APPENDIX A

Synchronization Setup

The NTP protocol was used to synchronise the virtual machines that make up this system. The VM of the Domain Controller will be the NTP Server while the Smart Building's and Third Party's VM will be NTP Clients. The NTP Server configuration is demonstrated below.

```
sudo apt-get install ntp
```

For switching to an NTP server pool closest to your location, it is necessary to edit the ntp.conf file with the chosen pools. The following command allows to edit the ntp.conf file.

```
sudo nano /etc/ntp.conf

Chosen pools:
server 0.pt.pool.ntp.org
server 1.pt.pool.ntp.org
server 2.pt.pool.ntp.org
server 3.pt.pool.ntp.org
```

Then it is required to restart the NTP service to apply the new settings.

```
sudo service ntp restart
```

The last step is to configure the firewall so that clients can access the NTP server.

```
sudo ufw allow from any to any port 123 proto udp
```

In the configuration of the NTP clients, it is necessary to install ntpdate, with the command below.

```
sudo apt-get install ntpdate
```

Then specify the IP and hostname of the NTP server in the host file with the following command.

```
sudo nano /etc/hosts

IP:               Hostname:
192.138.173.138   NTP-server-host
```

Next disable the systemd timesyncd service on the client.

```
sudo timedatectl set-ntp off
```

After that install NTP with the command below.

```
sudo apt-get install ntp
```

Next configure the /etc/ntp.conf file to add your NTP server as the new time server.

```
sudo nano /etc/ntp.conf

Add the line:
server NTP-server-host prefer iburst
```

Finally it is required to restart the NTP service to apply the new settings.

```
sudo service ntp restart
```

APPENDIX B

Daml

B.1. Installation

The following commands are required to install VS Code. The VSCode .deb package is first downloaded from the official website. Next, the editor is installed.

```
cd Downloads
sudo apt install ./code_amd64.deb
```

The command below selects the version of the Daml SDK that should be installed. In this dissertation it was installed the version 2.1.1.

```
curl -sSL https://get.daml.com/ | sh /dev/stdin ${SDK_VERSION}
```

B.2. Smart Contract's Code

The code developed for the smart contract can be found in the following link, in DAML CODE section: https://github.com/bclse/dissertation.

APPENDIX C

PostgreSQL Installation

The first command installs PostgreSQL and its dependencies. Once PostgreSQL has been installed, the second command starts the PostgreSQL service.

```
sudo apt install postgresql -y

systemctl start postgresql
```

The first command opens a PostgreSQL work interface for creating a database and user. The following commands create a database, a user, and a password. The final command gives the user full access to the database.

```
sudo su - postgres

CREATE USER user_example WITH PASSWORD 'password';

CREATE DATABASE db_example;

GRANT ALL PRIVILEGES ON DATABASE db_example to user_example;
```

APPENDIX D

IntelliJ Installation

The commands required to install IntelliJ IDEA are shown below.

Install required dependencies.

```
sudo apt install vim apt-transport-https curl wget software-properties-common
```

Install IntelliJ IDEA.

```
sudo apt install intellij-idea-community -y
```

APPENDIX E

SCIAPP Code

The code developed is presented in the following link, in SCIAPP section: https://github.com/b-clse/dissertation.

The following libraries were used to create the code:

```
<dependency>
    <groupId>org.eclipse.paho</groupId>
    <artifactId>org.eclipse.paho.client.mqttv3</artifactId>
    <version>1.1.0</version>
</dependency>
```

```
<dependency>
   <groupId>com.daml.ledger</groupId>
   <artifactId>bindings-rxjava</artifactId>
   <version>100.13.56-snapshot.20200331.3729.0.b43b8d86</version>
</dependency>
```

APPENDIX F

OpenHAB Installation

The two commands required to install JAVA are shown below, where the first command installs the default Java Runtime Environment (JRE 11) and the second command installs the Java Development Kit (JDK 11).

```
sudo apt install default-jre
sudo apt install default-jdk
```

Regarding OpenHAB, this program can only be installed when JAVA has been set up. First, it is added the repository key.

```
curl -fsSL ”https://openhab.jfrog.io/artifactory/api/gpg/key/public” | gpg --dearmor >
    openhab.gpg
sudo mkdir /usr/share/keyrings
sudo mv openhab.gpg /usr/share/keyrings
sudo chmod u=rw,g=r,o=r /usr/share/keyrings/openhab.gpg
```

Next, the HTTPS transport for APT is added.

```
sudo apt-get install apt-transport-https
```

Then it is added the repository.

```
echo ’deb [signed-by=/usr/share/keyrings/openhab.gpg] https://openhab.jfrog.io/artifactory/
    openhab-linuxpkg stable main’ | sudo tee /etc/apt/sources.list.d/openhab.list
```

Finally the OpenHAB distribution package version 3.2.0 is installed.

```
sudo apt-get install openhab=3.2.0
```

APPENDIX G

Basic UI Code

The UI code is divided into three files: demo.sitemap, demo.rules, and demo.items. Both Bob and Alice have these three files. The demo.sitemap file contains what is intended to be displayed on the UI. The items that can connect to real things, in this case the API, are contained in the demo.items file. The demo.rules file contains predefined rules. The code developed is presented in the following link, in OpenHAB section: https://github.com/bclse/dissertation

The script shown below, takes two arguments, which are the exact time when the switch was pressed and this time in epochtime and writes them in a txt file. The script executed in the rules is the same, it simply changes the name of the txt file to which it is written.

```
#!/bin/bash
echo ”Generated timestamp!”
echo $1 $2 >> /home/kali/Desktop/timestamp.txt
```

APPENDIX H

Mosquitto Setup

The commands for installing the mosquitto broker are shown below.

```
sudo apt -y install mosquitto
```

To add authentication to the broker it is first necessary to create a file. This file is called password.conf and has the format shown below.

```
allow_anonymous false
password_file /home/bob/mos/passwords.txt
```

The following commands must be entered in order to create the password file. The configuration file that was previously created is the first argument. The flag -c indicates that an existing file will be overwritten. The password file output is the second argument and the authentication username is the third argument. After running the command, a password must be assigned to the username. Then, after entering the password, the username and the encrypted password are both contained in the passwords.txt file as shown bellow.

```
password.conf -c passwords.txt mqtt
Password: mqtt
Reenter password: mqtt

cat passwords.txt
mqtt:$7$101$hiP1+p7hFtOgeCcm$1A9n7Nt1RHS77Yivc+
    cxIWFRAdtgNUCbTb3jpqJigiAMd6mAii2rCbUPTeWJxF5xztvPy/X5+W1YrlAO1xrtXw==
```