration is to manipulate the TLSconfig.cpp program in the opensource OpenDNP3. The basic configuration in this is adding the path of the peer certificate, public certificate, and private key. Peer certificate for master will be Outstation’s certificate and vice versa. Public certificate and private key are generated individually for each raspberry pi. Certificates and private key are stored in a folder called certificates.

Implementation:

Implementation of this architecture is with the use of openSSL on DNP3 Secure Authentication. The implementation is explained in the following architecture. The first part is in master's perspective, although both master and certificate have three files that are peer certificate, public certificate and private key, master has its own public certificate and private key. The peer certificate is the outstation's public certificate which is shared by outstation over wireless local area network. All these file paths are changed in the TLSConfig.cpp program in the OpenDNP3 opensource code.

After changing the code, the program is executed by CMAKE. After the make command, the demo files are created. The demo files are executed for our operation.

The second part is in outstation's perspective, which has the same architecture as master. After generation of the key's, the pem files are generated.



Evaluation:

In this section, we present the simulation results over the infrastructure. Using the architecture that we explained earlier, we show the master communicating with outstation and verifying the public certificate. After verifying, the password for the private key will be given for the communication. This TLS connection will be shown.

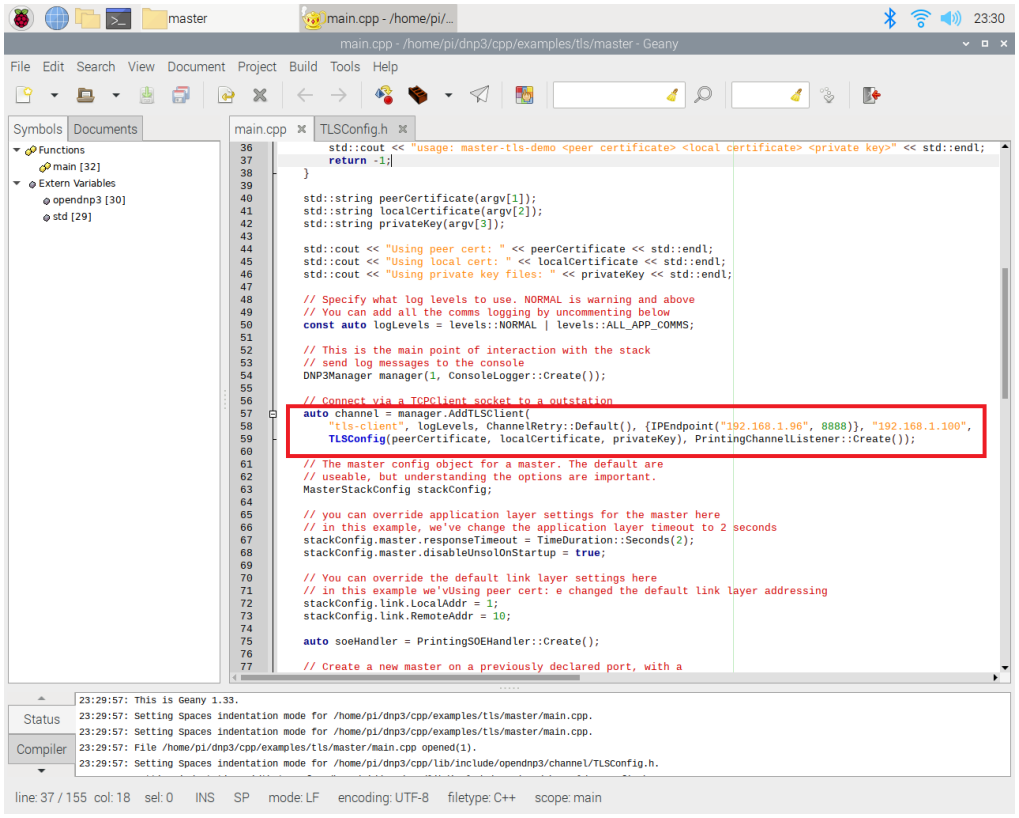


Fig 1.2: IP Change in Master main.cpp

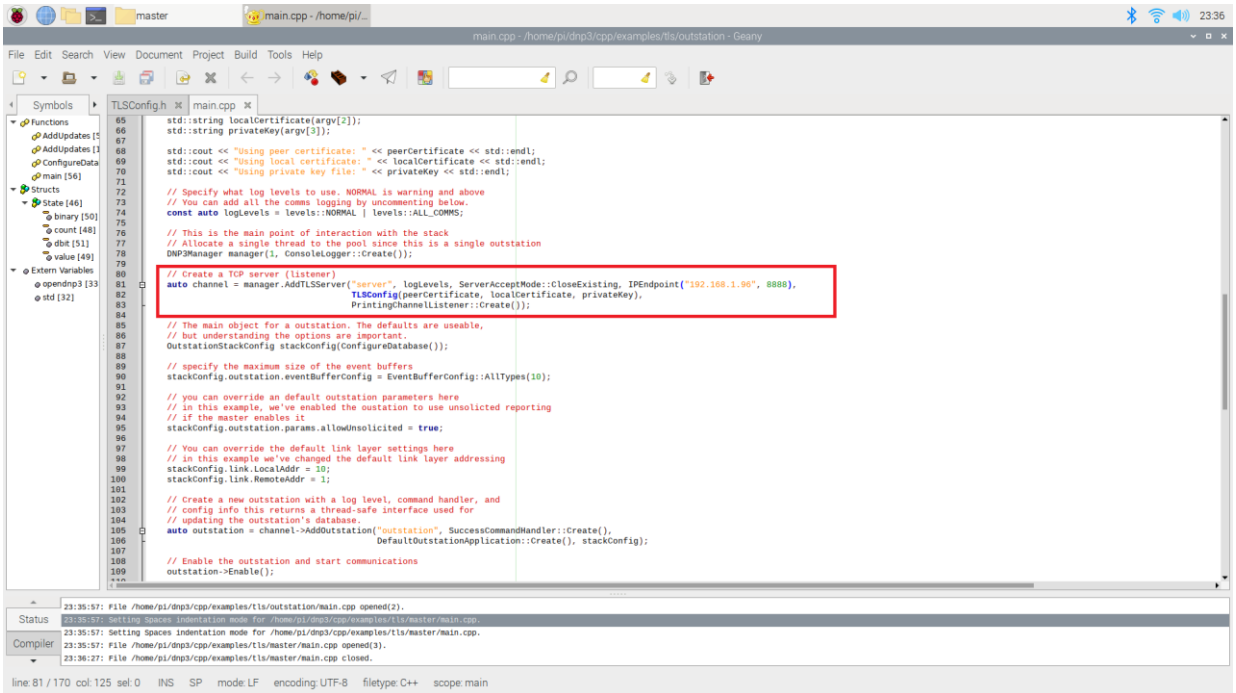


Fig 1.3: IP Change in Outstation main.cpp

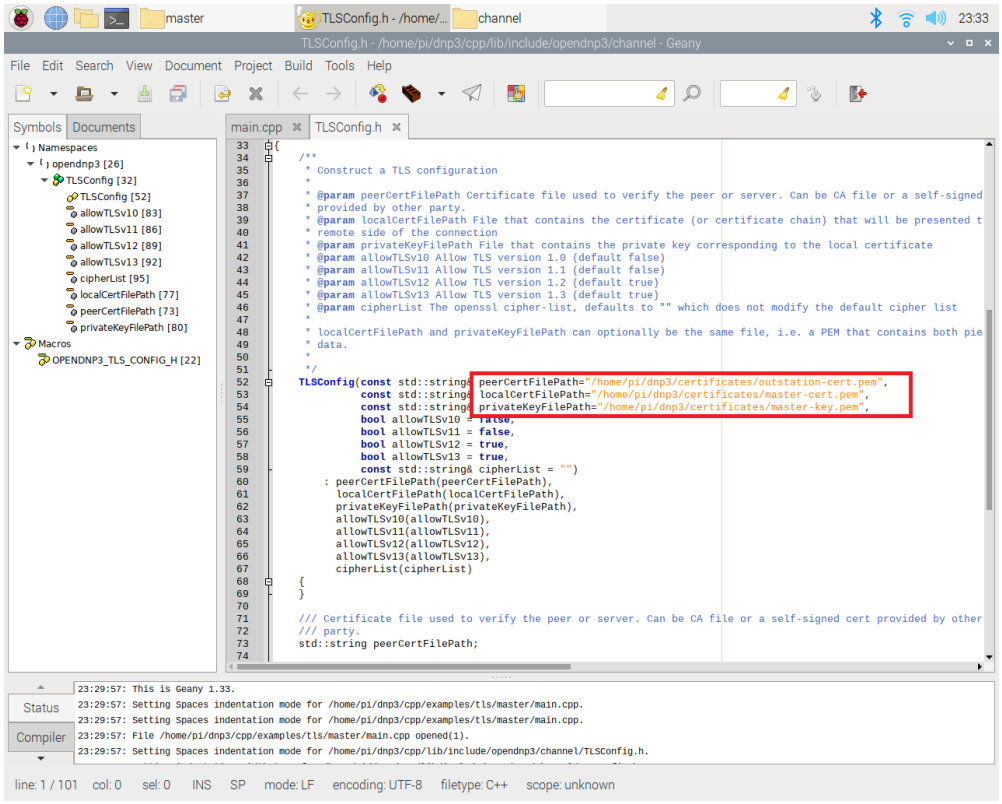


Fig 1.4: Setting TLS certs path in TLSConfig on Master Raspberry Pi

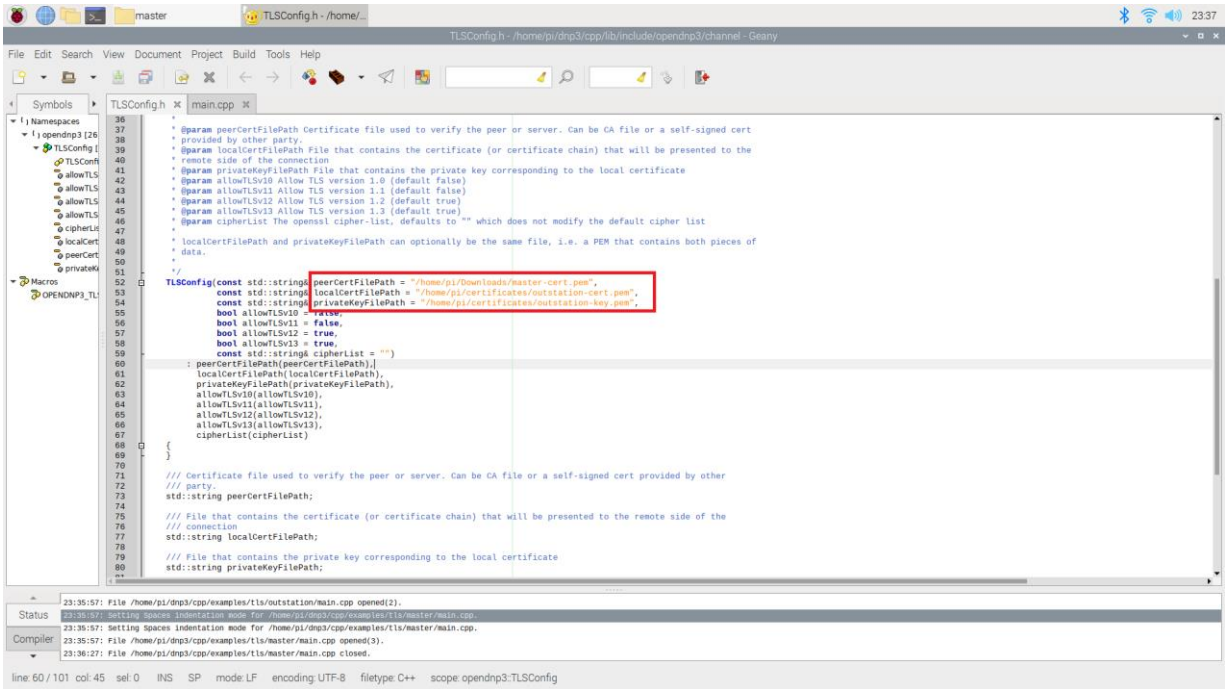


Fig 1.5: Setting TLS certs path in TLSConfig on Outstation Raspberry Pi

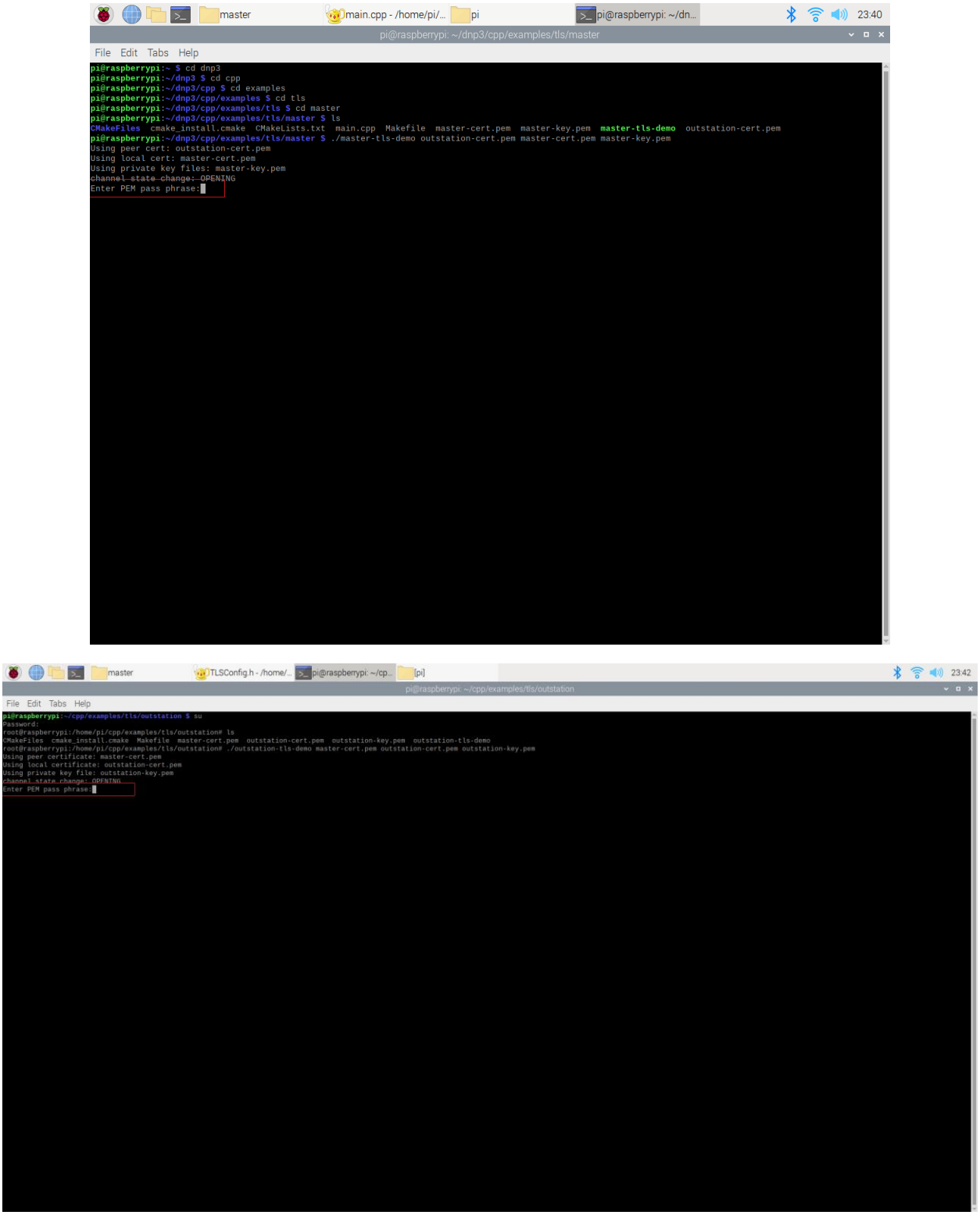
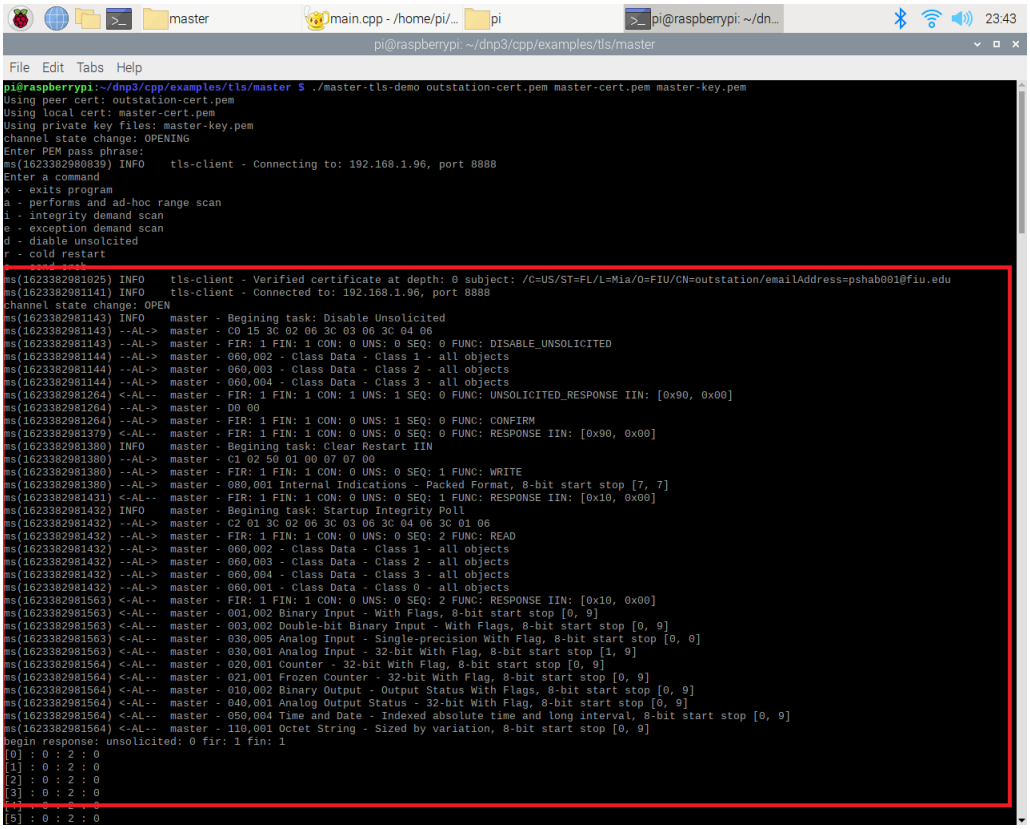


Fig 1.6: Entering cert password for master-tls-demo and outstation-tls-demo



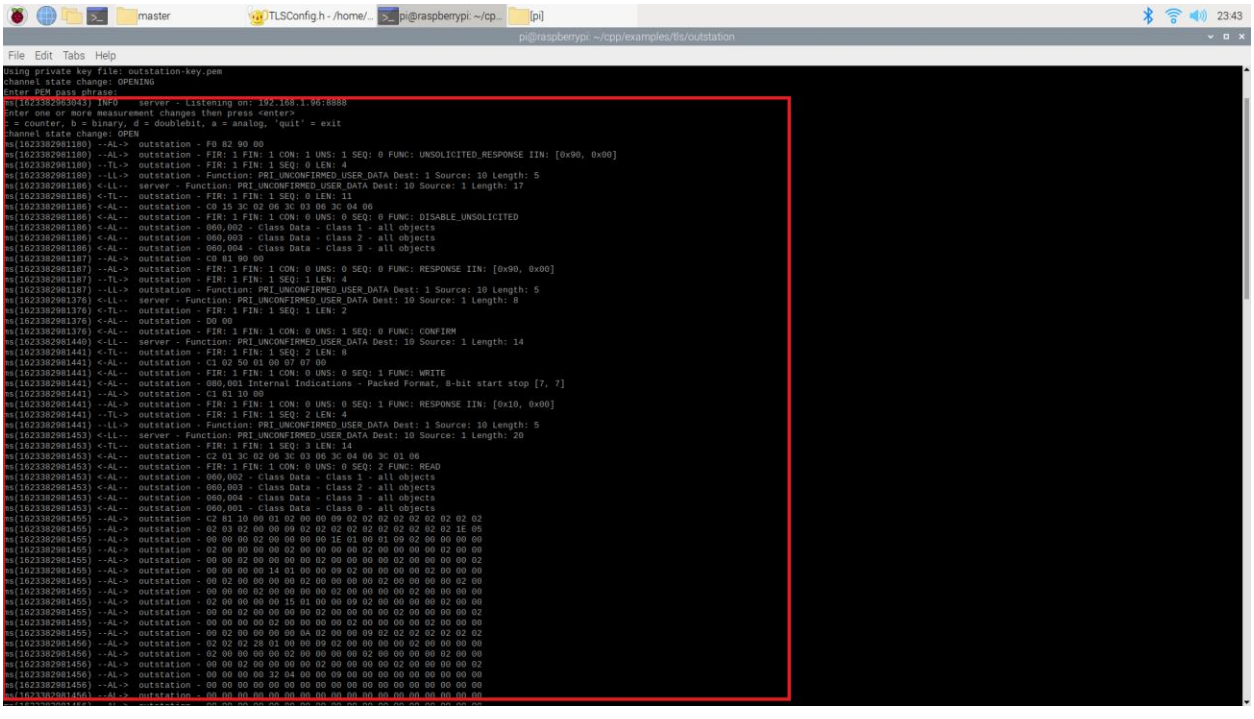


Fig 1.7: Secure TLS connection between Master and Outstation Raspberry Pi

Research continuity:

After establishing the secure communication between two raspberry pi's, data transfer from sensors is essential. Sensor data should be transferred from slave raspberry pi to master raspberry pi over secured DNP3 protocol. After sending sensor data over the raspberry pi's, the next step is to compare DNP3 with MODBUS to check for security and data integrity.

Additional work:

- Worked on research paper(Anomaly detection in smart grid) and submitted the paper to IEEE IECON conference.
- Migrated and activated dspace license.
- Updated all the computer's hardware and software to windows 10.
- Looking into the storage devices and hard drives available in labs.