

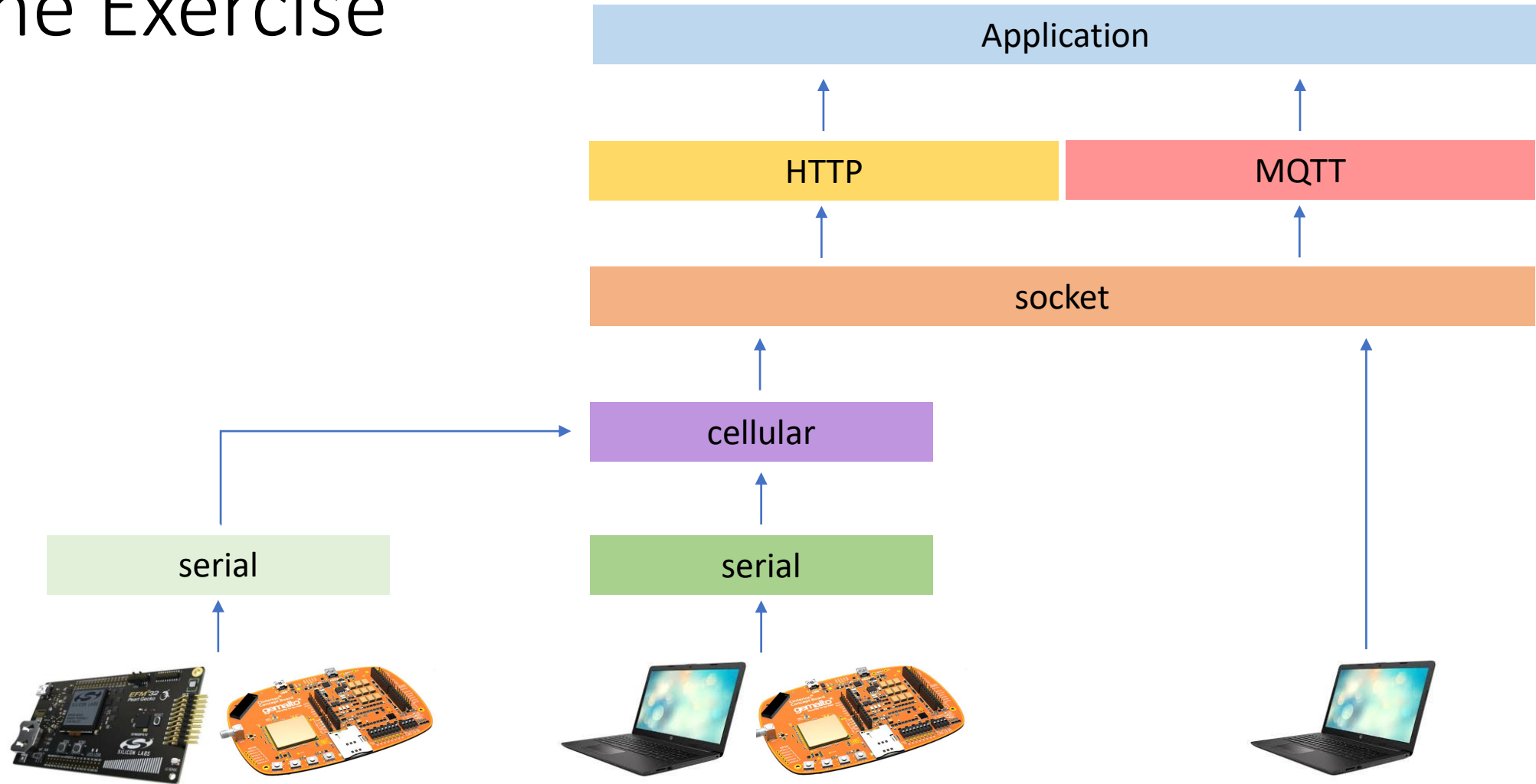
WORKSHOP ON INTERNET OF THINGS 67612

Exercise 6

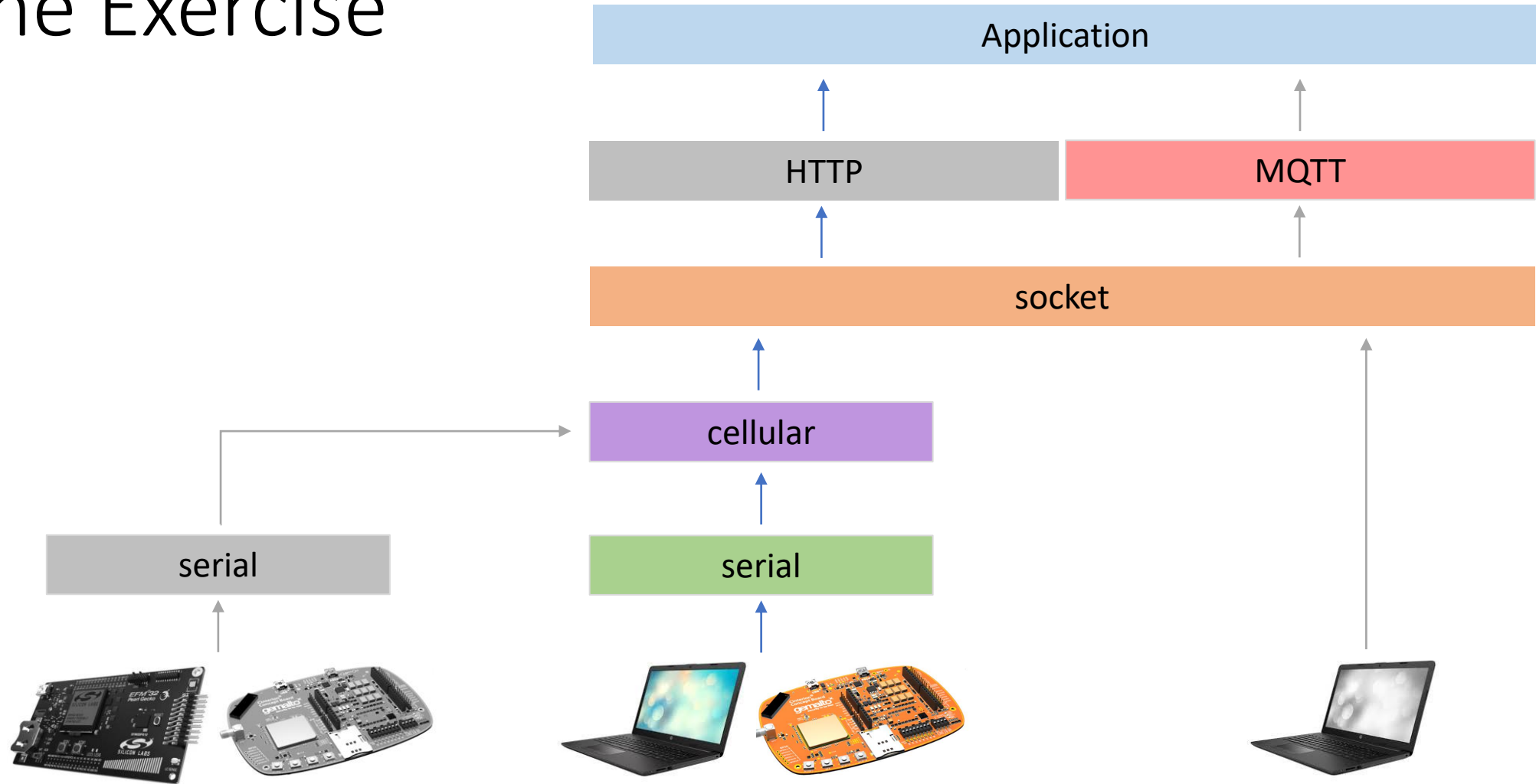
MQTT OVER BSD SOCKET

Prof. David Hay, Dr. Yair Poleg, Mr. Samyon Ristov

The Exercise

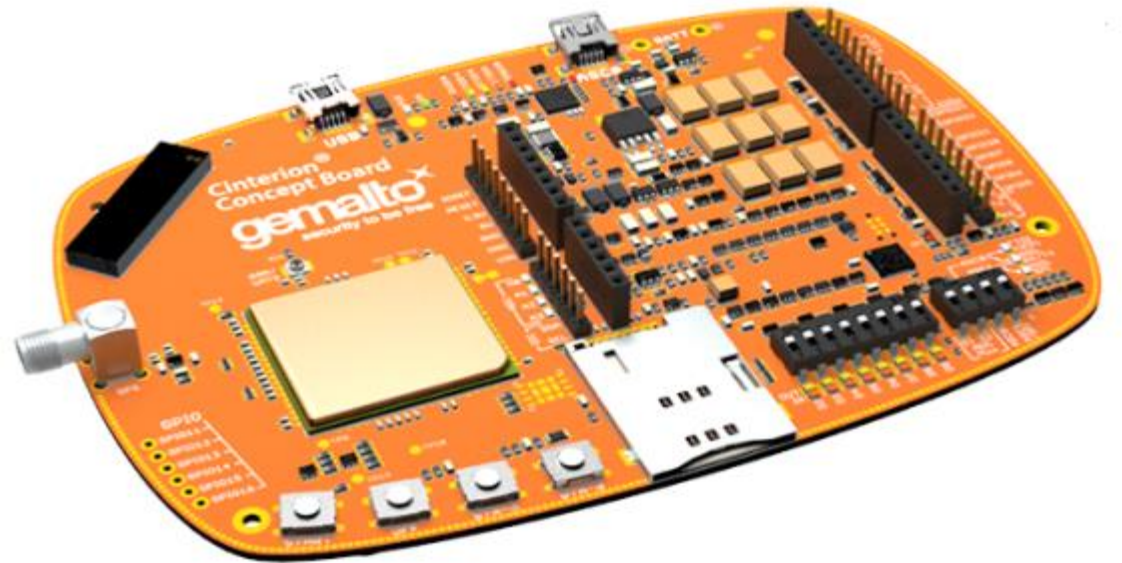


The Exercise



The Exercise

- Initiate an internet connection
- Connect to MQTT broker & publish LWT
- Publish MQTT messages
- Subscribe and receive MQTT message
- Print the progress of the communication



Guidance

- Use wolfMQTT library: <https://github.com/wolfSSL/wolfMQTT>
- Use the library without MQTT5, TLS, MQTT-SN, and preferably without the non-blocking option as well. In general, avoid any unnecessary library optional “defines” (unless you to work more)
- To use the library, download it to your computer (VM), copy the necessary files to your project, and add them to the compilation (makefile or cmake)

Guidance cont'd

- Create MQTTClient.c file with the following callback functions (check examples/mqttntnet.c and examples/mqttntnet.h for ideas):

Guidance cont'd

```
/*  
 * Connects the socket to the broker,  
 * to the given host and port.  
 * Returns 0 on success, and a negative number otherwise  
 * (one of MqttPacketResponseCodes)  
 * timeout_ms defines the timeout in milliseconds.  
 */  
static int NetConnect(void *context, const char* host, word16 port,  
int timeout_ms)
```

Guidance cont'd

```
/*  
 * Performs a network (socket) read from the connected broker,  
 * to the given buffer buf, and reads buf_len bytes.  
 * Returns number of read bytes on success, and a negative number  
 * otherwise (one of MqttPacketResponseCodes).  
 * timeout_ms defines the timeout in milliseconds.  
 */  
static int NetRead(void *context, byte* buf, int buf_len, int  
timeout_ms)
```


Guidance cont'd

```
/*  
 * Performs a network (socket) write to the connected broker,  
 * from the given buffer buf, with size of buf_len.  
 * Returns the number of sent bytes on success,  
 * and a negative number otherwise (one of MqttPacketResponseCodes)  
 * timeout_ms defines the timeout in milliseconds  
 */  
static int NetWrite(void *context, const byte* buf, int buf_len,  
int timeout_ms)
```

Guidance cont'd

```
/*  
 * Closes the network (socket) connection to the connected broker.  
 * Returns 0, and a negative number otherwise  
 * (one of MqttPacketResponseCodes)  
 */  
static int NetDisconnect(void *context)
```

Guidance cont'd

- Add MQTTClient.h file with the following functions and implement them in MQTTClient.c:

Guidance cont'd

```
/*  
 * Initializes the Net interface for communication  
 */  
int MqttClientNet_Init(MqttNet* net, MQTTCtx* mqttCtx)
```

Guidance cont'd

```
/*  
 * De-Initializes all that was allocated by MqttClientNet  
 */  
int MqttClientNet_DeInit(MqttNet* net)
```

Guidance cont'd

- Usage of `void *context` is optional, although recommended
- `MqttPacketResponseCodes` defined in `wolfMQTT/wolfmqtt/mqtt_types.h`
- Usage of `MQTTCtx` (defined in `wolfMQTT/examples/mqttxexample.h`) is optional and can be replaced with any other struct or set of variables
- If you use `MQTTCtx`, it can be helpful to copy it to `MQTTClient.h`
- Copying `MQTTCtx` to `MQTTClient.h` may require copying `MQTTCtxState` as well
- Initializing the Net interface basically means assigning `MqttNet` struct with the previously defined callbacks (`NetConnect`, `NetRead`, `NetWrite`, `NetDisconnect`)

Guidance cont'd

- Implement the previously defined files and functions
- Use the previously built serial, modem and socket interfaces (you can fix them, modify, re-implement, and share)
- Write main.c function that:
 - Initiates an internet connection and connects to a broker (also, uses LWT)
 - Subscribes to a topic
 - Publishes two messages to the broker
 - Receives a message
 - Publishes another message
 - Disconnects from the broker and exits the program
 - Prints the results of every step
 - If error occurs, disconnect and clean, if possible, and exit
- Use [wolfMQTT/examples/mqttclient/mqttclient.c](#) for implementation ideas

Guidance cont'd

- Message #1, published by the MQTT client to the broker
- Payload of message #1:

```
{  
    "Student1ID": "<student-1-id>",  
    "Student2ID": "<student-2-id>",  
    "Student1Name": "<student-1-name>",  
    "Student2Name": "<student-2-name>",  
    "Identifier": "<IMEI>"  
}
```

Topic of message #1:

huji_iot_class/2021_2022

- Retrieve the IMEI from the mode, don't use it hard-coded, and use it in the payload of message #1 and in the topic of messages #2, #3, LWT and the subscribed topic

Guidance cont'd

- Message #2, published by the MQTT client to the broker. The message contains data about currently available operators (COPS=?)
- Payload of message #2:

```
{  
  "AvailableOperators": [  
    {  
      "OperatorName": "<op name>",  
      "OperatorCode": "<op code>",  
      "AccessTechnology": "2G/3G"  
    },  
    {<op2>}, {<op3>}, ...  
  ]  
}
```

- Topic of message #2:

huji_iot_class/2021_2022/<IMEI>

Guidance cont'd

- Subscribe to:

`huji_iot_class/2021_2022/<IMEI>/recv/#`

- Receive any message published by the other client. **Don't proceed until a message is received** (or disconnected), retry receiving if timed out. Stop only if disconnected from the broker, and/or exited transparent mode, or message received.

Guidance cont'd

- Message #3, published by the MQTT client to the broker
- Payload of message #3:

```
{  
    "DisconnectedGracefully":true  
}
```

- Topic of message #3:

```
huji_iot_class/2021_2022/<IMEI>/disconnect
```

Guidance cont'd

- LWT message payload:

```
{  
    "DisconnectedGracefully":false  
}
```

- LWT message Topic:

- `huji_iot_class/2021_2022/<IMEI>/disconnect`

Guidance cont'd

- Work (connect, send, receive) with the following broker:
 - Host: broker.mqttdashboard.com
 - Port: 1883 (port 8000 is used only for web-sockets)
 - <http://www.hivemq.com/demos/websocket-client/>
- Additional tools to monitor and debug your MQTT session:
 - **mosquitto** - <https://mosquitto.org/>
 - Tools that are described here: <https://ubidots.com/blog/top-3-online-tools-to-simulate-an-mqtt-client/>)
 - MQTTLens
 - MQTT.fx
 - MQTT-Spy

Guidance cont'd

- Connect:
 - No username and password
 - Keep alive = 60
 - Clean session = 0
 - Generate any client-id
 - Use LWT
- LWT:
 - QoS = 1
 - No retain message
- Publish:
 - QoS = 1
 - No retain message
- Subscribe:
 - QoS = 1

Exercise #6

- Work & submit in pairs
- Make sure that your submissions works on the VM (try importing it as a new project)
- Deliverables:
 - Provide all the project files, and/or export the project
 - Create makefile or CMakeLists.txt (in CLion).
 - A README file with your names, email addresses, IDs and adequate level of documentation of the deliverables and software design-architecture-flow description
 - If anything special is needed (compilation instructions and environment requirements), add it to the README
- Pack all the deliverables as .zip or .tar and upload to Moodle
- Deadline: 7.12.21, 23:59
- The grade will be based on code's functionality, description, and clear implementation

Contact

- Moodle's 'Workshop Discussions' forum is the best place for questions.
- But if needed, contact us personally:
- David Hay – dhay@cs.huji.ac.il
- Yair Poleg – yair.poleg@mail.huji.ac.il
- Samyon Ristov – samyon.ristov@mail.huji.ac.il