Implementation of SCION on IoT Applications

Vishnu Mohan
Email: mvishnu@st.ovgu.de
Baizil MD
Email: baizil.mulakka@st.ovgu.de

***Abstract*— There are several Internet of Things (IoT) applications running on legacy networks which are not flexible in terms of path selections, network outages and security concerns. Thus, the current situation demands switching over to a new network platform which is highly scalable, secure, and isolated in terms of its architecture. Thus, a basic IoT application is being attempted which will use SCION architecture and hence, surpasses the existing concerns on the current network protocols. The chosen IoT application is based on Retail Offer Management wherein immediate discounts could be provided based on the monitored weights of the articles available for sale. The implementation follows the method of client server communication where the weight sensor data is written to an LCD display with necessary addons including the count of the item, temperature and humidity. The total implementation of the concept includes a Load cell, Weight sensor, Radio Frequency Identification (RFID) unit, Raspberry Pi and 16 x 2 LCD as the components.**

I. INTRODUCTION

THE objective is to create an IoT application over a highly secure network which make use of Raspberry Pi as the microcontroller unit and a 16 x 2 LCD to display the contents/data that are received from the load cell end. Thus, the source of the data displayed will have a sensor unit, load cell, weight sensor and a RFID unit which will be serially connected to the Raspberry Pi. Any data sent from the sensor unit will be fetched and displayed on the LCD display. This application can be easily implemented on Retail Offer Management where a particular product's quantity and count in numbers could be monitored to provide immediate discounts. Moreover, the temperature & humidity of storage location could also be monitored so that adjustments could be made in case if temperature drops. This idea is useful especially for cold storage area of supermarkets where power failure in the storage equipment could change the temperature and result in spoiling for the product stored. Unlike regular communication protocols, the IoT application uses Scalable, Control and Isolation on Next Generation Networks (SCION) for communication among the two units. The idea is to make sure that the communication happening between a client and server machine purely goes through SCION network/path which is a welcoming trend in networking domain, thus creating an environment which purely includes all distinguishing factors of SCION when compared with regular TCP/ IP networks. Additionally, the contents on the 16 x 2 LCD is visualized on a Graphical User Interface (GUI) using Node- RED, which is a flow-based programming tool wiring together hardware devices for Internet of Things (IoT).

A. Overview

Section II shows a few information about SCION network including the security aspects. This is followed by Section III describing how SCION is configured on different platforms. Then, the device specifications along with interfacing and code logic for LCD are explained in Section IV and V respectively. Section VI gives an insight into client server communication and Section VII briefs the script description. Finally, Section VIII and IX summarizes the results and conclusion along with the references and appendix.

II. SCION

THIS chapter discusses in detail about the SCION network which is chosen as the communication protocol for the implementation of the application discussed.

A. SCION in a Nutshell

The popularity of internet has created a high-level dependency on communication. If communication isn't made available for a minute, majority of the processes happening around the globe comes to a halt. However, there lies minimal safety on the network with the rise in technology. Even though huge level of investments and efforts are put on to overcome the current issues, it has been constrained by the present architecture of internet. Thus, SCION [1][4] address the issues on data transmission, scaling and mobility of existing architecture. As mentioned above, the concept of Isolation Domain (ISD) [1] lies in as the fundamental building block for solving the existing concerns. An ISD, in short is a group of Autonomous Systems (AS) [1] as shown on Fig 1.

An ISD includes multiple AS es, creating an ISD Core with core AS es. SCION includes two levels of routing, namely inter ISD and intra ISD and they make use of Path Segment Construction Beacons (PCBs) to create routing paths. The paths are identified and provided to users through **Control Plane.** There is a **Data Plane** too, which ensures packet forwarding through the available paths. Thus, a SCION packet is being forwarded to next AS through border routers based on AS-level path in the packet header. The border router at destination AS checks the destination address and forward it to respective local host.

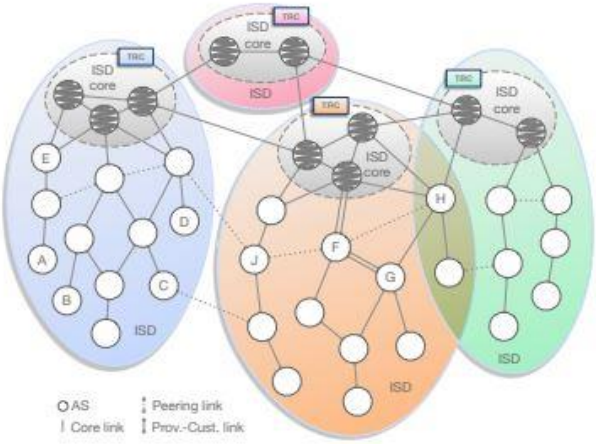

Fig 1. An Autonomous System of 4 ISDs

B. Security Aspects

SCION is designed with various security features to protect against network attacks and to provide secure data/control planes. In this, every AS signs the PCB it forwards, enabling validation by all entities. For path assurance, each AS use a secret key shared among beacon servers and border routers and thereby generating a message authentication code (MAC). Moreover, SCION supports multiple signatures by an AS. The authentication in SCION uses digital certificates that attach identifiers to the keys, carrying digital signatures.

C. SCION Control Message Protocol (SCMP)

Unlike Internet Control Message Protocol (ICMP) in current internet, the control plane in SCION includes SCION Control Message Protocol. If the existing method of digital signature was implemented for authentication, the result would have been a bottleneck creation of SCMP messages. Instead, the method adapted was introduction of DRKey (Dynamically Re-creatable Key) where each AS uses a local secret key known only to border routers. Since this method is dynamic, faster computations will result for fetching the key.

III. CONFIGURING SCION

THIS chapter discusses how the SCION environment is created in a virtual machine and an IoT device to implement client-server communication.

The IoT Application is built, based on the Client –Server architecture. For configuring the SCION AS server, a SCION Virtual Machine is installed on the local machine/PC and for configuring a client SCION AS, a SCION image is flashed on to the IoT device. But in the whole scope of the application, there will be interaction between two AS es deployed on two similar IoT devices. Thus, the scope of using virtual machine is shrunk to testing purposes only.

A. Installation of SCION Virtual Machine

After logging in to www.scionlab.org [2][4] and creating a required login account, create a new AS by clicking on *Generate a new SCION Lab AS*, select the desired attachment point as 19-ffaa:0:1303(OVGU) and choose the option *Install inside a virtual machine*. Then tick the option *Use an Open VPN connection for this AS* and specify the port as 50000. Once completed, download the VM configuration by clicking on *Create and Download SCION Lab VM Configuration*. Now the configuration directory downloaded as a compressed tar file (tgz) is extracted to a directory of our choice. It follows with installation of *VirtualBox* and *Vagrant* applications. In last step, we use the command prompt where we move to the file directory, run *vagrant up* followed by *vagrant ssh* to start a VM shell. After completion of the same, change the directory to *go/src/github.com/scionproto/scion*, run scion using the command *./scion.sh start*. The installation can be verified using the command *tail -f $SC/logs/bs*.DEBUG* which should display the message as *[DEBUG] (MainThread) Successfully verified PCB ca8e78c198a*. This VM is made useful for displaying sample inputs passed from server end as a testing protocol in final application.

B. Installation of SCION on Raspberry Pi

After logging in to www.scionlab.org [2][4] and creating a required login account, create a new AS by clicking on *Generate a new SCION Lab AS*, select the desired attachment point as 19-ffaa:0:1303(OVGU) and choose the option *Install on a dedicated SCION system*. Then tick the option *Use an Open VPN connection for this AS* and specify the port as 50000. Once completed, download the VM configuration by clicking on *Create and Download SCION Lab AS Configuration*. Since the required OS isn't available readily for raspberry pi 3B+, two alternative methods are available of which method 1 described below was chosen for the same. The second method is described in Appendix A.

1) Method 1

- Downloading a copy of Raspbian OS [14] and Ubuntu MATE for Raspberry Pi 3B+ [15]
- Copying and replacing the following files from Raspbian OS to Ubuntu MATE: *bootcode.bin,*

fixup.dat, start.elf, bcm270-rpi-3-b-plus.dtb, kernel17.img, all contents from */lib/modules/4.9.80-v7+* and */lib/firmware/brcm/*

- Using etcher, flash the new OS to the SD card and follow the same steps as described for Ubuntu MATE on www.scionlab.org

Instead of command prompt, we make use of Putty to run SCION OS in raspberry pi. Thus, the SCION network is established on the IoT device which acts as the client SCION AS for the IoT Application.

IV. DEVICE SPECIFICATIONS AND INTERFACING

The Liquid Crystal Display is a commonly used electronic display module having a wide range of applications like calculators, wrist watches etc. The chosen display is the 16 x 2 HD44780 [6][7] dot matrix model to display the incoming load cell data. The display is capable to show two lines of 16 character each, totaling to 32 characters where each character is displayed using 5x8 pixel matrix. The device can be interfaced to any microcontroller in 4 bit or 8 bit mode, differing in how data is sent to the LCD. To write a character in 8 bit mode, an 8 bit ASCII data is sent through data pins D0 to D7 and data is transmitted through Enable pin of the LCD. In case of 4 bit more, data pins used are from D4 to D7 where an 8 bit character ASCII date and command data gets divided into two parts and then sent sequentially through the data pins. Though 4 bit mode is slower than 8 bit mode, we neglect it since the LCDs are known as slow speed devices for decades. The salient features include the backlit and brightness adjustment, lower power requirement (2.7 to 5.5 V).

The IoT device chosen is the Raspberry Pi 3B+ [8][9] where we make use of General Purpose Input Output (GPIO) pins, 5V supply and Ground pins only. The interfacing of LCD to the raspberry pi as follows

- PIN 1 of LCD to ground
- PIN 2 of LCD to 5V supply
- PIN 3 of LCD to a Potentiometer to adjust contrast
- PIN 4 of LCD (Register Select/RS) to GPIO6 on Pi
- PIN 5 of LCD (R/W) to ground
- PIN 6 of LCD (Enable) to GPIO5 on Pi
- PIN 11 of LCD (D4) to GPIO25 on Pi
- PIN 12 of LCD (D5) to GPIO24 on Pi
- PIN 13 of LCD (D6) to GPIO23 on Pi
- PIN 14 of LCD (D7) to GPIO17 on Pi
- PIN 15 and 16 of LCD to 5V supply and ground respectively

The pins D4-D7 on the Raspberry Pi are the data pins which actually displays the data from sensor. Once the connections are made, the 5V supply is given to LCD and Raspberry Pi by connecting the micro USB port of pi to the USB port of the PC/Laptop. The network connection is established by connecting an ethernet cable (RJ 45) between Raspberry Pi and PC/Laptop.

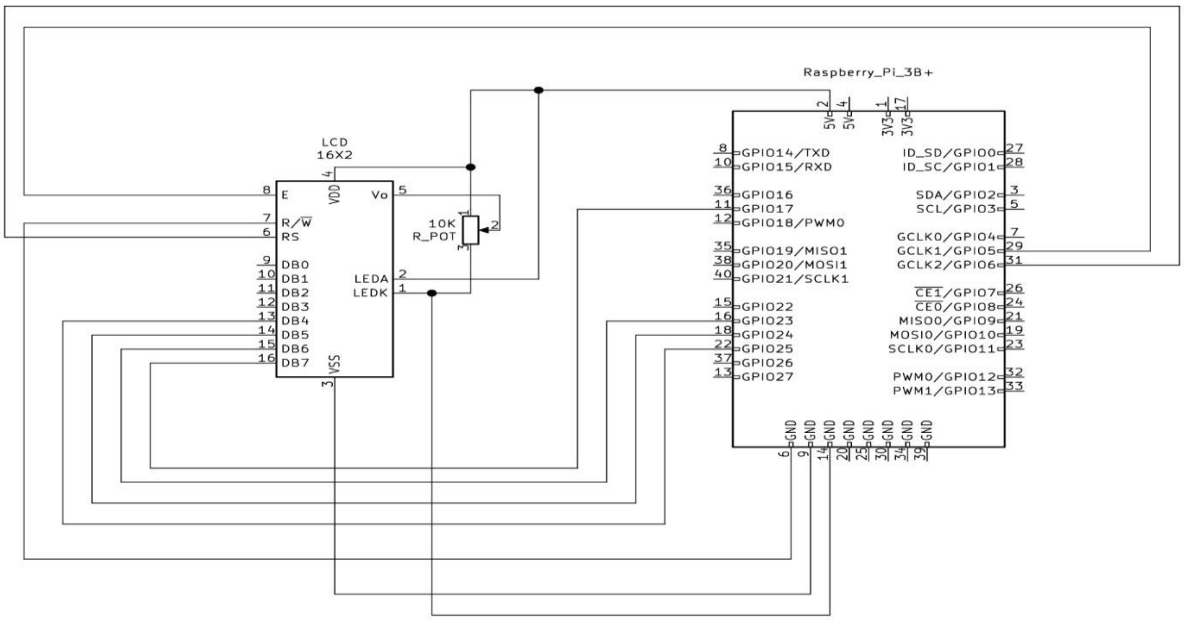

Fig 2. Interfacing Raspberry Pi with LCD

V. CODE LOGIC FOR 16 x 2 LCD

THIS chapter explains the logic of how data is fetched by an LCD and the responsible PINs for the operations.

The data pin DB7 of the display is the busy flag and is in read mode when R/W=1 and RS=0. If DB7=1, it means that LCD is busy and won't accept data. To send data to LCD, we configure R/W =0 and RS=1 to specify write operation and select data register respectively. The data is written only when Enable pin is made high and should rollback to low once data is written. For sending an operation/command to the LCD, we pass an initial command to the data pins with R/W and RS on low state. As said earlier, a high pulse is given at Enable pin when data is sent. An LCD has command and data registers, where the former stores the commands/instructions given to the LCD like initialization, clearing screen, assigning cursor position etc and latter stores the data to be displayed which usually is the ASCII value of the character.

VI. CLIENT SERVER COMMUNICATION

A. *Flow of data*

The simple logic is that client AS requests server AS for load cell data and displays the same, once made available.

The flow of loadcell data to the display is as follows:

- The Raspberry Pi at server side with load cell data will pass on the weight value to the server AS, as soon as a request is made and keeps the server active, waiting for a client request.
- Once a request is passed from client AS, the server AS responds back with the load cell data.
- Now, the data is passed onto the display which read and show the data available at raspberry pi on client side.
- Additionally, the same data is available on GUI, implemented using Node RED [10]

B. *Focus point of the project*

Since the task to be accomplished is to fetch a sensor value and display the same, the scope lies to the right side of dashed vertical line of Figure 3, named as Part B. The left side of the vertical dashed line, named as Part A will be implemented as a different project and finally clubbed together as a Single Retail Offer Management application. As described earlier, the Raspberry Pi is connected to the display and communication is established with server AS through SCION.

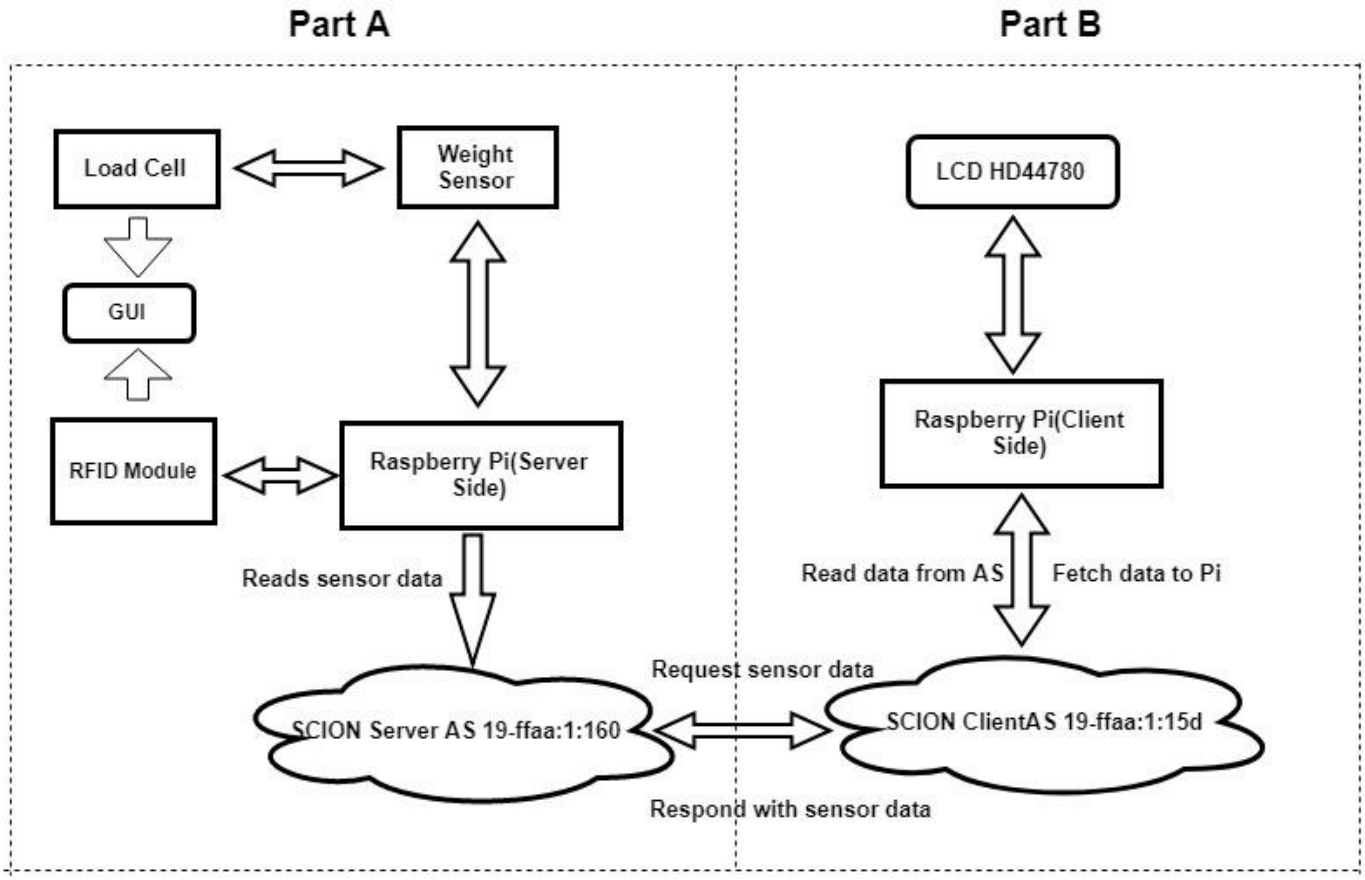

Fig 3. Flow Diagram

C. The Distinguishing Factor- SCION

Unlike the age-old method of communication through regular TCP/IP network, the communication happens purely on SCION network which is built on top of TCP/IP. It stands for Scalability, Control, Isolation on Next-Generation Networks.

The reason to choose the method is that it prevents routing attacks, formally verified protocols, high assurance of communication, path control availability to client and server and presence of Isolation Domain (ISD). It follows an isolation architecture for the path of transmission where as a transparent architecture for data transmission.

VII. SCRIPT DESCRIPTION

The code of the final product will have three main parts namely

- SCION client-server connection establishment
- JSON Object reception
- LCD Writing

The programming language chosen for the implementation is "*Go"* [16]. The simple reason is because of the fact that majority of the development in SCION uses *Go* as the programming language. This reduces probability of errors while using SCION libraries. In order to locally test the application, we assign the Virtual Machine as server and Raspberry Pi as client. The server will send data to be displayed on LCD to the client (Raspberry Pi). The Raspberry Pi always listens to the SCION UDP connection created from the virtual machine. Since the final product makes use of another Raspberry Pi equipped with loadcell, weight sensor and RFID as server, the data will be received as JSON object from server and will be written to the LCD. The JSON object will get decoded using *Unmarshal* function and then stored as a variable which is part of a structure named *Message*, created in the *weight_client.go* file. This implementation of JSON object reception is mentioned under Appendix B

The most important pin in LCD is the E pin, known as Enable pin. When we send data to the LCD, we make it as high to low for each character with some delay. The displaying of data starts the first position of first line using the Hex code 0x80. Initially when the pi receives data over SCION, it will send commands to LCD in order to initialize it and will switch over to data mode to display data

VIII. RESULTS

THIS chapter describes the results seen after the completion of assigned tasks. There will be a description of the test results (where server is the virtual machine) and the final application result (where server is the raspberry pi with loadcell, weight sensor and RFID)

A. Testing Phase

During testing phase, we keep the virtual machine as server and passes the string "Testing 123" continuously with a fixed time interval in milliseconds. Once server client connection is made, the string "Testing 123" will get displayed on the LCD and it keeps on working on a loop until the client execution is stopped.

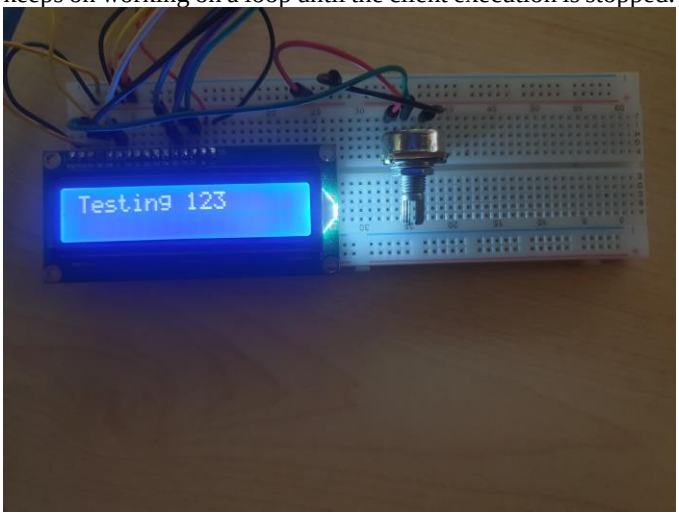

Fig 4. LCD Output when virtual machine is made as client with test data as "Testing 123"

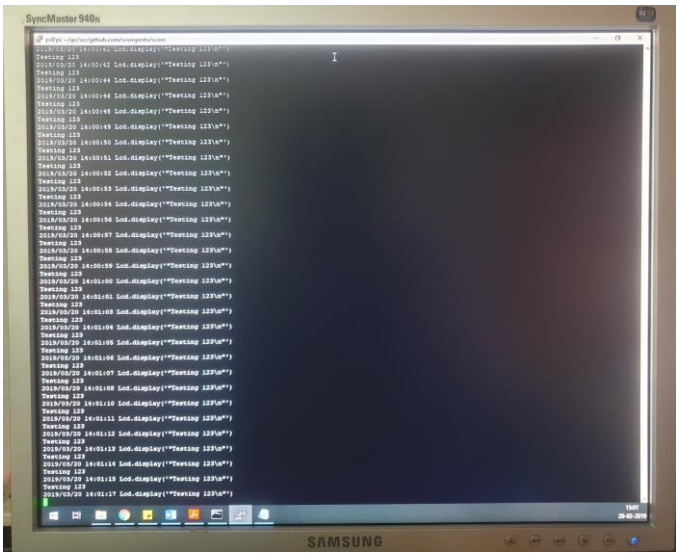

Fig 5. Same output as Fig 4 on command prompt

B. Actual Implementation

In the final product, once a connection is established with the server AS (where the RFID, load cell and weight sensor are deployed), the client side requests the data from server and server gives the data to client and gets displayed on the LCD. The first line of LCD displays the product name, the total weight and second line of LCD displays the count of the item weighed, temperature and humidity. An example for a product named *Sallos* with initial count of four numbers with a prefixed limit of 21 on the palette will be as follows:

Line 1: *Sallos* Wt: 22gms
Line 2: Count 4/21 T:22 H:48

Here, *T* is the temperature in Degree Celsius and *H* is the humidity.

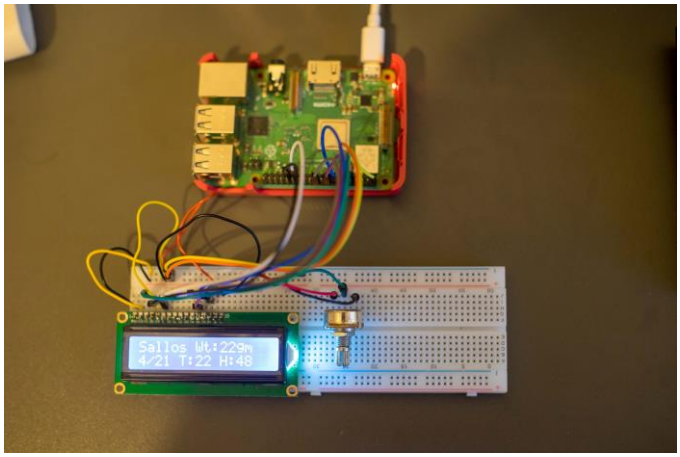

Fig 6. LCD Output

Once we add more quantity to the palette, the weight in grams along with the count will increase and can be observed in real time.

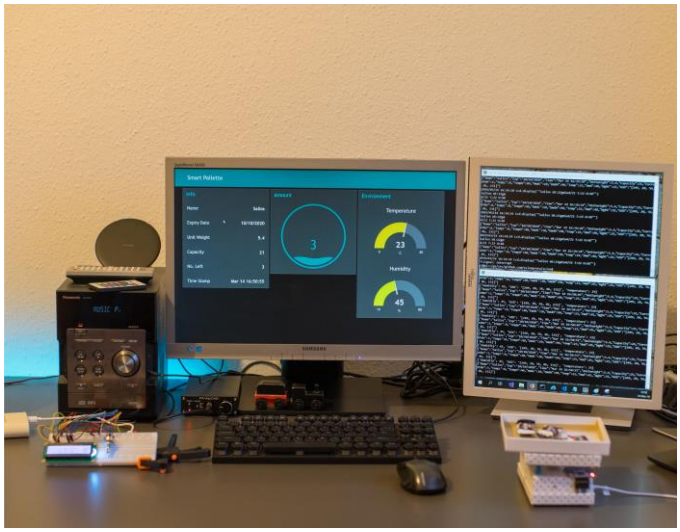

Fig 7. Total implementation of the project

C. GUI Implementation

The contents viewable at the LCD has been visualized on the virtual machine, kept as local client AS. As mentioned on the introduction part, the flow based, visual programming development tool, NODE-Red has been employed for this purpose. Node-Red requires data in the JSON Object format so that the desired flow can be developed using nodes. Thus, the data packets are transferred over the SCION network as JSON objects. This acts as a GUI for the users so that they can monitor the inventory status of products in real time which eliminates the need to do frequent physical inventory inspection and take business decisions accordingly. This also facilitates the users to take business decisions either to replenish the decreasing stock or to grant discounts in real time.

Fig 8. NODE Red dashboard

IX. CONCLUSION AND FURTHER WORK

This scientific work presented a basic concept to implement an IoT application over SCION network. Though the work is a micro model of Real World Retail Offer Management Systems, an extension of this approach is possible by bringing state of the art displays, auto announcement of discounts after the weight of a product drops to last few kilograms of the stock, had time not been a constraint for the project.

Another possible development to the project is the inclusion of temperature sensors to the module so that it can be made available to cold storage sections as well. Thus, the temperature of cold storage can be monitored 24 x 7 so that alerts could be given incase if the temperature rises above a limit or power failure occurs in the storage, in turn reducing loss.

During the initial stages of the project, idea was broader and complex than the present implementation. Initially, the plan was to have a pair of raspberry pi and virtual machine each where one team reads the load cell and sensor data through the raspberry pi, set as server I and sent to a virtual machine, set as client I. Then team 2 would be reading the same data to another virtual machine, set as server II which eventually is passed on to the LCD attached to raspberry pi, acting as client II. This process involved four steps and the final implementation reduced the complexity to two steps.

The existing implementation can only support maximum of 16 characters per line. If the idea of scrolling text is included, then there will be choice of including infinite number of characters/ texts over each line, increasing the clarity of the product details.

REFERENCES

[1] Adrian Perrig, Pawel Szalachowski, Raphael M Reischuk and Laurent Chuat, "SCION: A Secure Internet Architecture", ETH Zurich, Aug 2017. [Online]Available: https://www.scion-architecture.net/pdf/SCION-book.pdf

[2] SCION tutorials, Welcome to SCION Tutorials. Available: https://netsec-ethz.github.io/scion-tutorials/

[3]URL:https://netsecethz.github.io/sciontutorials/native_setup/image_builder

[4] SCIONLab Coordination Service, SCION is the first clean-slate Internet architecture designed to provide route control,failure isolation, and explicit trust information for end-to-end communication. Available: https://www.scionlab.org

[5] David Barrera, Laurent Chuat, Adrian Perrig, Raphael M Reischuk and Pawel Szalachowski, "The SCION Internet Architecture", ETH Zurich, Mar 2017. [Online]Available: https://www.scion-architecture.net/pdf/2017-SCION-CACM.pdf

[6]URL:https://www.amazon.de/Zeichen-Display-HD44780-Backlight-Arduino-Blau/dp/B009GEPZRE

[7]URL:https://www.sparkfun.com/datasheets/LCD/HD44780.pdf

[8]URL:https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

[9] Raspberry Pi - A small and affordable computer that you can use to learn programming. Available: https://www.raspberrypi.org/

[10] Node-Red, Flow-based programming for Internet of things. Available: https://nodered.org/

[11] Go, The Go Programming Language. Available: https://golang.org/

[12] docker, Build, Ship, and Run Any App, Anywhere. Learn about the only enterprise-ready container platform to cost effectively build and manage your application Available: https://www.docker.com/

[13] Wireshark, Wireshark is the world's foremost and widely used network protocol Analyzer. Available: https://www.wireshark.org/

[14] Raspbian, Raspbian is the Foundation's official supported operating system. Available: https://www.raspberrypi.org/downloads/raspbian/

[15] Ubuntu Mate for Raspberry Pi 2 and 3, Available: https://ubuntu-mate.org/raspberry-pi/

[16] Go, The Go Programming Language. Available: https://golang.org/

APPENDIX

A. Method 2 to install SCION on Raspberry Pi 3B+

- After downloading *SCIONLab AS Configuration*, download Raspbian Strech Lite from *https://downloads.raspberrypi.org/raspbian_lite_latest*, unzip the image and burn it to SD card of raspberry pi using Etcher
- To enable SSH login to the device, place a file with name *ssh* to the SD card and connect the pi over Ethernet to a network
- Boot up the device and login to the pi using PUTTY terminal with *username* and *password* as *pi* and *raspberry* respectively. Ensure that the same IP address found using Wireshark [13] is used to login to the Raspberry Pi.
- Install gitcurl and docker using the commands *sudo apt install git curl* and *curl -fsSL get.docker.com -o get-docker.sh && sh get-docker.sh*
- Add docker [12] user using the command *sudo gpasswd -a $USER docker; newgrp docker*
- Configure scion using the commands: *git config –global url.https://github.com/.insteadOf git@github.com:* and *git1clone–recursive git@github.com:scionproto/scion* and *cd scion*
- This is followed by creation and building of docker base image using the commands *./docker.sh base* and *./docker.sh build*
- Extract *SCION Lab AS Configuration* to */home/pi* and install OpenVPN using the command *sudo apt install openvpn*
- Copy *client.conf* file from SCIONLab AS configuration to */etc/openvpn* and start openvpn using the command *sudo service openvpn@client start*
- Finally, run docker using the command *DOCKER_ARGS="v/home/pi/19_ffaa:1_15d/gen:/home/scion/go/src/github.com/scionproto/scion/gen --network host" ./docker.sh run #* and run SCION using the command *./scion.sh start* (The AS configuration name varies as created on www.scionlab.org)

B. Implementation of JSON Object at Client side

//decode the json object and store to the data which is a variable of structure 'Message'

```
err = json.Unmarshal(receivePacketBuffer[:n], &data)
if err == nil
{
infoDisp := data.Name + " Wt:" + strconv.Itoa(data.Wght) +
"gm\n" + strconv.Itoa(data.CurrentNo) + "/" +
```

```
strconv.Itoa(data.Capacity) + " T:" + strconv.Itoa(data.Temp)
+ " H:" + strconv.Itoa(data.Hmd)

fmt.Println(infoDisp)
disp.Display(string(infoDisp))
    }

fmt.Println(string(receivePacketBuffer[:n]))
time.Sleep(time.Second)
```