

# **IOT PROJECT BASED LAB (PBL)**

## **END REPORT:**

**Module:** School Children Attendance Monitoring

**Project:** School-Aged Child Monitoring System

### **Team Members:**

D Sai Kushal Reddy - 207219

Akuthota Ravikanth - 207205

Ahmad Baasim Hussain-207203

### **Sensor Network:**

- We have used contiki for creating the Wireless sensor network for our module.
- The sensors(active rfid tags) used in our module transmit their tag ids to the rpl-border-router. This data is transmitted by the border router and stored in the blockchain.
- We have written Project.c which simulates the working of the sensors by sending data to the router for every 5 seconds.

### **Hyperledger Fabric:**

- We have used Hyperledger fabric for security purpose. It is used to store the sensor data properly.

- We have written demo.js script to fetch the data and submit to Hyperledger network.
- We have written assetTransfer.js, which is the chaincode for our network.
- For every new data object received, demo.js first checks if the corresponding asset is already present in the Hyperledger.
- If the data is not present then it creates a new asset or else it updates the existing asset.

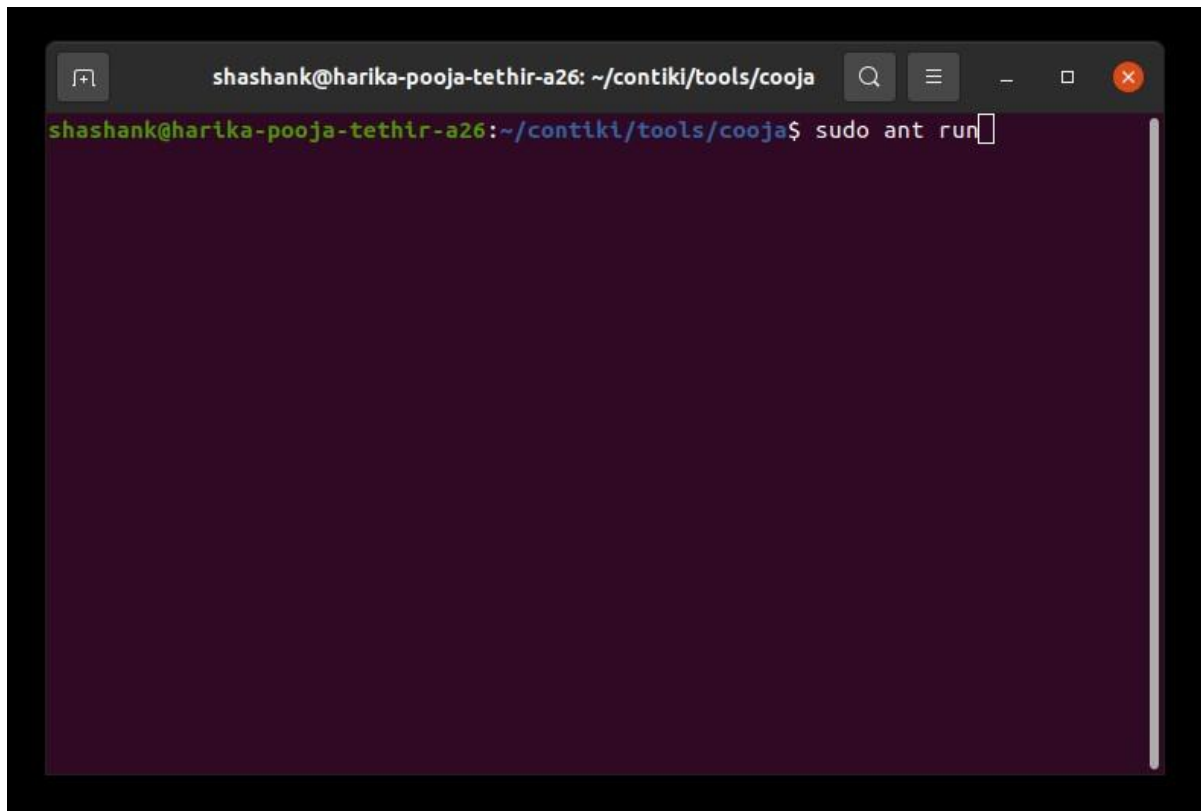
## Dashboard:

- A simple html file name dashboard.html is written, which acts as a dashboard webpage.
- A javascript named demo2.js is written. This script acts as an api end point which fetches all the assets in the Hyperledger and sends it.
- The file dashboard.html has some javascript embedded in it which fetches all the assets from api endpoint on loading and displays the tag id and their last time seen.

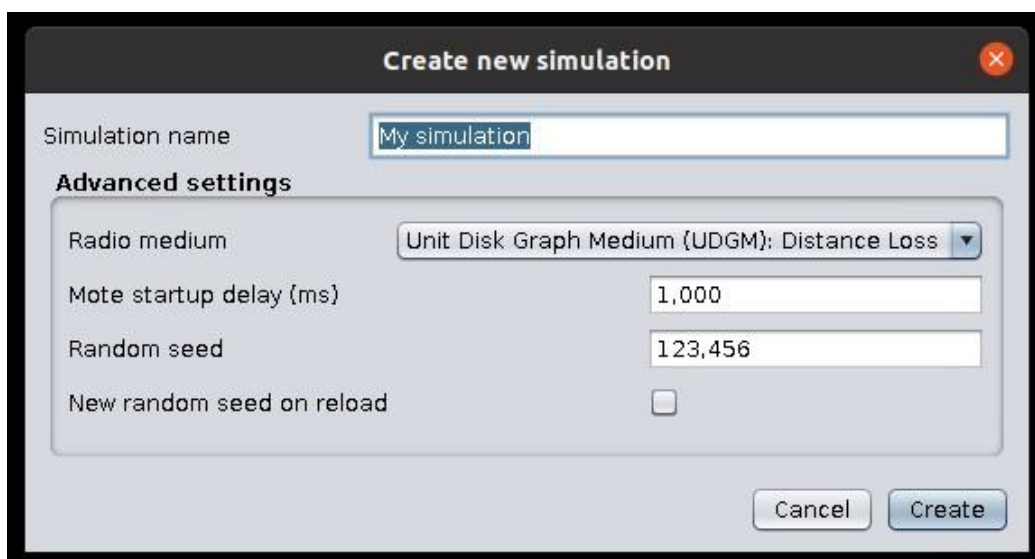
## STEPS INVOLVED:

### Run the Network:

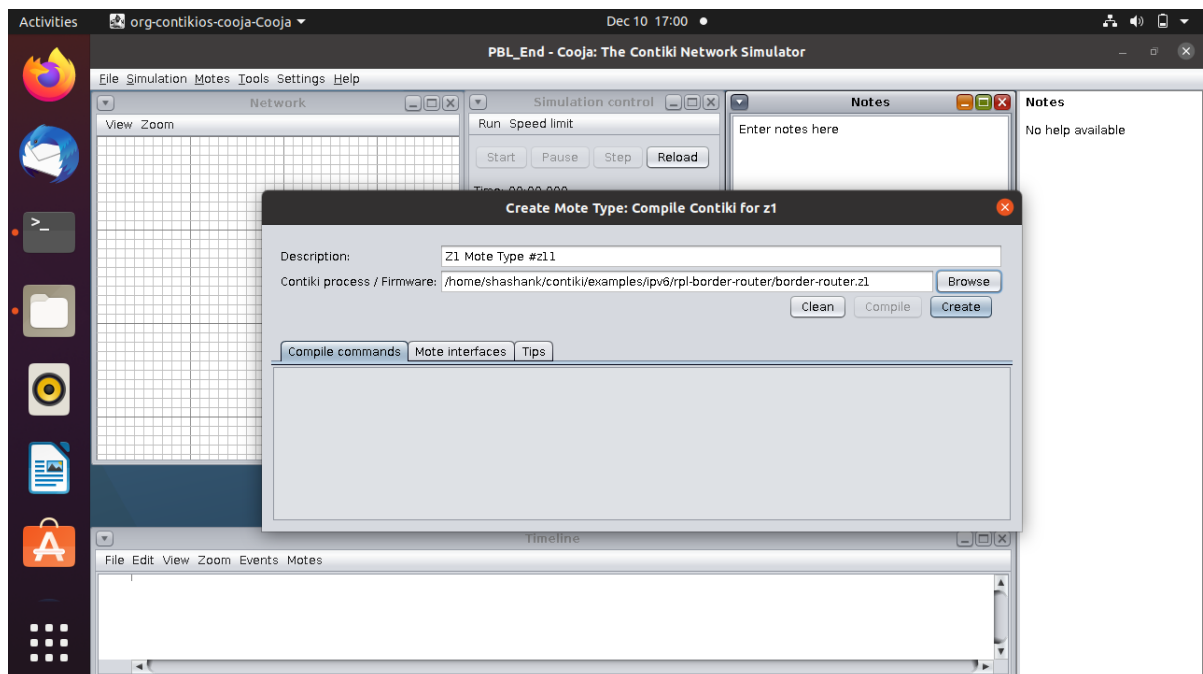
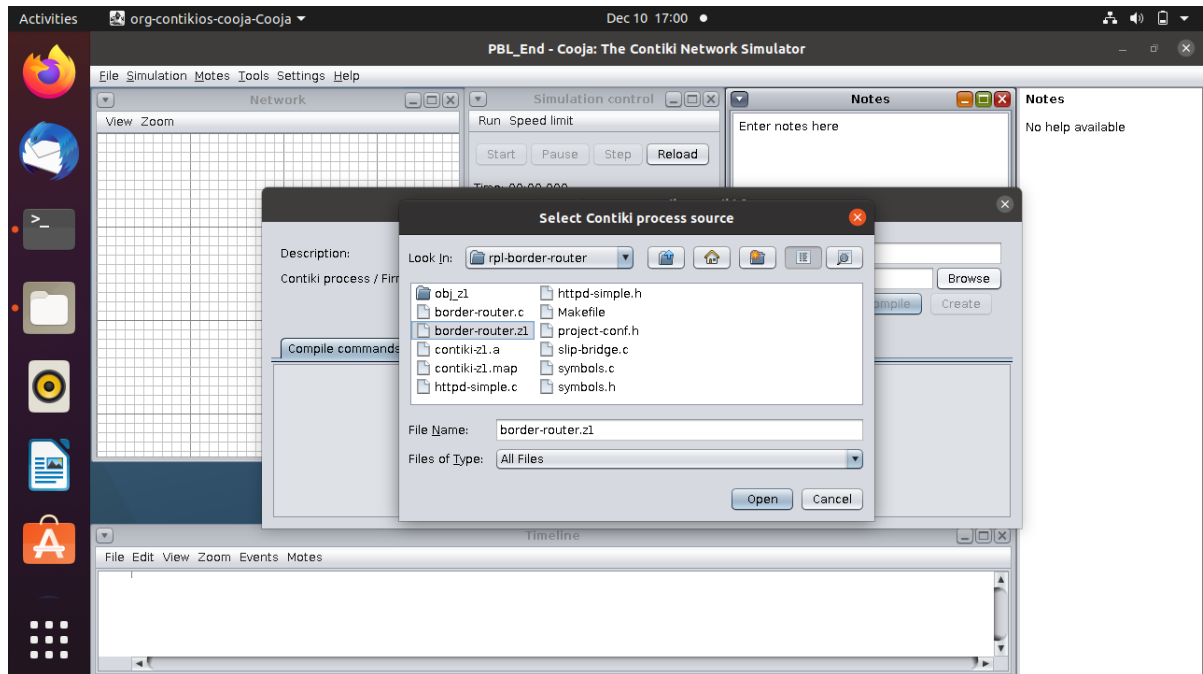
1. Open cooja simulator



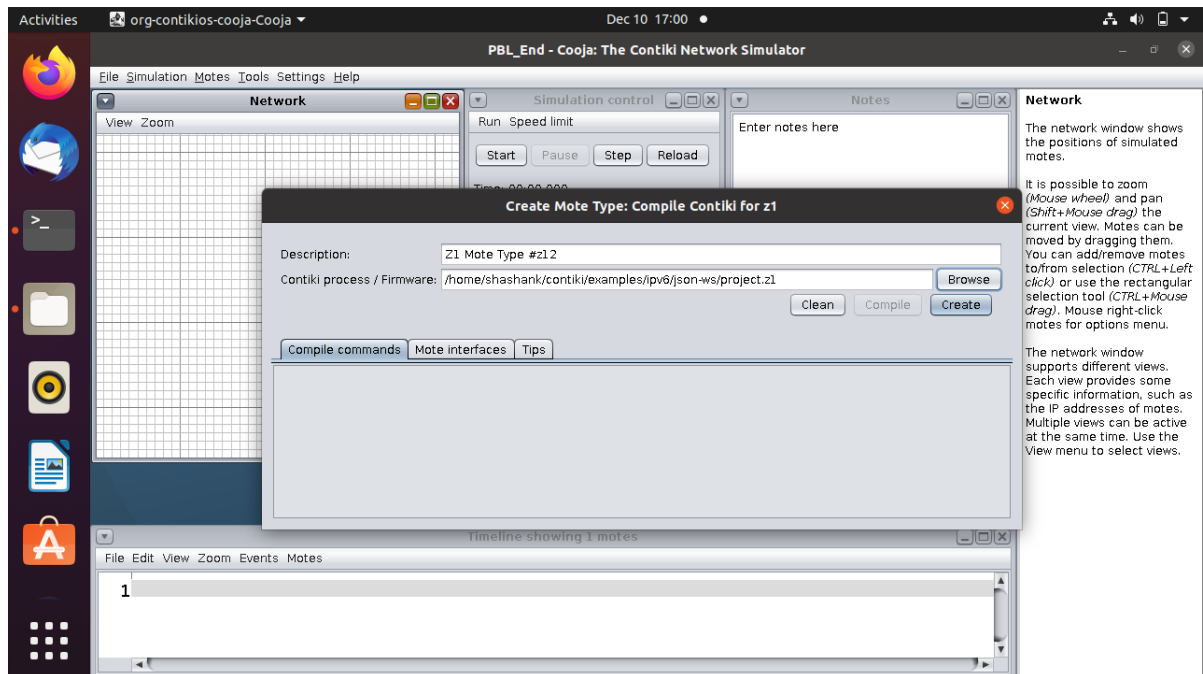
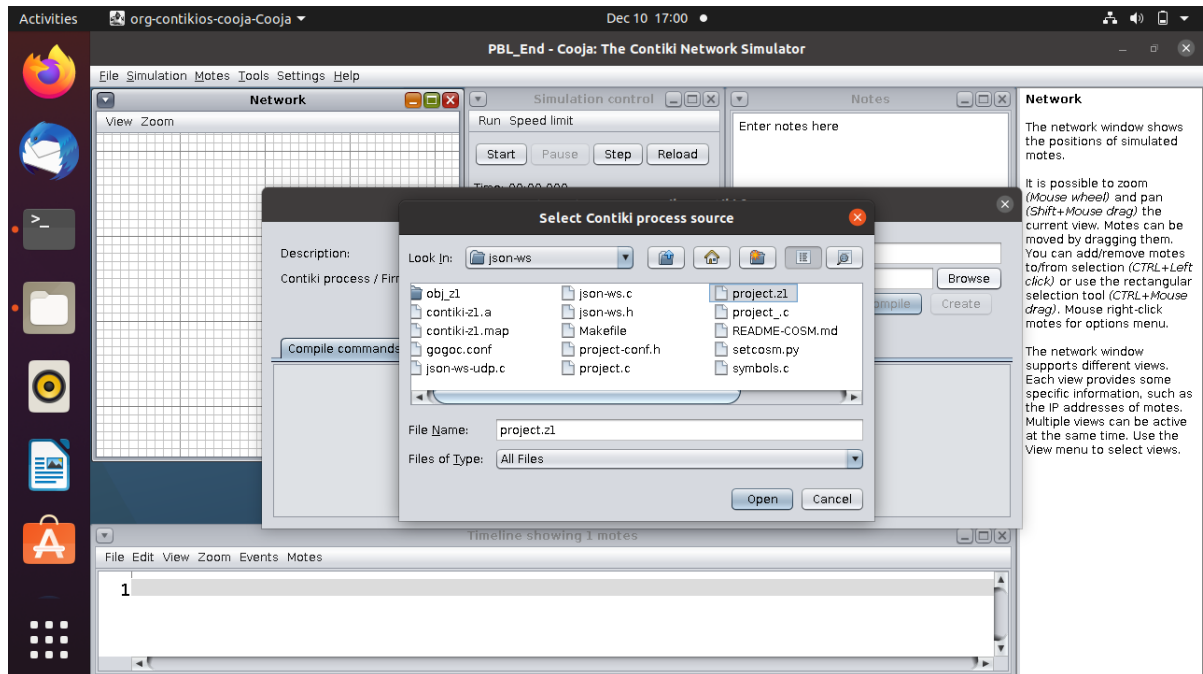
2. Create new simulation

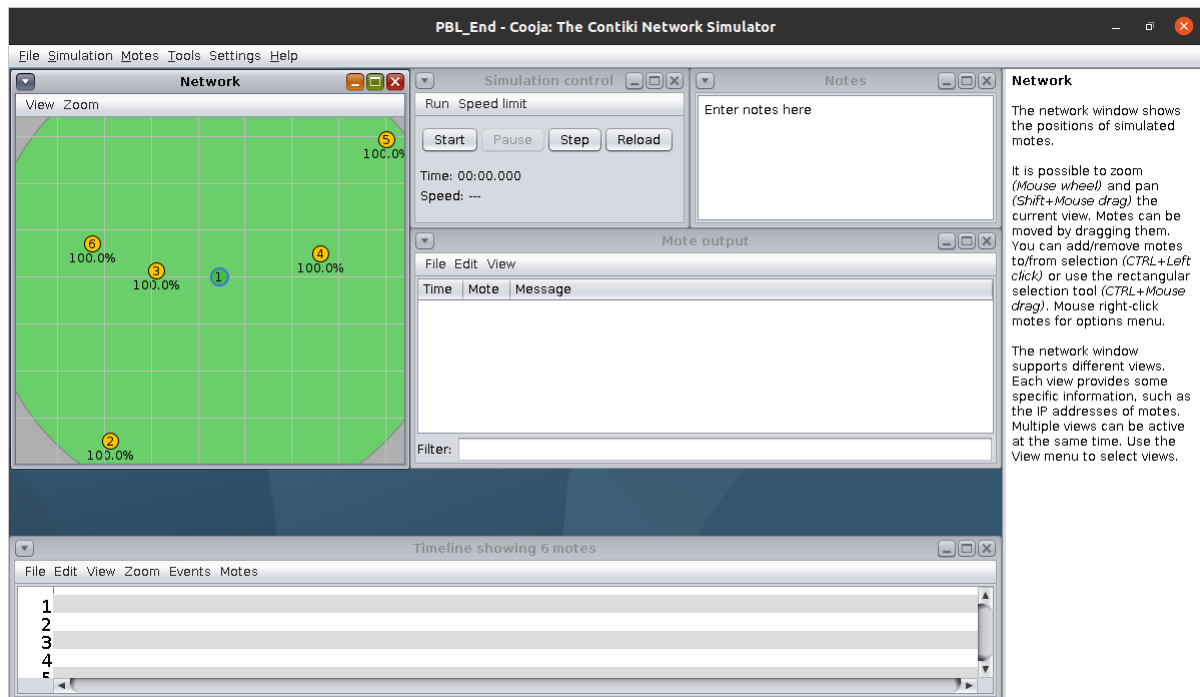


### 3. Create RPL border router, set it as server.

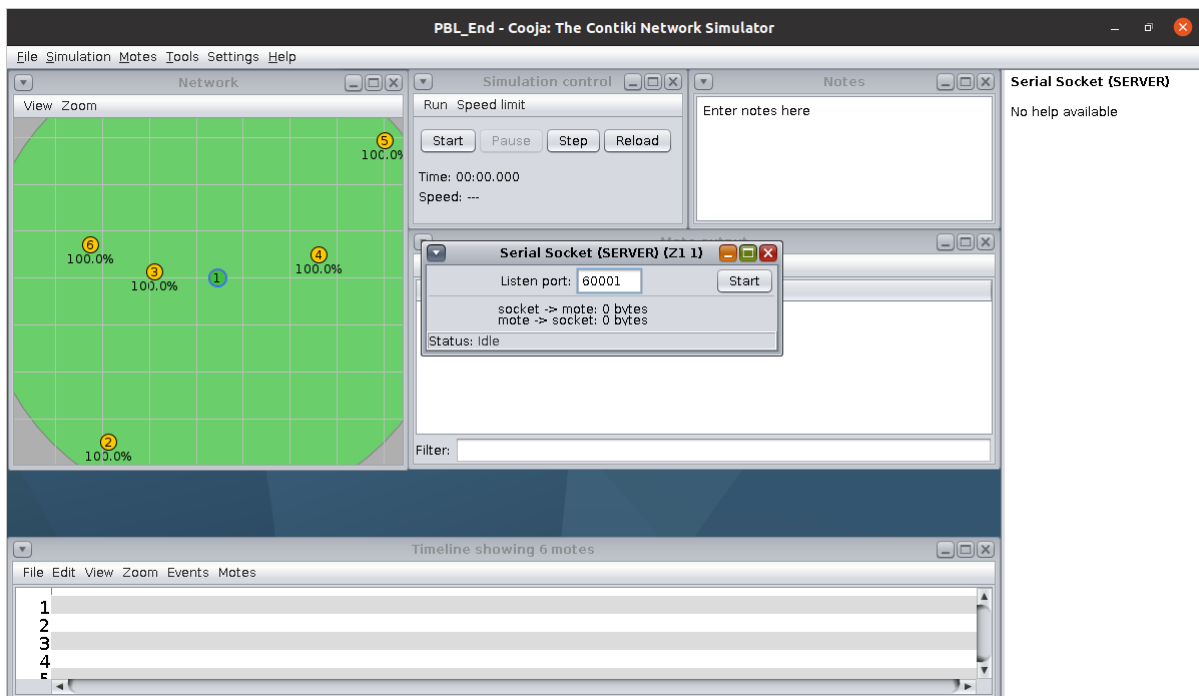


#### 4. Create 5 nodes of z1 mote type from Project.c

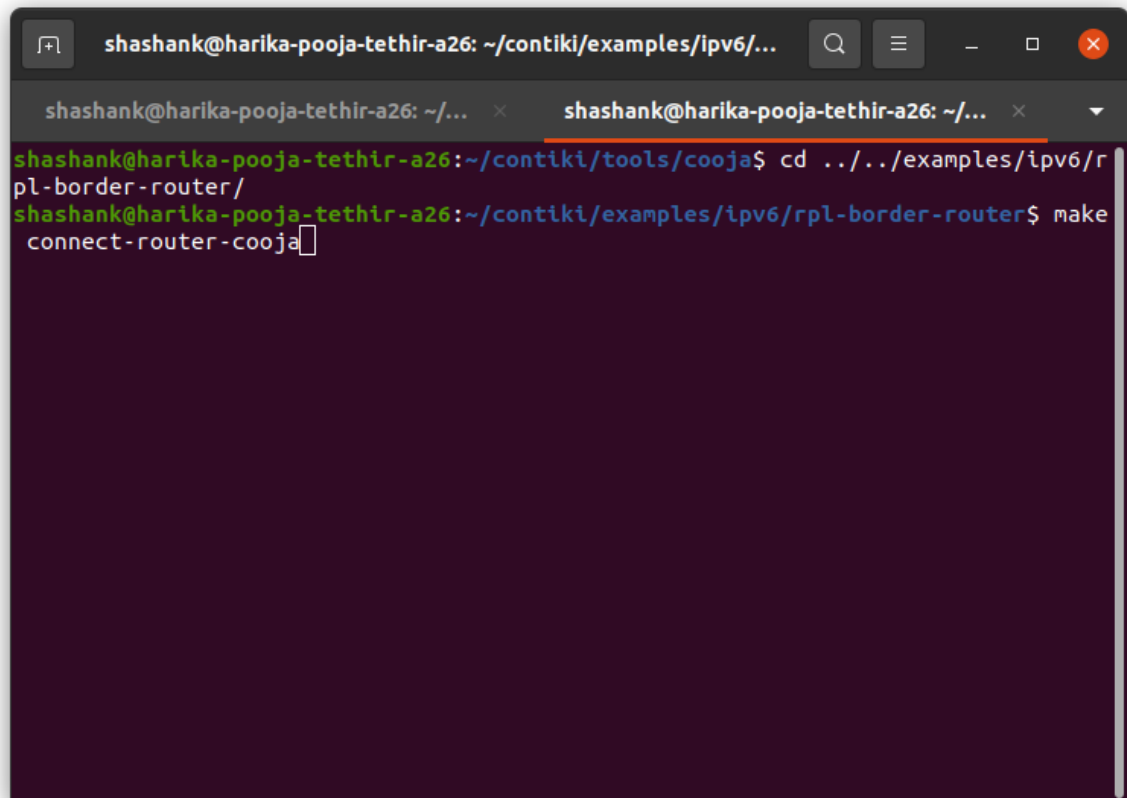




## 5. Create serial socket server for rpl-border-router



6. Run the tunslip command in new terminal



```
shashank@harika-pooja-tethir-a26: ~/contiki/examples/ipv6/...  
shashank@harika-pooja-tethir-a26: ~/... x shashank@harika-pooja-tethir-a26: ~/... x  
shashank@harika-pooja-tethir-a26:~/contiki/tools/cooja$ cd ../../examples/ipv6/r  
pl-border-router/  
shashank@harika-pooja-tethir-a26:~/contiki/examples/ipv6/rpl-border-router$ make  
connect-router-cooja
```

## 7. Start the simulation.

The screenshot shows the Cooja network simulator interface. The main window displays a network topology with six nodes (1-6) connected in a mesh. Node 1 is the central node, connected to nodes 2, 3, 4, 5, and 6. All links show 100.0% signal strength. The 'Simulation control' panel on the right includes buttons for 'Start', 'Pause', 'Step', and 'Reload', along with a 'Speed limit' slider. The 'Mote output' panel shows a log of messages, including 'Rime started with address 193.12.0.0.0.0.3', 'MAC c1:0c:00:00:00:00:03 Contiki 3.0 started. Node id ...', 'CSMA nullrdc, channel check rate 128 Hz, radio channel 26', 'Tentative link-local IPv6 address fe80:0000:0000:0000:c30...', 'Starting 'Websense (sky)', 'Server IPv6 addresses: aaaa::c30c:0:0:1', and 'fe80::c30c:0:0:1'. The 'Control Panel' on the far right provides instructions for the simulation controls. The 'Timeline showing 6 motes' at the bottom displays the execution timeline for each node.

**Simulation control**

Run Speed limit

Start Pause Step Reload

Time: 00:27.824  
Speed: 426.14%

**Mote output**

| Time      | Mote | Message  |
|-----------|------|--|
| 00:01.761 | ID:3 | Rime started with address 193.12.0.0.0.0.3                   |
| 00:01.768 | ID:3 | MAC c1:0c:00:00:00:00:03 Contiki 3.0 started. Node id ...    |
| 00:01.773 | ID:3 | CSMA nullrdc, channel check rate 128 Hz, radio channel 26    |
| 00:01.781 | ID:3 | Tentative link-local IPv6 address fe80:0000:0000:0000:c30... |
| 00:01.783 | ID:3 | Starting 'Websense (sky)'                                    |
| 00:02.265 | ID:1 | Server IPv6 addresses:                                       |
| 00:02.267 | ID:1 | aaaa::c30c:0:0:1   |
| 00:02.269 | ID:1 | fe80::c30c:0:0:1   |

**Control Panel**

The control panel controls the simulation.

Start starts the simulation.

Pause stops the simulation.

The keyboard shortcut for starting and pausing the simulation is **Ctrl+S**.

Step runs the simulation for one millisecond.

Reload reloads and restarts the simulation.

Simulation speed is controlled via the Speed limit menu.

```
shashank@harika-pooja-tethir-a26: ~/contiki/examples/ipv6/...  
shashank@harika-pooja-tethir-a26: ~/... x shashank@harika-pooja-tethir-a26: ~/... x  
shashank@harika-pooja-tethir-a26:~/contiki/tools/cooja$ cd ../../examples/ipv6/rpl-border-router/  
shashank@harika-pooja-tethir-a26:~/contiki/examples/ipv6/rpl-border-router$ make  
connect-router-cooja  
TARGET not defined, using target 'native'  
fatal: not a git repository: '../..../.git'  
sudo ../../tools/tunslip6 -a 127.0.0.1 aaaa::1/64  
[sudo] password for shashank:  
slip connected to `127.0.0.1:60001'  
opened tun device `/dev/tun0'  
ifconfig tun0 inet `hostname` up  
ifconfig tun0 add aaaa::1/64  
ifconfig tun0 add fe80::0:0:0:1/64  
ifconfig tun0  
  
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500  
inet 10.0.2.15 netmask 255.255.255.255 destination 10.0.2.15  
inet6 aaaa::1 prefixlen 64 scopeid 0x0<global>  
inet6 fe80::1 prefixlen 64 scopeid 0x20<link>  
inet6 fe80::ed1b:d3cb:76c0:9ca3 prefixlen 64 scopeid 0x20<link>  
unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500  
(UNSPEC)  
RX packets 0 bytes 0 (0.0 B)  
RX errors 0 dropped 0 overruns 0 frame 0
```



```
shashank@harika-pooja-tethir-a26: ~/contiki/examples/ipv6/...
shashank@harika-pooja-tethir-a26: ~/... x shashank@harika-pooja-tethir-a26: ~/... x
tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
  inet 10.0.2.15 netmask 255.255.255.255 destination 10.0.2.15
  inet6 aaaa::1 prefixlen 64 scopeid 0x0<global>
  inet6 fe80::1 prefixlen 64 scopeid 0x20<link>
  inet6 fe80::ed1b:d3cb:76c0:9ca3 prefixlen 64 scopeid 0x20<link>
  unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 500
(UNSPEC)
  RX packets 0 bytes 0 (0.0 B)
  RX errors 0 dropped 0 overruns 0 frame 0
  TX packets 0 bytes 0 (0.0 B)
  TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

Rime started with address 193.12.0.0.0.0.1
MAC c1:0c:00:00:00:00:01 Contiki 3.0 started. Node id is set to 1.
CSMA ContikiMAC, channel check rate 8 Hz, radio channel 26
Tentative link-local IPv6 address fe80:0000:0000:0000:c30c:0000:0000:0001
Starting 'Border router process' 'Web server'
*** Address:aaaa::1 => aaaa:0000:0000:0000
Got configuration message of type P
Setting prefix aaaa::
Server IPv6 addresses:
  aaaa::c30c:0:0:1
  fe80::c30c:0:0:1
```

8. Open the Ipv6 address to see the data generated

```
ContikiRPL x +
[aaaa::c30c:0:0:1]
Neighbors
fe80::c30c:0:0:6
fe80::c30c:0:0:3
fe80::c30c:0:0:4
fe80::c30c:0:0:5
fe80::c30c:0:0:2
Routes
aaaa::c30c:0:0:5/128 (via fe80::c30c:0:0:4) 16711369s
aaaa::c30c:0:0:3/128 (via fe80::c30c:0:0:3) 16710950s
aaaa::c30c:0:0:4/128 (via fe80::c30c:0:0:4) 16710949s
aaaa::c30c:0:0:6/128 (via fe80::c30c:0:0:6) 16710950s
aaaa::c30c:0:0:2/128 (via fe80::c30c:0:0:6) 16711323s
```

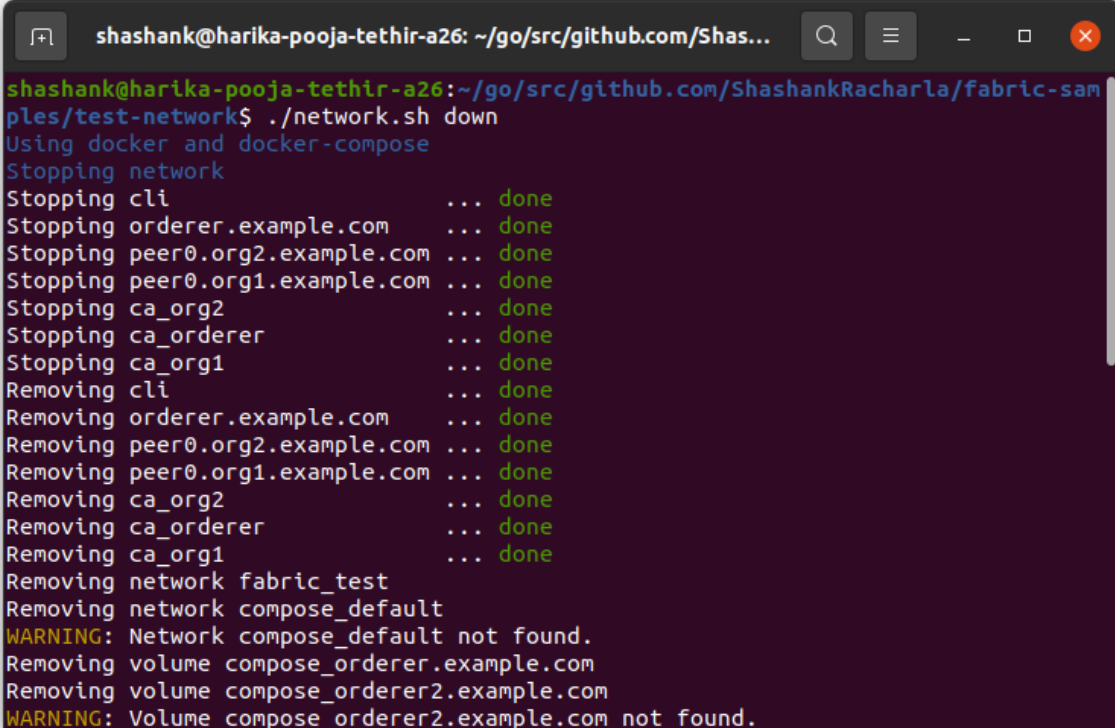
Project.c file simulates the working of rfid active tag by generating data objects corresponding to its id.

## Push the data into Hyperledger:

1. Open test network in fabric samples and run the following command.

`./network.sh down`

- Remove all the existing docker images related to Hyperledger.

A terminal window with a dark purple background. The title bar shows the user 'shashank@harika-pooja-tethir-a26' and the current directory '~/go/src/github.com/Shas...'. The terminal output shows the command './network.sh down' being executed. It starts with 'Using docker and docker-compose' and 'Stopping network'. A list of services is shown being stopped, each followed by '... done'. Then, a list of services is shown being removed, each followed by '... done'. Finally, it shows 'Removing network fabric\_test' and 'Removing network compose\_default'. There are two 'WARNING' messages: 'WARNING: Network compose\_default not found.' and 'WARNING: Volume compose\_orderer2.example.com not found.'.

```
shashank@harika-pooja-tethir-a26: ~/go/src/github.com/Shas...
shashank@harika-pooja-tethir-a26:~/go/src/github.com/ShashankRacharla/fabric-samples/test-network$ ./network.sh down
Using docker and docker-compose
Stopping network
Stopping cli ... done
Stopping orderer.example.com ... done
Stopping peer0.org2.example.com ... done
Stopping peer0.org1.example.com ... done
Stopping ca_org2 ... done
Stopping ca_orderer ... done
Stopping ca_org1 ... done
Removing cli ... done
Removing orderer.example.com ... done
Removing peer0.org2.example.com ... done
Removing peer0.org1.example.com ... done
Removing ca_org2 ... done
Removing ca_orderer ... done
Removing ca_org1 ... done
Removing network fabric_test
Removing network compose_default
WARNING: Network compose_default not found.
Removing volume compose_orderer.example.com
Removing volume compose_orderer2.example.com
WARNING: Volume compose_orderer2.example.com not found.
```

```
shashank@harika-pooja-tethir-a26: ~/go/src/github.com/Shas...
Removing volume compose_peer0.org3.example.com
WARNING: Volume compose_peer0.org3.example.com not found.
Error response from daemon: get docker_orderer.example.com: no such volume
Error response from daemon: get docker_peer0.org1.example.com: no such volume
Error response from daemon: get docker_peer0.org2.example.com: no such volume
Removing remaining containers
Removing generated chaincode docker images
Untagged: dev-peer0.org2.example.com-basic_1.0.1-4a41162f84fab36f9d3b119f096f836
234d29ae3e8f895510329c3a0d5c5c7d3-ffdee96c9722c18962f8ae1e53f7eae65c70a506cfe37d
93282a7bbd8e1602fe:latest
Deleted: sha256:666f23fb0a58c2b048f4de88d56e0b2ded87d8291bea8e458066296d750f6fe7
Deleted: sha256:3545fa3b22ed3c750c8d489d6d8d3e69917b1bbbd115ef4496eed392edc59d47
Deleted: sha256:4b683404315176d81f02ef0d7e90816f6c003cbfe5f9e8c96d201594d3e5cc6d
Deleted: sha256:6522843645c6307ad5e48daff41457a78621d95672d1c32fdeab171aeb5a8cef
Untagged: dev-peer0.org1.example.com-basic_1.0.1-4a41162f84fab36f9d3b119f096f836
234d29ae3e8f895510329c3a0d5c5c7d3-6a2c75b4c80235215c015a706afd110960cef916ec485a
9ac31b7f0db5f728ff:latest
Deleted: sha256:060354542d0c38f92db4497bd430af4a5b9549859358eab8222fcfd1ac6b7b1
Deleted: sha256:4d95bc2999b469c6d6fffd35b31337929a31a76787800026d2425d57ff6de52c
Deleted: sha256:05675787b52d339d03b81d95c507de7e1d08dd9b1f332346c6dc3f7dc0d13a28
Deleted: sha256:ac2ef10cc1ba699af6dbcaa2d7fd8f8a882dd341f845aaa09ff5f23ee0cc541a
shashank@harika-pooja-tethir-a26:~/go/src/github.com/ShashankRacharla/fabric-sam
ples/test-network$ ./network.sh up createChannel -c mychannel -ca
```

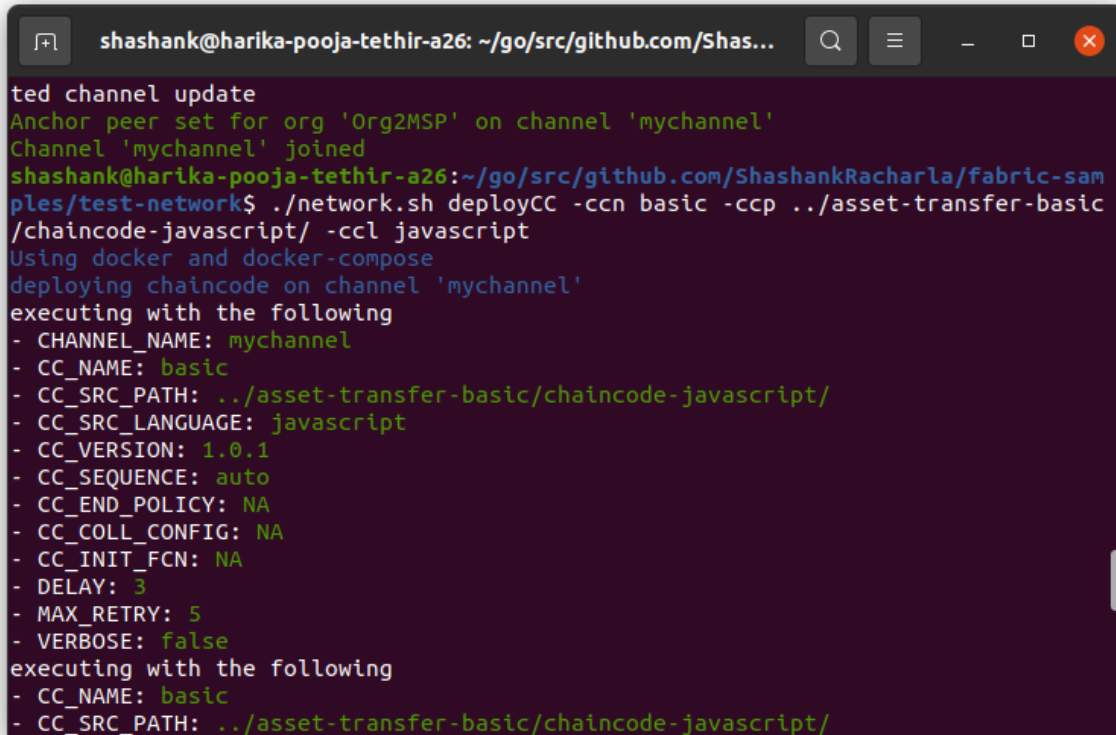
2. Now in the same folder create network and channel  
./network.sh up createChannel -c myChannel -ca

- The above command starts 2 peer nodes in the network and will be joined to a common channel called “mychannel”.
- We have written the app.js and assetTransfer.js such that the app.js fetches the data from the Ipv6 addresses of the sensors from simulated in contiki and invokes the chaincode – assetTransfer.js to create an asset with some fields and values.
- In the assetTransfer, the parameters are received and a new record is created with unique Row ID value (rid).



3. Now in the same folder deploy the chain code to all the peers in the network by running the following command

```
./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-javascript/ -ccl javascript
```

A terminal window with a dark background and light-colored text. The window title is 'shashank@harika-pooja-tethir-a26: ~/go/src/github.com/Shas...'. The terminal output shows the execution of a script to deploy chaincode. It starts with 'ted channel update', followed by 'Anchor peer set for org 'Org2MSP' on channel 'mychannel'', and 'Channel 'mychannel' joined'. Then, the command './network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-javascript/ -ccl javascript' is entered. The output continues with 'Using docker and docker-compose', 'deploying chaincode on channel 'mychannel'', and 'executing with the following'. A list of environment variables follows: CHANNEL\_NAME: mychannel, CC\_NAME: basic, CC\_SRC\_PATH: ../asset-transfer-basic/chaincode-javascript/, CC\_SRC\_LANGUAGE: javascript, CC\_VERSION: 1.0.1, CC\_SEQUENCE: auto, CC\_END\_POLICY: NA, CC\_COLL\_CONFIG: NA, CC\_INIT\_FCN: NA, DELAY: 3, MAX\_RETRY: 5, VERBOSE: false. Finally, it says 'executing with the following' and lists CC\_NAME: basic and CC\_SRC\_PATH: ../asset-transfer-basic/chaincode-javascript/.

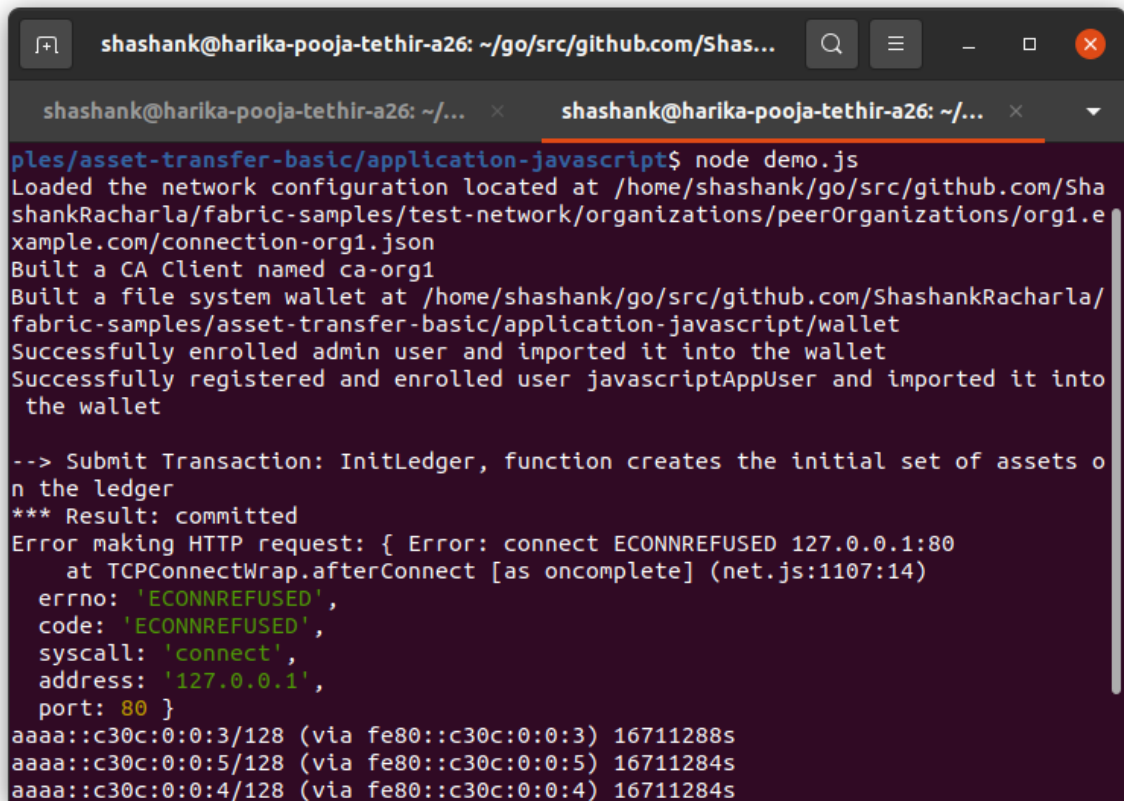
```
shashank@harika-pooja-tethir-a26: ~/go/src/github.com/Shas...
ted channel update
Anchor peer set for org 'Org2MSP' on channel 'mychannel'
Channel 'mychannel' joined
shashank@harika-pooja-tethir-a26:~/go/src/github.com/ShashankRacharla/fabric-samples/test-network$ ./network.sh deployCC -ccn basic -ccp ../asset-transfer-basic/chaincode-javascript/ -ccl javascript
Using docker and docker-compose
deploying chaincode on channel 'mychannel'
executing with the following
- CHANNEL_NAME: mychannel
- CC_NAME: basic
- CC_SRC_PATH: ../asset-transfer-basic/chaincode-javascript/
- CC_SRC_LANGUAGE: javascript
- CC_VERSION: 1.0.1
- CC_SEQUENCE: auto
- CC_END_POLICY: NA
- CC_COLL_CONFIG: NA
- CC_INIT_FCN: NA
- DELAY: 3
- MAX_RETRY: 5
- VERBOSE: false
executing with the following
- CC_NAME: basic
- CC_SRC_PATH: ../asset-transfer-basic/chaincode-javascript/
```

4. Now go to application-javascript and run the demo.js cmd: node demo.js

```
shashank@harika-pooja-tethir-a26: ~/go/src/github.com/Shas...
shashank@harika-pooja-tethir-a26: ~/... x shashank@harika-pooja-tethir-a26: ~/... x
shashank@harika-pooja-tethir-a26:~/go/src/github.com/ShashankRacharla/fabric-sam
ples/test-network$ cd ../asset-transfer-basic/application-javascript/
shashank@harika-pooja-tethir-a26:~/go/src/github.com/ShashankRacharla/fabric-sam
ples/asset-transfer-basic/application-javascript$
```

```
shashank@harika-pooja-tethir-a26: ~/go/src/github.com/Shas...
shashank@harika-pooja-tethir-a26: ~/... x shashank@harika-pooja-tethir-a26: ~/... x
shashank@harika-pooja-tethir-a26:~/go/src/github.com/ShashankRacharla/fabric-sam
ples/test-network$ cd ../asset-transfer-basic/application-javascript/
shashank@harika-pooja-tethir-a26:~/go/src/github.com/ShashankRacharla/fabric-sam
ples/asset-transfer-basic/application-javascript$ node demo.js
```

5. You can see the new data from the 5 sensors being pushed/committed into the ledger ( for demo purpose the data is pushed into the ledger for every 5 sec).



```
shashank@harika-pooja-tethir-a26: ~/go/src/github.com/Shas...  
shashank@harika-pooja-tethir-a26: ~/... x shashank@harika-pooja-tethir-a26: ~/... x  
ples/asset-transfer-basic/application-javascript$ node demo.js  
Loaded the network configuration located at /home/shashank/go/src/github.com/ShashankRacharla/fabric-samples/test-network/organizations/peerOrganizations/org1.example.com/connection-org1.json  
Built a CA Client named ca-org1  
Built a file system wallet at /home/shashank/go/src/github.com/ShashankRacharla/fabric-samples/asset-transfer-basic/application-javascript/wallet  
Successfully enrolled admin user and imported it into the wallet  
Successfully registered and enrolled user javascriptAppUser and imported it into the wallet  
  
--> Submit Transaction: InitLedger, function creates the initial set of assets on the ledger  
*** Result: committed  
Error making HTTP request: { Error: connect ECONNREFUSED 127.0.0.1:80  
  at TCPConnectWrap.afterConnect [as oncomplete] (net.js:1107:14)  
  errno: 'ECONNREFUSED',  
  code: 'ECONNREFUSED',  
  syscall: 'connect',  
  address: '127.0.0.1',  
  port: 80 }  
aaaa::c30c:0:0:3/128 (via fe80::c30c:0:0:3) 16711288s  
aaaa::c30c:0:0:5/128 (via fe80::c30c:0:0:5) 16711284s  
aaaa::c30c:0:0:4/128 (via fe80::c30c:0:0:4) 16711284s
```



```
shashank@harika-pooja-tethir-a26: ~/go/src/github.com/Shas...
shashank@harika-pooja-tethir-a26: ~/... x shashank@harika-pooja-tethir-a26: ~/... x
aaaa::c30c:0:0:4/128 (via fe80::c30c:0:0:4) 16711284s
aaaa::c30c:0:0:6/128 (via fe80::c30c:0:0:6) 16711281s
aaaa::c30c:0:0:2/128 (via fe80::c30c:0:0:2) 16711277s
2023-12-10T11:42:09.802Z
4a44dc153642
Created
*** Result: committed
*** Result: {
  "RID": "15",
  "SID": "1",
  "TagID": "4a44dc153642",
  "Time": "2023-12-10T11:42:09.802Z"
}
2023-12-10T11:42:19.833Z
5feceb66ffc8
Created
*** Result: committed
*** Result: {
  "RID": "5",
  "SID": "2",
  "TagID": "5feceb66ffc8",
  "Time": "2023-12-10T11:42:19.833Z"
}
```

```
shashank@harika-pooja-tethir-a26: ~/go/src/github.com/Shas...
shashank@harika-pooja-tethir-a26: ~/... x shashank@harika-pooja-tethir-a26: ~/... x
"RID": "25",
"SID": "0",
"TagID": "f5ca38f748a1",
"Time": "2023-12-10T11:43:39.940Z"
}
2023-12-10T11:43:50.030Z
ef2d127de37b
Updated
*** Result: committed
*** Result: {
  "type": "Buffer",
  "data": []
}
2023-12-10T11:44:00.706Z
e629fa6598d7
Updated
*** Result: committed
*** Result: {
  "type": "Buffer",
  "data": []
}
2023-12-10T11:44:09.865Z
f5ca38f748a1
```

The 5 sensors data is differentiated by the SID field which is used to identify the sensor number.

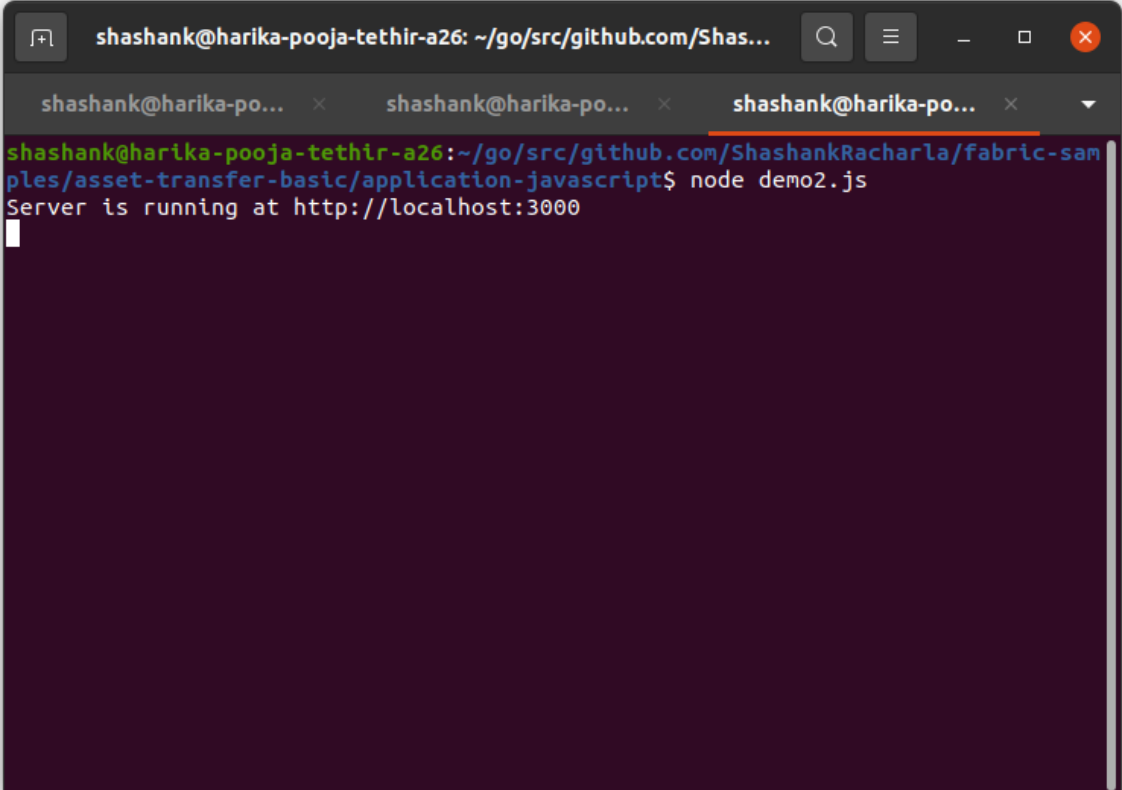


## Display data on Dashboard:

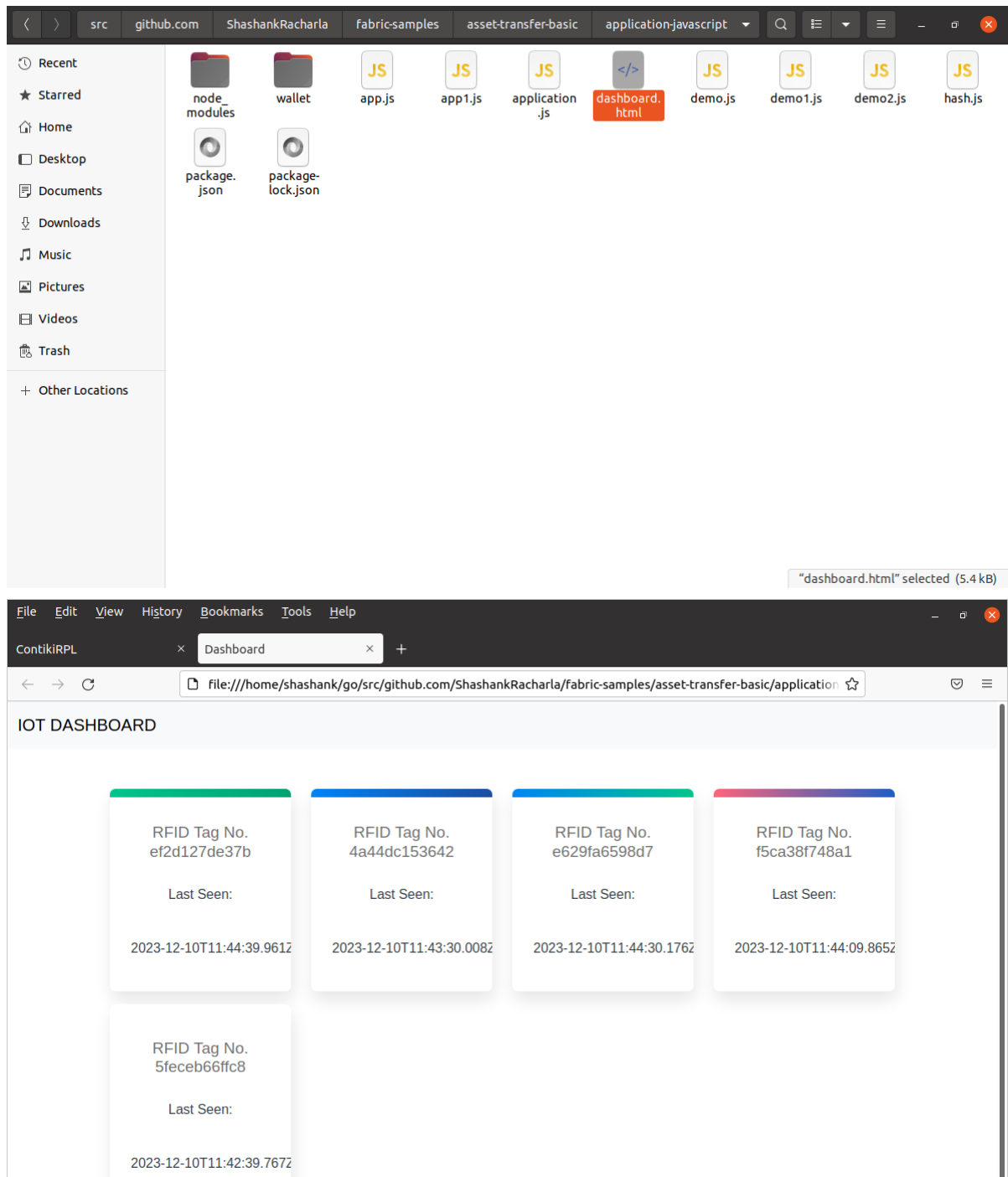
1. Open the new terminal from the same folder to run demo2.js file

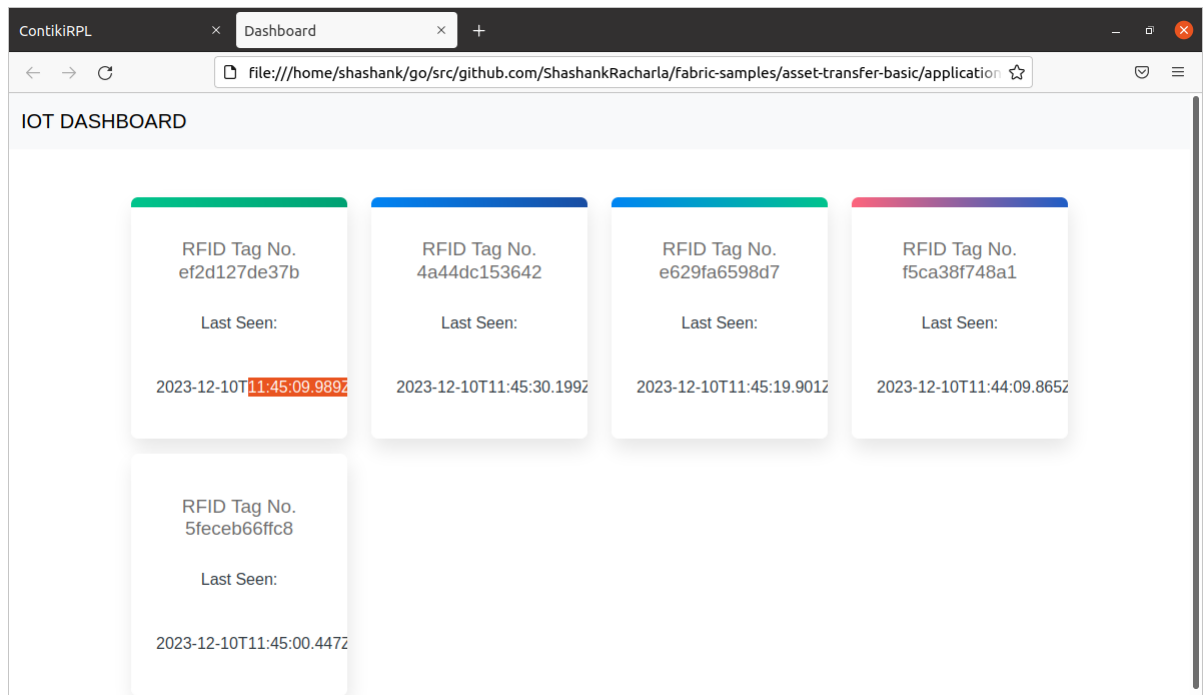
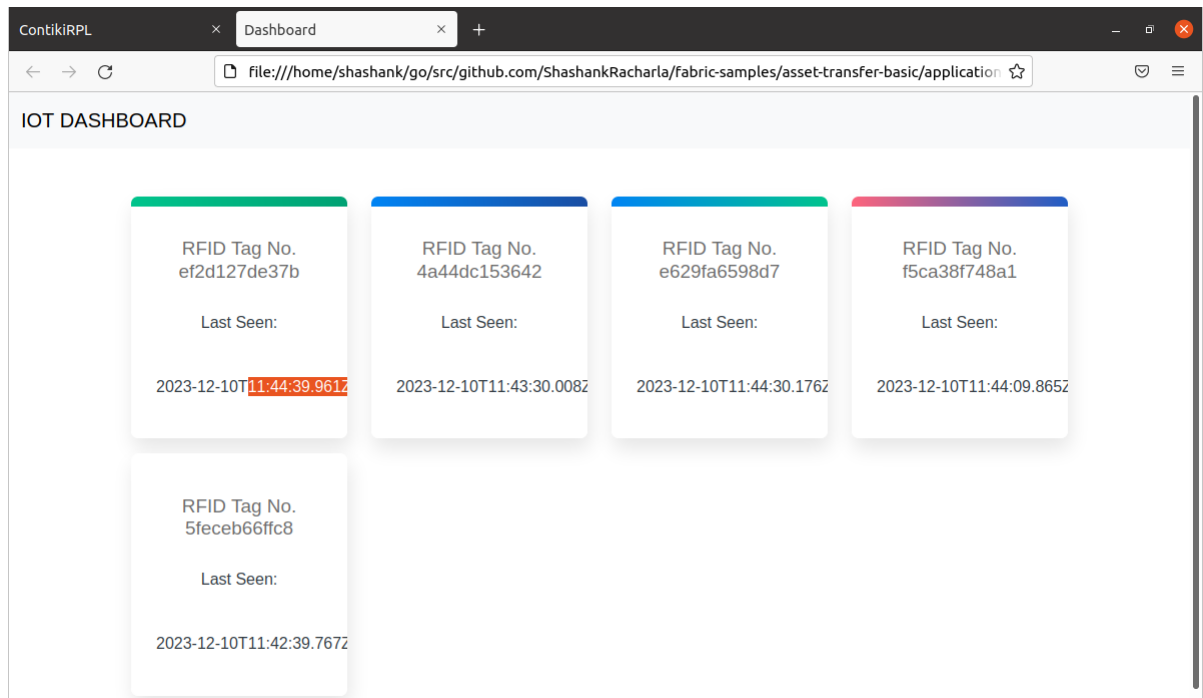
cmd: node demo2.js

This creates an api endpoint at <http://localhost:3000> which fetches all the asset data from the Hyperledger.

A screenshot of a terminal window with a dark background. The window title bar shows the user 'shashank@harika-pooja-tethir-a26' and the current directory '~/go/src/github.com/Shas...'. The terminal shows the command 'node demo2.js' being executed, and the output 'Server is running at http://localhost:3000'.

2. Open dashboard.html file in the browser
  - The javascript embedded in the file fetches data from the api endpoint <http://localhost:3000> on loading each time and displays the data.
  - The data updates every time after reloading the page if new data corresponding to the sensor is committed to the Hyperledger.





## **The Objectives achieved:**

- 1) Creating a WSN for the sensors.
- 2) Pushing the data into the Hyperledger.
- 3) Displaying the real time data on the dashboard.