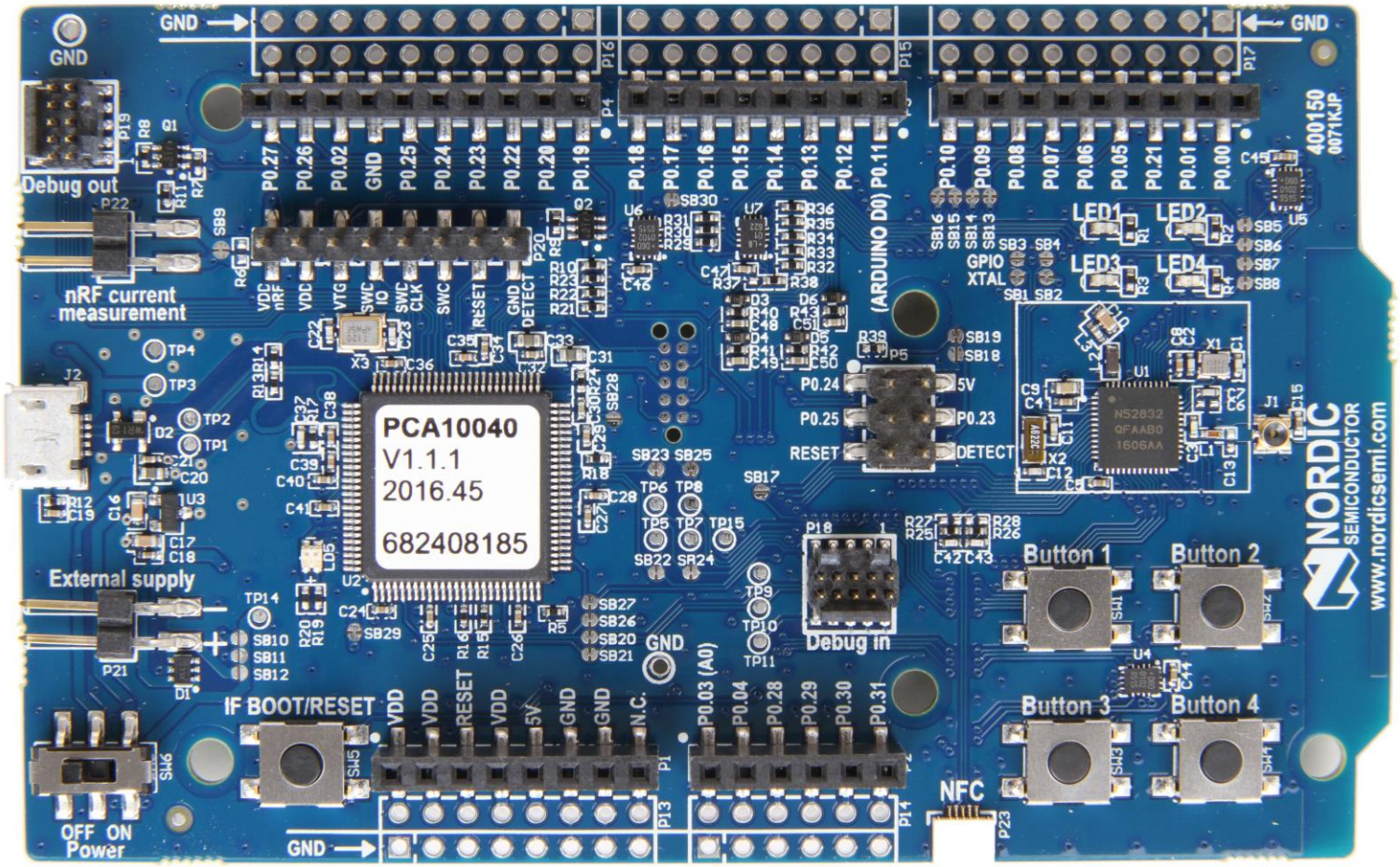
Easy TCP/IP for IoT



Implementing TCP/IP with a Nordic nRF52832

October 22, 2019

Fred Eady

Presented by:

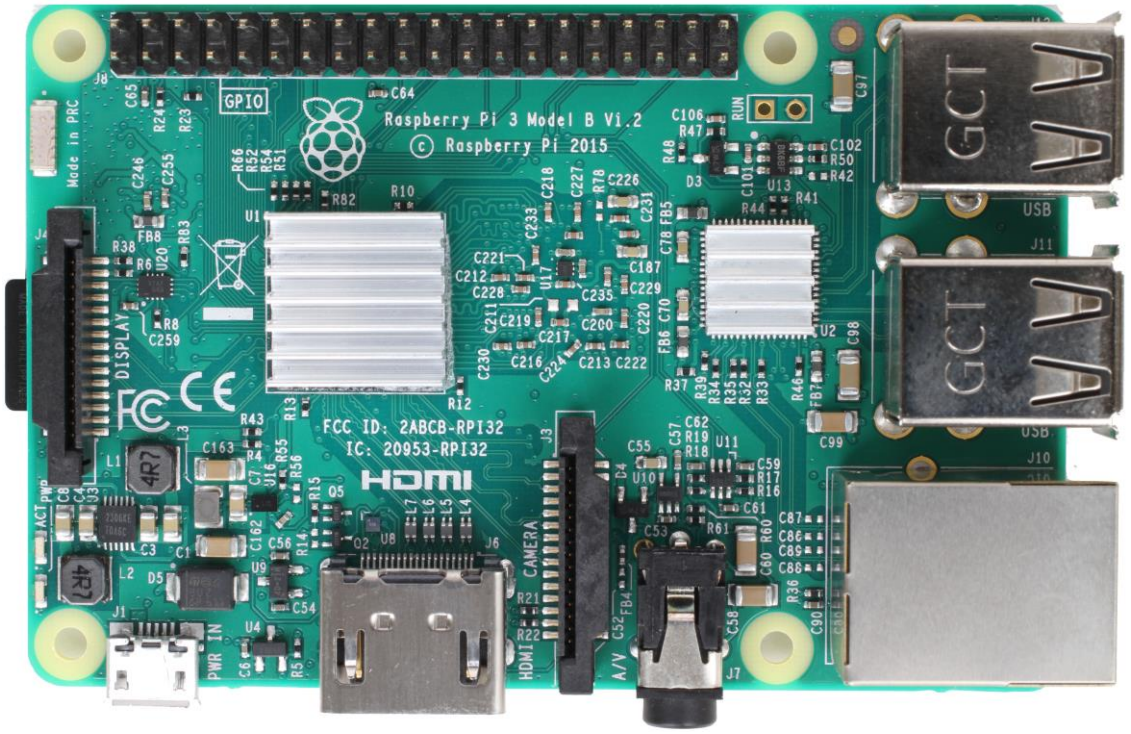
CEC CONTINUING EDUCATION CENTER

Digi-Key ELECTRONICS

Easy TCP/IP for IoT

AGENDA

- Hardware - Linux Toolbox
- Firmware - IPv6 Fundamentals
- Firmware – Raspberry Pi-Based Border Router
- Firmware – Client Application
- Day 2 Summary

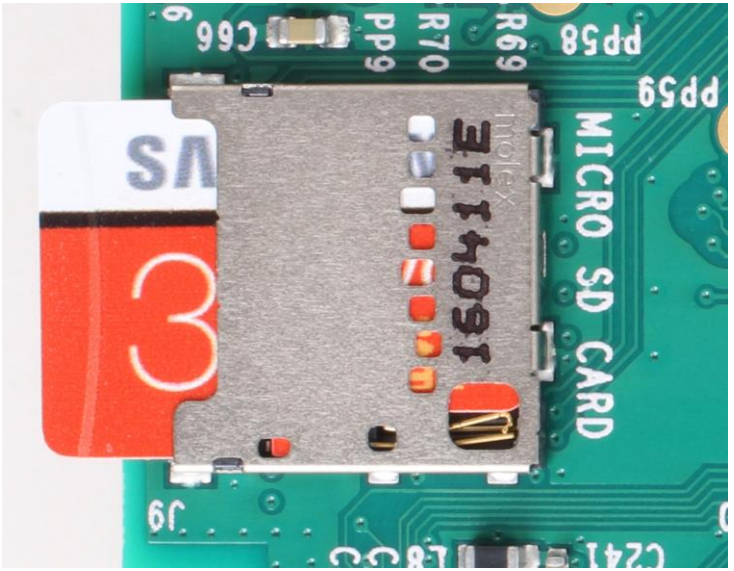


Easy TCP/IP for IoT

Hardware - Linux Toolbox

Required Equipment

- microSD Card
- Raspbian Jessie
- Raspberry Pi 3 Model B
- Etcher
- SmarTTY



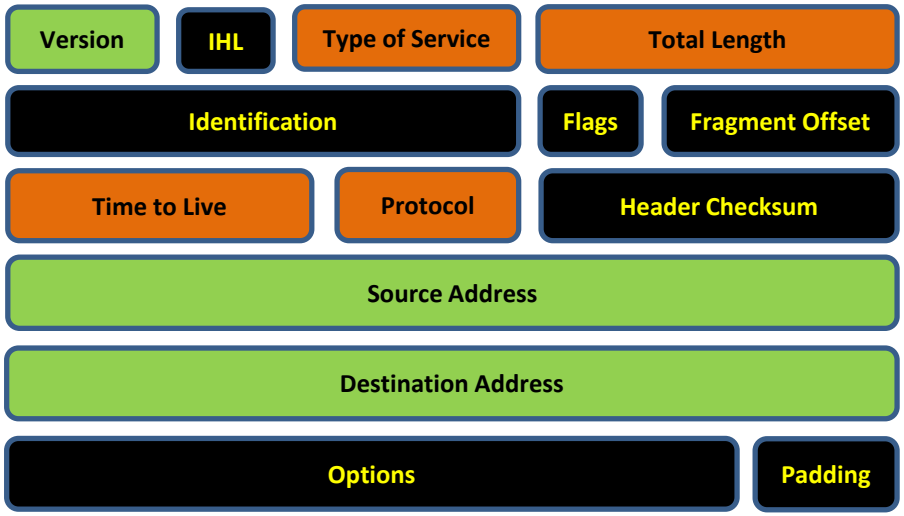
Optional Equipment

- HDMI-Equipped Color Monitor
- HDMI Cable
- USB Keyboard and Mouse

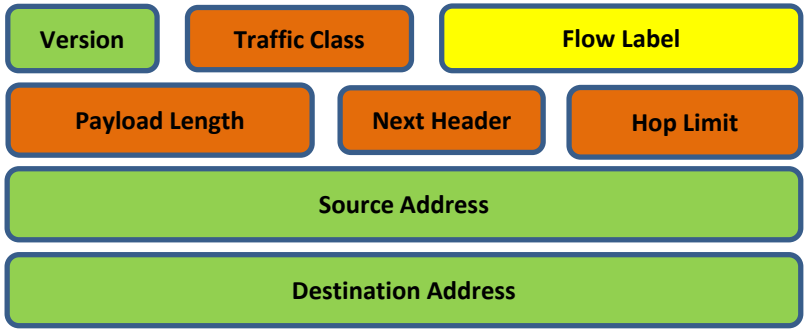
Easy TCP/IP for IoT

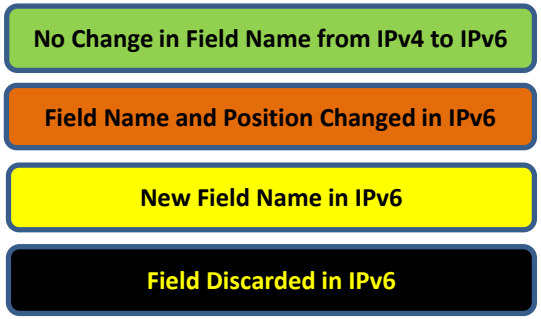
Firmware – IPv6 Fundamentals

IPv4 Header



IPv6 Header





IPv6 Notation –

2001:0DB8:0000:AAAA:0000:0000:2222:3333

- **128 bits Organized as Eight 16-bit Blocks**
- **Each Block of 4 Hexadecimal Digits Delimited by a Colon**

Shortened IPv6 Notation –

- **Eliminate Leading Zero**
  2001:DB8:0000:BBBB:0000:0000:2222:3333
- **Eliminate Leading Zeroes**
  2001:DB8:0:BBBB:0:0:2222:3333
- **Replace Consecutive Zero Fields with ::**
  2001:DB8:0:BBBB::2222:3333

Easy TCP/IP for IoT

Firmware – IPv6 Fundamentals

BLE IPv6 Link-Local Address

- At network initialization, 6LN and 6LBR devices generate and assign IPv6 link-local addresses to their BLE network interface. Link-local addresses are based on the 48-bit BLE device address.
- A 64-bit Interface Identifier (IID) is formed from the 48-bit BLE device address.
- Invert the “U” bit to signify the uniqueness of the MAC derived address.
- The newly-generated IID is appended with the prefix FE80::/64.

B8:27:EB:E2:02:F8

BA:27:EB:FF:FE:E2:02:F8

BA27:EBFF:FEE2:02F8

FE80:0000:0000:0000:BA27:EBFF:FFE2:02F8

FE80::BA27:EBFF:FFE2:2F8

Easy TCP/IP for IoT

Firmware – IPv6 Fundamentals

Address Types

IPv6 addresses are 128-bit identifiers for device interfaces.

Unicast

- **An identifier for a single interface. A packet sent to a unicast address is delivered to the interface identified by that address.**

Unicast Example - 2001:0DB8:0000:0000:0008:200C:417A

Anycast

- **An identifier for a set of interfaces that typically belong to differing nodes. A packet sent to an anycast address is delivered to the nearest one of the interfaces identified by that address. Anycast addresses are taken from unicast address spaces and are not syntactically distinguishable from unicast addresses.**

Multicast

- **An identifier for a set of interfaces that typically belong to differing nodes. A packet sent to a multicast address is delivered to all interfaces identified by that address.**

Multicast Example – FF01:0000:0000:0000:0000:0000:0000:0101

Easy TCP/IP for IoT

Firmware – IPv6 Fundamentals

Address Types (RFC3513)

Address Type	Binary Prefix	IPv6 Notation
Unspecified	00 . . . 0 (128 bits)	::/128
Loopback	00 . . . 1 (128 bits)	::1/128
Multicast	11111111	FF00::/8
Link-local unicast	1111111010	FE80::/10
Site-local unicast	1111111011	FEC0::/10
Global unicast	(everything else)	

Easy TCP/IP for IoT

Firmware – Raspberry Pi-Based Border Router – LE Scan

File Edit View SCP Tools Help

File List

Filter:

File name

..
Desktop
Documents
Downloads
Music
Pictures
Public
python_games
Templates
Videos
.cache
.config
.dbus
.gnupg
.local
.nano
.ssh
.themes
.vnc
2019-10-05-113145_1824x98
2019-10-05-113151_1824x98
2019-10-05-113421_1824x98
.bash_history
.bash_logout
.bashrc

```
root@raspberrypi:/home/pi# modprobe bluetooth_6lowpan
root@raspberrypi:/home/pi# echo 1 > /sys/kernel/debug/bluetooth/6lowpan_enable
root@raspberrypi:/home/pi# hcitool lescan
LE Scan ...
00:05:8A:E8:EB:0C TCP_Client
1A:01:3F:FB:36:B2 (unknown)
00:AC:1E:A6:54:F2 (unknown)
67:A2:27:1F:3A:8C (unknown)
67:A2:27:1F:3A:8C (unknown)
54:BD:79:19:D1:91 (unknown)
5A:7F:24:BE:21:03 (unknown)
5A:7F:24:BE:21:03 (unknown)
00:A4:40:B8:89:8A TCP_Server
5F:A5:15:63:4F:C1 (unknown)
00:05:8A:E8:EB:0C (unknown)
1C:0C:E7:F5:32:FB (unknown)
14:7D:A9:73:A7:4E (unknown)
90:DD:5D:EE:36:98 (unknown)
00:A4:40:B8:89:8A (unknown)
90:DD:5D:EE:36:98 (unknown)
67:A5:4C:4E:17:4E (unknown)
90:DD:5D:EE:34:E3 (unknown)
90:DD:5D:EE:34:E3 (unknown)
67:A5:4C:4E:17:4E (unknown)
30:8C:FB:3A:2F:16 (unknown)
5F:A5:15:63:4F:C1 (unknown)
root@raspberrypi:/home/pi#
```

pi@192.168.10.101:~

SCP: No transfers

10171B sent, 44KB received

Easy TCP/IP for IoT

Firmware – Raspberry Pi-Based Border Router – Activate 6lowPAN

File Edit View SCP Tools Help

File List

Filter:

File name	Size
..	<dir>
Desktop	<dir>
Documents	<dir>
Downloads	<dir>
Music	<dir>
Pictures	<dir>
Public	<dir>
python_games	<dir>
Templates	<dir>
Videos	<dir>
.cache	<dir>
.config	<dir>
.dbus	<dir>
.gnupg	<dir>
.local	<dir>
.ssh	<dir>
.themes	<dir>
.vnc	<dir>
.bash_history	27
.bash_logout	220
.bashrc	3523
.profile	675
.Xauthority	170
.xsession-errors	3954
.xsession-errors.old	3954

```
Welcome to Smart Terminal.
pi@192.168.10.101:~$ sudo su
root@raspberrypi:/home/pi# modprobe bluetooth_6lowpan
root@raspberrypi:/home/pi# echo 1 > /sys/kernel/debug/bluetooth/6lowpan_enable
root@raspberrypi:/home/pi# echo "connect 00:05:8a:e8:eb:0c 1" > /sys/kernel/debug/bluetooth/6lowpan_control
root@raspberrypi:/home/pi# ifconfig bt0
bt0: flags=4177<UP,POINTOPOINT,RUNNING,MULTICAST>  mtu 1280
        inet6 fe80::ba27:ebff:fee7:6a12  prefixlen 64  scopeid 0x20<link>
        unspec B8-27-EB-FF-FE-E7-6A-12-00-00-00-00-00-00-00-00  txqueuelen 1  (UNSPEC)
        RX packets 1  bytes 28 (28.0 B)
        RX errors 0  dropped 2  overruns 0  frame 0
        TX packets 8  bytes 293 (293.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@raspberrypi:/home/pi# ping6 -I bt0 fe80::0205:8aff:fee8:eb0c -c 4
PING fe80::0205:8aff:fee8:eb0c(fe80::205:8aff:fee8:eb0c) from fe80::ba27:ebff:fee7:6a12%bt0 bt0: 56 data bytes
64 bytes from fe80::205:8aff:fee8:eb0c%bt0: icmp_seq=1 ttl=255 time=123 ms
64 bytes from fe80::205:8aff:fee8:eb0c%bt0: icmp_seq=2 ttl=255 time=136 ms
64 bytes from fe80::205:8aff:fee8:eb0c%bt0: icmp_seq=3 ttl=255 time=80.9 ms
64 bytes from fe80::205:8aff:fee8:eb0c%bt0: icmp_seq=4 ttl=255 time=92.3 ms

--- fe80::0205:8aff:fee8:eb0c ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 80.929/108.324/136.254/22.495 ms
root@raspberrypi:/home/pi# _
```

pi@192.168.10.101:~

SCP: No transfers

20KB sent, 55KB received

Easy TCP/IP for IoT

Firmware – Raspberry Pi-Based Border Router – Connect Client/Server

File Edit View SCP Tools Help

File List

Filter:

File name

..
Desktop
Documents
Downloads
Music
Pictures
Public
python_games
Templates
Videos
.cache
.config
.dbus
.gnupg
.local
.nano
.ssh
.themes
.vnc
2019-10-05-113145_1824x98
2019-10-05-113151_1824x98
2019-10-05-113421_1824x98
.bash_history
.bash_logout
.bashrc

```
Welcome to Smart Terminal.
pi@192.168.10.101:~$ sudo su
root@raspberrypi:/home/pi# echo "connect 00:a4:40:b8:89:8a 1" > /sys/kernel/debug/bluetooth/6lowpan_control
root@raspberrypi:/home/pi# echo "connect 00:05:8a:e8:eb:0c 1" > /sys/kernel/debug/bluetooth/6lowpan_control
root@raspberrypi:/home/pi# ifconfig bt0 add 2001:db8::1/64
root@raspberrypi:/home/pi# sysctl -w net.ipv6.conf.all.forwarding=1
net.ipv6.conf.all.forwarding = 1
root@raspberrypi:/home/pi# ifconfig bt0
bt0: flags=4177<UP,POINTOPOINT,RUNNING,MULTICAST>  mtu 1280
        inet6 2001:db8::1  prefixlen 64  scopeid 0x0<global>
        inet6 fe80::ba27:ebff:fee7:6a12  prefixlen 64  scopeid 0x20<link>
        unspec B8-27-EB-FF-FE-E7-6A-12-00-00-00-00-00-00-00-00  txqueuelen 1  (UNSPEC)
        RX packets 8  bytes 308 (308.0 B)
        RX errors 0  dropped 4  overruns 0  frame 0
        TX packets 26  bytes 1645 (1.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@raspberrypi:/home/pi# _
```

pi@192.168.10.101:~

SCP: No transfers

8763B sent, 45KB received

Easy TCP/IP for IoT

Firmware – Raspberry Pi-Based Border Router – Ping6

File Edit View SCP Tools Help

File List

Filter:

File name

..
Desktop
Documents
Downloads
Music
Pictures
Public
python_games
Templates
Videos
.cache
.config
.dbus
.gnupg
.local
.nano
.ssh
.themes
.vnc
2019-10-05-113145_1824x98
2019-10-05-113151_1824x98
2019-10-05-113421_1824x98
.bash_history
.bash_logout
.bashrc

```
        unspec B8-27-EB-FF-FE-E7-6A-12-00-00-00-00-00-00-00-00  txqueuelen 1  (UNSPEC)
        RX packets 8  bytes 308 (308.0 B)
        RX errors 0  dropped 4  overruns 0  frame 0
        TX packets 26  bytes 1645 (1.6 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@raspberrypi:/home/pi# ping6 -I bt0 2001:db8::02a4:40ff:feb8:898a -c 4
PING 2001:db8::02a4:40ff:feb8:898a(2001:db8::2a4:40ff:feb8:898a) from 2001:db8::1 bt0: 56 data bytes
64 bytes from 2001:db8::2a4:40ff:feb8:898a: icmp_seq=1 ttl=255 time=161 ms
64 bytes from 2001:db8::2a4:40ff:feb8:898a: icmp_seq=2 ttl=255 time=240 ms
64 bytes from 2001:db8::2a4:40ff:feb8:898a: icmp_seq=3 ttl=255 time=185 ms
64 bytes from 2001:db8::2a4:40ff:feb8:898a: icmp_seq=4 ttl=255 time=197 ms

--- 2001:db8::02a4:40ff:feb8:898a ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3001ms
rtt min/avg/max/mdev = 161.888/196.344/240.127/28.354 ms
root@raspberrypi:/home/pi# ping6 -I bt0 2001:db8::0205:8aff:fee8:eb0c -c 4
PING 2001:db8::0205:8aff:fee8:eb0c(2001:db8::205:8aff:fee8:eb0c) from 2001:db8::1 bt0: 56 data bytes
64 bytes from 2001:db8::205:8aff:fee8:eb0c: icmp_seq=1 ttl=255 time=143 ms
64 bytes from 2001:db8::205:8aff:fee8:eb0c: icmp_seq=2 ttl=255 time=219 ms
64 bytes from 2001:db8::205:8aff:fee8:eb0c: icmp_seq=3 ttl=255 time=165 ms
64 bytes from 2001:db8::205:8aff:fee8:eb0c: icmp_seq=4 ttl=255 time=176 ms

--- 2001:db8::0205:8aff:fee8:eb0c ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3004ms
rtt min/avg/max/mdev = 143.336/176.194/219.510/27.714 ms
root@raspberrypi:/home/pi# _
```

pi@192.168.10.101:~

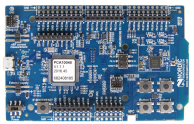
SCP: No transfers

13KB sent, 56KB received

Easy TCP/IP for IoT

Firmware – Client Application – IPv6 Server Address

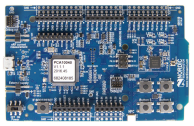
```
/** Remote TCP Port Address on which data is transmitted.
 *  Modify m_remote_addr according to your setup.
 *  The address provided below is a place holder.  */
static const ip6_addr_t                    m_remote_addr =
{
    .addr =
    {HTONL(0x20010DB8),
    0x00000000,
    HTONL(0x02A440FF),
    HTONL(0xFEB8898A)}

};
```

Easy TCP/IP for IoT

Firmware – Client Application – Request Connection

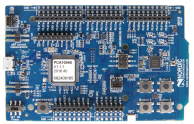
```
/**@brief Request TCP Port Connection.
 *
 * @details Request TCP Port Connection, connection procedure is complete when
 *          tcp_connected_callback called.
 */
void tcp_request_connection(void)
{
    err_t err = tcp_connect(mp_tcp_port, &m_remote_addr, TCP_SERVER_PORT, tcp_connection_callback);
    APP_ERROR_CHECK(err);
    APPL_LOG(">> TCP Connection Requested.");
}
```

Easy TCP/IP for IoT

Firmware – Client Application – Client Connected

```
/**@brief TCP Port Connection Callback.
 *
 * @details Callback registered with TCP for connection complete. err indicates if the
 * @param[in]   p_arg     Receive argument set on the port.
 * @param[in]   p_pcb     PCB identifier of the port.
 * @param[in]   err       Event result indicating error associated with the receive,
 *                        if any, else ERR_OK.
 */
static err_t tcp_connection_callback(void           * p_arg,
                                     struct tcp_pcb * p_pcb,
                                     err_t            err)
{
    APPL_LOG (">> TCP Connected, result 0x%08X.", err);

    //Ensure connection establishment was successful.
    APP_ERROR_CHECK(err);

    //Set the state of TCP port and associated handlers/information.
    m_tcp_state = TCP_STATE_CONNECTED;

    tcp_setprio(p_pcb, TCP_PRIO_MIN);
    tcp_arg(p_pcb, NULL);
    tcp_recv(p_pcb, tcp_recv_data_handler);
    tcp_err(p_pcb, tcp_error_handler);
    tcp_poll(p_pcb, tcp_connection_poll, 0);

    LEDS_ON(TCP_CONNECTED_LED);

    return ERR_OK;
}
```

Easy TCP/IP for IoT

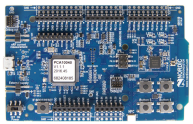
Firmware – Client Application – Send Data

```
APPL_LOG ("Available TCP length 0x%08lx", len);

if (len >= TCP_DATA_SIZE)
{
    m_sequence_number++;

    //Register callback to get notification of data reception is complete.
    tcp_sent(p_pcb, tcp_write_complete);
    uint8_t tcp_data[TCP_DATA_SIZE];

    tcp_data[0] = (uint8_t )((m_sequence_number >> 24) & 0x000000FF);
    tcp_data[1] = (uint8_t )((m_sequence_number >> 16) & 0x000000FF);
    tcp_data[2] = (uint8_t )((m_sequence_number >> 8)  & 0x000000FF);
    tcp_data[3] = (uint8_t )(m_sequence_number         & 0x000000FF);

    tcp_data[4] = 'P';
    tcp_data[5] = 'i';
    tcp_data[6] = 'n';
    tcp_data[7] = 'g';

    //Enqueue data for transmission.
    err = tcp_write(p_pcb, tcp_data, TCP_DATA_SIZE, 1);
```

Easy TCP/IP for IoT

Firmware – Client Application – APPL_LOG

```
<info> app: Application started.

<info> app: Physical layer in connectable mode.

<info> app: Physical layer: connected.

<info> app: IPv6 interface up.

<info> app: >> TCP Connection Requested.

<info> app: >> TCP Connected, result 0x00000000.

<info> app: >> TCP TX Data.

<info> app: Available TCP length 0x00000E00
```

Easy TCP/IP for IoT

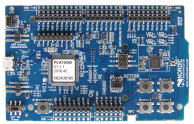
Firmware – Client Application – Server Received Data

```
/**@brief Callback registered for receiving data on the TCP Port.
 *
 * @param[in]   p_arg     Receive argument set on the TCP port.
 * @param[in]   p_pcb     TCP PCB on which data is received.
 * @param[in]   p_buffer  Buffer with received data.
 * @param[in]   err       Event result indicating error associated with the receive,
 *                        if any, else ERR_OK.
 */
err_t tcp_recv_data_handler(void           * p_arg,
                            struct tcp_pcb * p_pcb,
                            struct pbuf    * p_buffer,
                            err_t            err)
{
    APPL_LOG (">> TCP Data.");

    //Check event result before proceeding.
    if (err == ERR_OK)
    {
        uint8_t *p_data = p_buffer->payload;

        if (p_buffer->len == TCP_DATA_SIZE)
        {
            uint32_t sequence_number = 0;

            sequence_number  = ((p_data[0] << 24) & 0xFF000000);
            sequence_number |= ((p_data[1] << 16) & 0x00FF0000);
            sequence_number |= ((p_data[2] << 8)  & 0x0000FF00);
            sequence_number |= (p_data[3]         & 0x000000FF);

            LEDS_OFF(ALL_APP_LED);
```

Easy TCP/IP for IoT

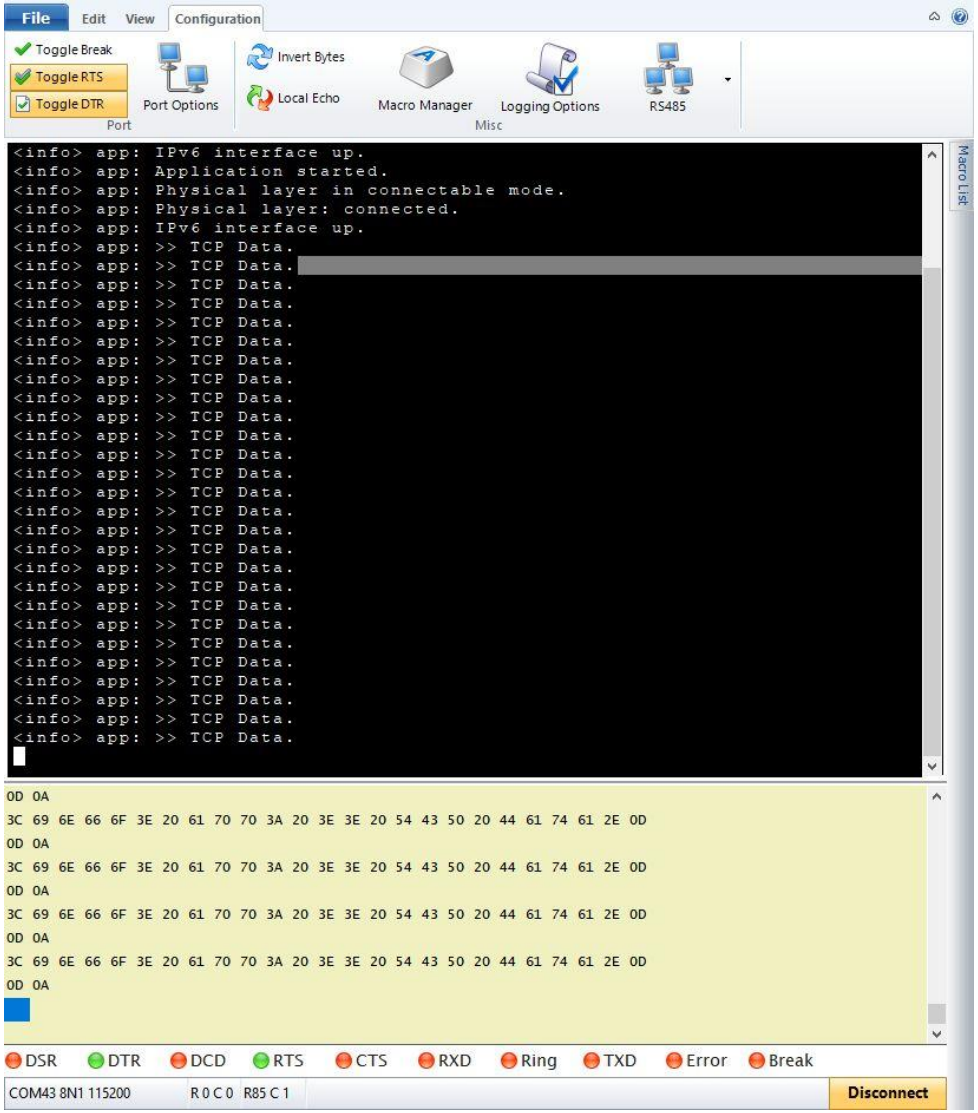
Firmware – Client Application – Server Send Data

```
/**@brief Send test data on the port.
 *
 * @details Sends TCP data in Request of size 8 in format described in description above.
 *
 * @param[in]   p_pcb     PCB identifier of the port.
 */
static void tcp_send_data(struct tcp_pcb * p_pcb, uint32_t sequence_number)
{
    err_t  err = ERR_OK;

    if (m_tcp_state != TCP_STATE_DATA_TX_IN_PROGRESS)
    {
        //Register callback to get notification of data reception is complete.
        tcp_sent(p_pcb, tcp_write_complete);
        uint8_t tcp_data[TCP_DATA_SIZE];

        tcp_data[0] = (uint8_t )((sequence_number >> 24) & 0x000000FF);
        tcp_data[1] = (uint8_t )((sequence_number >> 16) & 0x000000FF);
        tcp_data[2] = (uint8_t )((sequence_number >> 8)  & 0x000000FF);
        tcp_data[3] = (uint8_t )(sequence_number         & 0x000000FF);

        tcp_data[4] = 'P';
        tcp_data[5] = 'o';
        tcp_data[6] = 'n';
        tcp_data[7] = 'g';

        //Enqueue data for transmission.
        err = tcp_write(p_pcb, tcp_data, TCP_DATA_SIZE, 1);
```

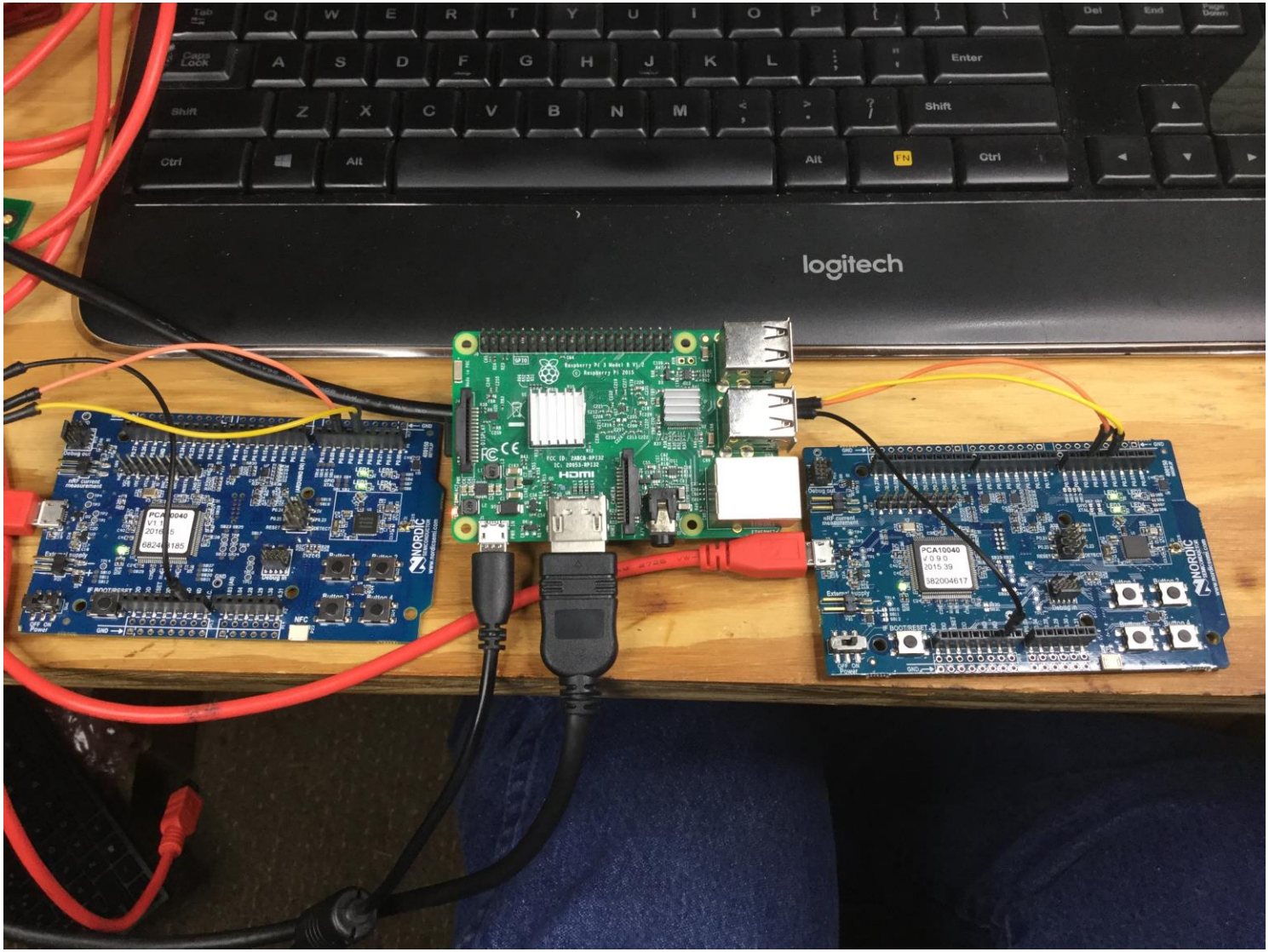
Easy TCP/IP for IoT

Firmware – Client Application – Server Response – APPL_LOG



Easy TCP/IP for IoT

Day 2 Summary



Presented by: