TI-RSLK MAX
Texas Instruments Robotics System Learning Kit
Module 20

Lecture: Internet of Things
Internet of Things

You will learn in this module

- Basic approach to the internet of things
- TCP/IP Transport Layer
- Domain Name Service
- Client-server Paradigm
The Internet of Things; challenges

- Standardization
  - SimpleLink™ implements a light-weight stack
- Interoperability
  - Technologies, vendors, companies
- Evolution
  - Incremental/continuous vs revolutionary
- Stability
- Abstraction
- Scalability
  - 50 million to 50 billion
- Security
  - Confidentiality, integrity, availability
The Internet of Things: A bird’s-eye view

IoT Data passes from physical hardware layers to software layers back and forth, connecting the real and digital worlds.

Think products
- Anything that can talk to the internet or connect to something else that can talk to the internet

Think infrastructure
- Routers, switches, cell towers, fiber optic cable, satellite transmitters, phone lines

Think services
- Google, Amazon, Facebook, etc.

Think servers
- Server farms & data centers

Hardware + Software at every stage

The Internet

Connected Things
- Phones

Computers

Edge

Cloud
The Internet of Things: A bird’s-eye view

IoT Data passes from physical hardware layers to software layers back and forth, connecting the real and digital worlds.

Operational Technology (OT)
- Think products: Anything that can talk to something else that can talk to the internet
- Sensors, Actuators, Nodes

Information Technology (IT)
- Think infrastructure: Routers, switches, cell towers, fiber optic cables, satellite transmitters, phone lines
- Cloud, Servers, Routers

Think services: Google, Amazon, Facebook, etc.

Hardware + Software at every stage

The Internet Connected Things
Computers
Phones

Edge: Hardware + Software at every stage
The Wi-Fi standard

Pros
- Ubiquitous infrastructure
- Direct connection to Internet
- Access a wide variety of APIs directly
- Only requires domain expertise in internet and firmware
- High data rate
- Security

Cons
- Poor for mobile and rural use cases
- Higher power consumption relative to some wireless standards
- Heavily reliant on network availability

Wi-Fi Primary Use Cases
- Smart Home
- Industrial/Commercial
- Fixed position connectivity
- Medical
Domain Name Service (DNS)

Internet

Network Address Translation (NAT)

Node
Node
Node
Node

192.168.1.1
192.168.1.2
192.168.1.3
192.168.1.4

public IP

Intranet

IP Addresses, IPv4
TCP is a more reliable way of Internet communication compared to UDP

- Transport Layer in the OSI model
- Use of Sockets

**OSI Model**

<table>
<thead>
<tr>
<th>Layer</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Physical</td>
<td>Bits</td>
</tr>
<tr>
<td>Data Link</td>
<td>Frames</td>
</tr>
<tr>
<td>Network</td>
<td>Packets</td>
</tr>
<tr>
<td>Transport</td>
<td>Segments</td>
</tr>
<tr>
<td>Session</td>
<td>Data</td>
</tr>
<tr>
<td>Presentation</td>
<td>Data</td>
</tr>
<tr>
<td>Application</td>
<td>Data</td>
</tr>
</tbody>
</table>

**TCP Four Way Handshake**

- User A: Fin/Ack, Ack, Fin/Ack, Ack
- Server B: Ack, Fin/Ack, Ack

**TCP Sockets**

- Client: Socket(), Connect(), Send(), Recv(), Close()
- Server: Socket(), Bind(), Listen(), Accept(), Send(), Recv(), Close()

**IP address + port number**
Layered protocol

- **Application**
- **UDP or TCP**
- **IP**
- **Physical**

**Frames:** Data link layer
- Ethernet, IEEE802.11, IEEE802.15.4

**UDP segment or TCP segment**
- UDP/TCP header
- Application Data

**IP datagram**
- IP header
- UDP/TCP header
- Application Data

**Message**
- Application
- User Data
- UDP/TCP header
- Application Data
- IP header
- Application Data
Let’s walk through the steps where two clients are accessing the same server.

**Server on openweathermap.org**

- Create a socket at Port 80
- Wait for client, accept client
- Create one thread per client to serve

---

Client-server paradigm
The server will create a connection socket, allowing requests from anywhere in the world.
Client-server paradigm

All one needs is the IP address and the port number to connect
Client-server paradigm

The server waits for clients to initiate (a server is the slave, and responds to client requests)
Let’s create a client (e.g., your RSLK robot)
Let’s create a second client (there could be 0, 1 or 100,000 clients)
Client-server paradigm

Sockets are software data structures through which communication occurs.
Client-server paradigm

Through the clientSocket the client accesses the conSocket requesting a connection
The server responds to the connection request by creating a unique thread to service that client.
The server will create a unique server socket for each client, and respond to client its existence.
The clientSocket in the client is uniquely connected to the server socket in the server.

“What’s it like in Austin?”
“What’s it like in Austin?”

The TCP packet traverses the internet from client to server; a second client starts...
The server creates a second thread and server socket to service the second client
Client-server paradigm

“What’s it like in Austin?”

The request from Client A is received (and a second request from Client B has started)

“What’s it like in Dallas?”
“What’s it like in Dallas?”

Let’s walk through the steps where two clients are accessing the same server.
Client-server paradigm

“*It is sunny and 20 C!*”

“*What’s it like in Dallas?*”

Let’s walk through the steps where two clients are accessing the same server.
Let's walk through the steps where two clients are accessing the same server.

“What’s it like in Dallas?”

“It is sunny and 20 C!”
Client-server paradigm

"It is sunny and 20 C!"

"What's it like in Dallas?"

The server is processing the request from Client B
Client-server paradigm

The response back to Client A is received

"It is sunny and 20 C!"
“Client-server paradigm”

The server is sending a response to client B, while client A is processing the response it received.

“"It is sunny and 20 C!"”

“"It is cloudy and 15 C!"”
"It is cloudy and 15 C!"

When the client is finished with the server it will close the socket
“It is cloudy and 15 C!”

The server will recognize the unique clientsocket-serversocket pairing has been broken.
Client-server paradigm

"It is cloudy and 15 C!"

The server will close the server socket associated with the broken connection to client A
Client-server paradigm

Client B follows the same steps as A

“It is cloudy and 15 C!”
Client-server paradigm

Client B processes the response it received

“It is cloudy and 15 C!”
The server is done with client A, so it kills the thread associated with the connection with client A
Client-server paradigm

When the connection from Client B is closed, the server will close the associated server socket.
The server is done with client B, so it kills the thread associated with the connection with client B.
Summary

- Internet of Things
- TCP/IP
- Sockets
- Client-server paradigm
Module 20
Lecture: SimpleLink™
You will learn in this module

- Fundamentals of synchronous serial communication
- How to interface a Wi-Fi radio to TI’s LaunchPad Development board
- Make use of software driver (set of functions to create an abstract module)
- Connect to cloud services

MSP432

SPI

Software

SimpleLink™ Wi-Fi® CC3100 wireless network processor BoosterPack™ plug-in module
SimpleLink™
MCU Platform
One environment. Unlimited potential.

Texas Instruments
**TI SimpleLink™ SDK Software Tools**

- **Configure**
  - PinMux
  - GCC
  - UniFlash

- **Build**
  - EnergyTrace

- **Debug**

---

**Tool Chain Support**

<table>
<thead>
<tr>
<th>Code Composer Studio™ IDE</th>
<th>CCS Cloud IDE</th>
<th>Energia IDE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Industry-leading tool suite for advanced optimization</td>
<td>Quickest path to developing a real application</td>
<td>Brings Arduino (Wiring) APIs to TI SimpleLink MCUs</td>
</tr>
</tbody>
</table>

- Multiple IDE support: TI CCS, CCS Cloud, Energia
- Local & Cloud-based access
- Multiple toolchain options to match your development needs

---

**Get things done with TI Cloud Tools from your web browser**

1. Immerse yourself in SimpleLink ecosystem
2. Import into CCS Cloud IDE
3. Edit, Build, Debug & Flash
Resource Explorer and SimpleLink™ Academy

Access Resource Explorer to import the latest code examples to CCS

- Accessible from inside CCS (View → Resource Explorer) or from dev.ti.com
- Materials for all TI processors searchable by part number and EVM
- When searching for MSP432 inside Resource Explorer it also contains SimpleLink Academy training with labs that can be imported into CCS that cover topics like TI-RTOS and Connectivity
- Support for TI-RTOS, FreeRTOS, and non-RTOS based code examples
Review of Synchronous Serial Communication on the MSP432

- Synchronous means send clock and data
  - Send data on one edge of clock
  - Receive data on other edge

- Serial Peripheral Interface (SPI) Protocol

![Diagram of MSP432 SPI Communication](image-url)

- MSP432
  - Write data: UCTXIFG
  - Read data: UC3RXBUF

- I/O device slave
  - STE
  - CLK
  - MOSI
  - SOMI

- Shift registers

- UCTXIFG 1 means TXBUFEempty
- UCRXIFG 1 means RXBUF has data

Lab 11
Serial Peripheral Interface (SPI) Timing

Signals

- Clock
- Data out
- Data in
- Enable

\[ n = 7 \text{ or } 8 \text{ bits} \]
CC3100 Hardware

<table>
<thead>
<tr>
<th>P1</th>
<th>P3</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCC(3.3V)</td>
<td>+5V</td>
</tr>
<tr>
<td>UN-USED</td>
<td>GND</td>
</tr>
<tr>
<td>UART1_TX</td>
<td>NC</td>
</tr>
<tr>
<td>UART1_RX</td>
<td>NC</td>
</tr>
<tr>
<td>nHIB</td>
<td>NC</td>
</tr>
<tr>
<td>UN-USED</td>
<td>NC</td>
</tr>
<tr>
<td>SPI_CLK</td>
<td>NC</td>
</tr>
<tr>
<td>UN-USED</td>
<td>NC</td>
</tr>
<tr>
<td>UN-USED</td>
<td>NC</td>
</tr>
</tbody>
</table>

Uses SPI
CC3100 Internet on a Chip
Summary

Wi-Fi provides

1. Communication from the robot can log debugging information on the cloud
2. Communication to the robot for remote control or to receive external data
3. Robot can autonomously query information from the web that may be relevant to its operation
IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265
Copyright © 2019, Texas Instruments Incorporated