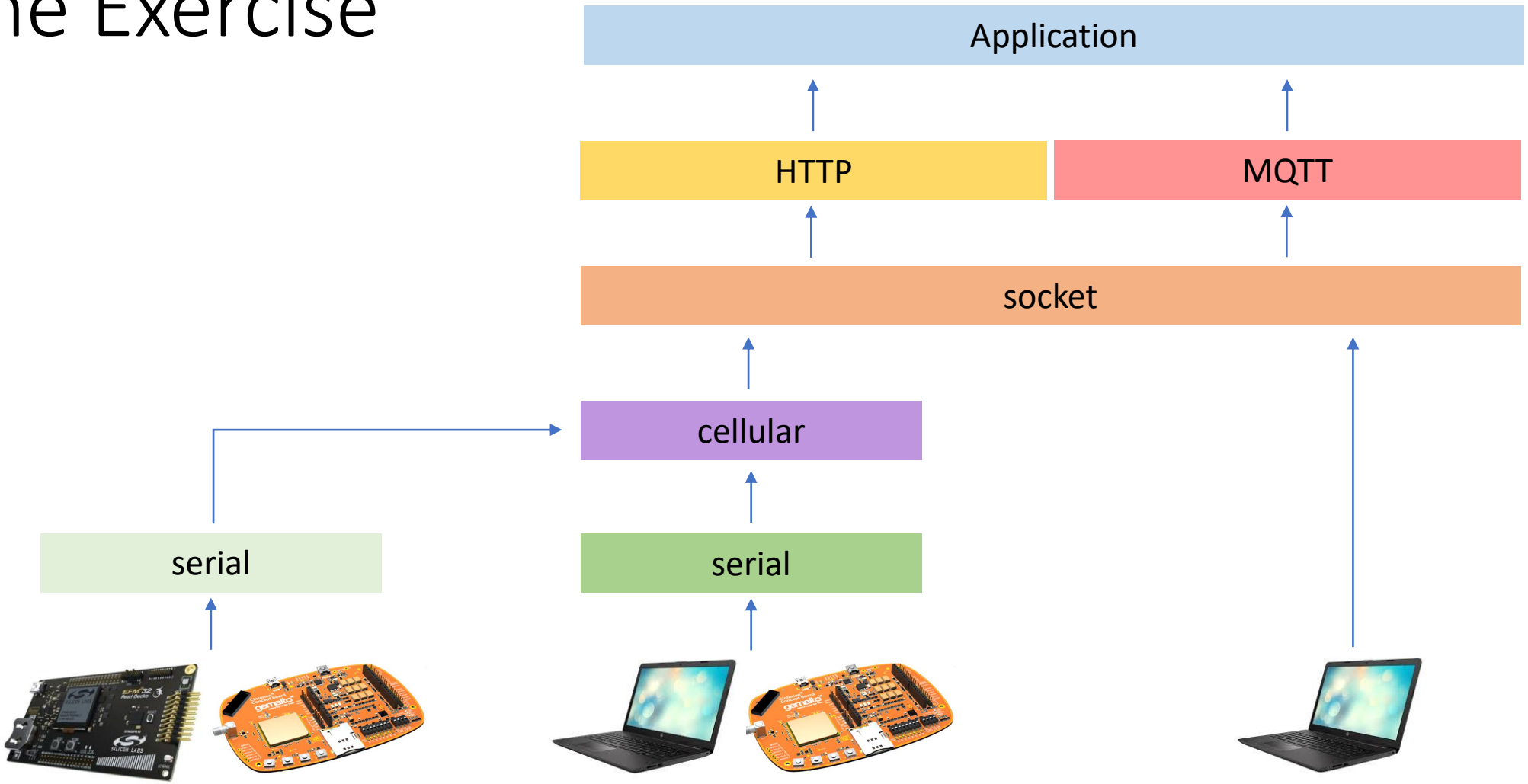
WORKSHOP ON
INTERNET OF THINGS
67612

Exercise 3

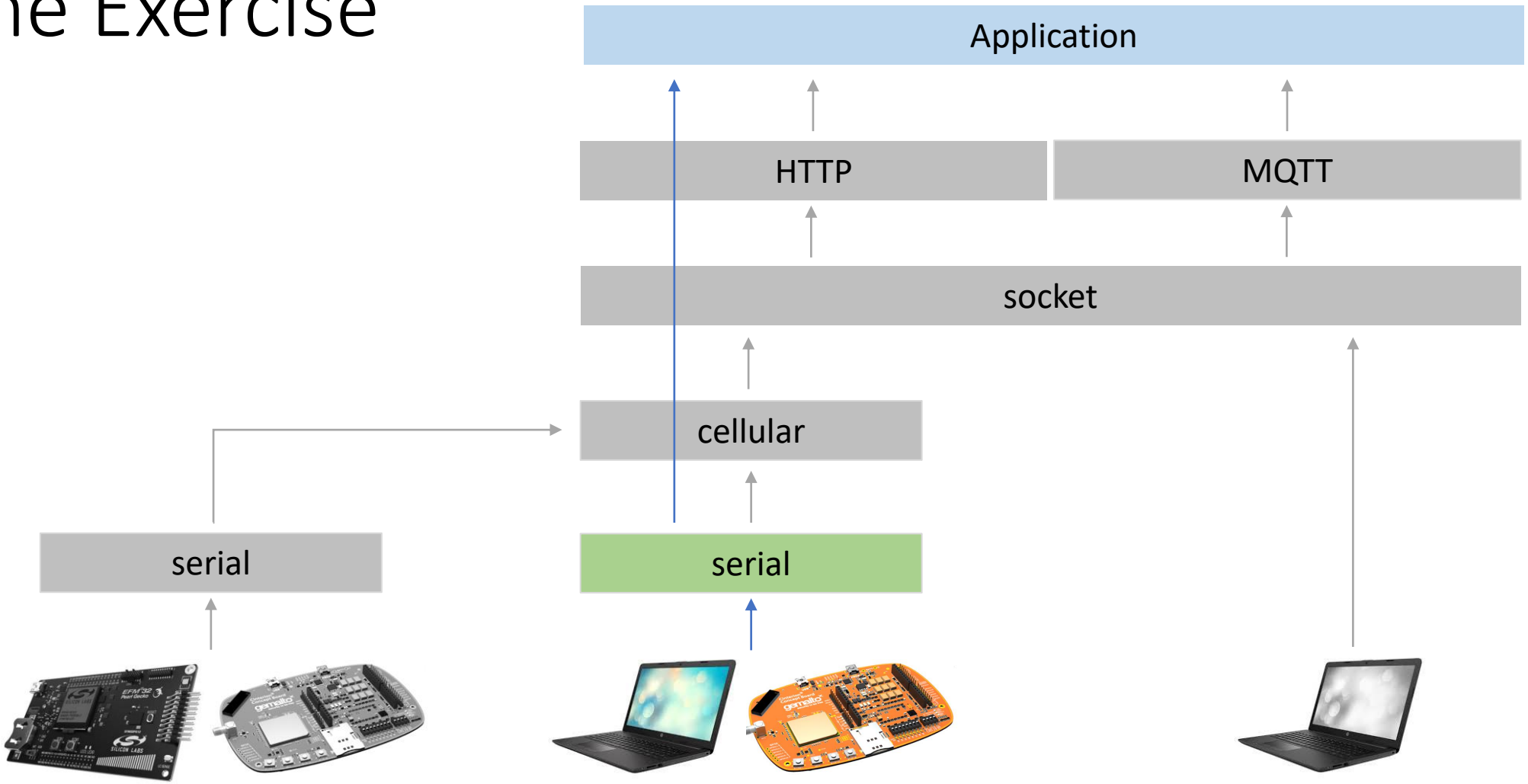
COMMUNICATING WITH CELLULAR MODEM

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

The Exercise

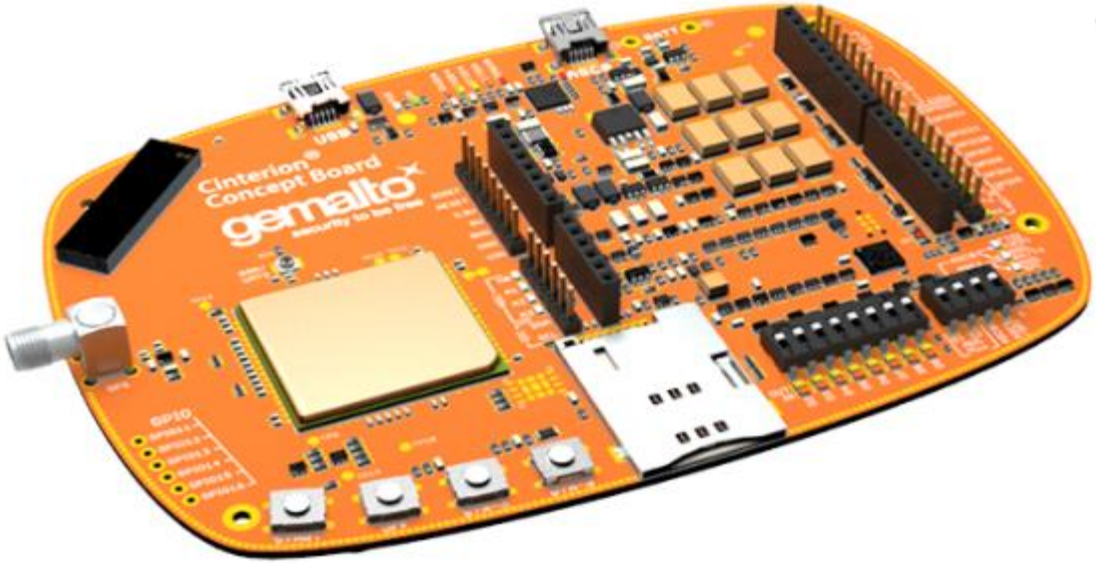


The Exercise



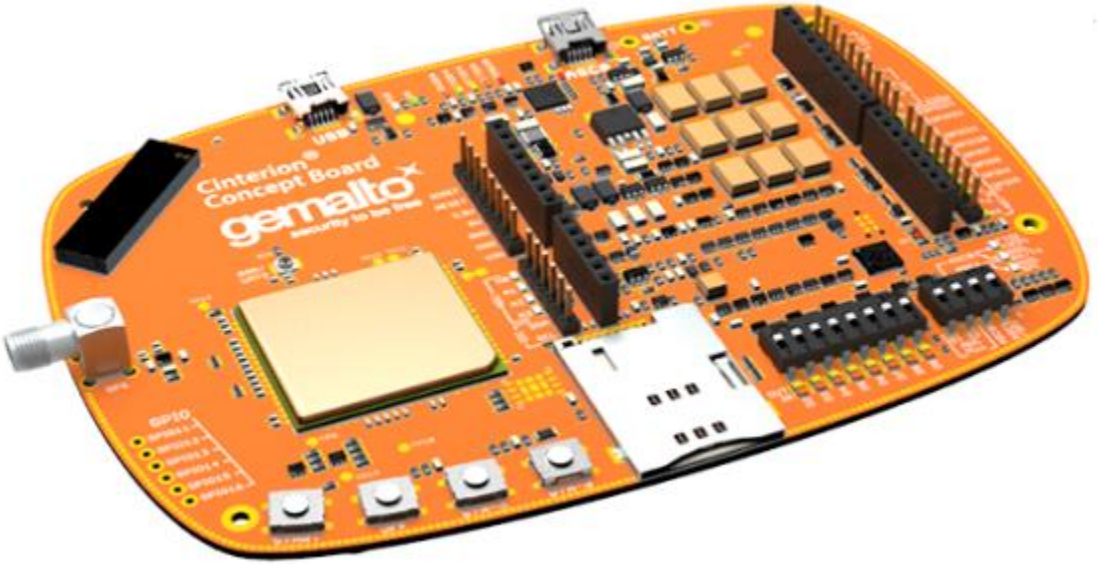
The Exercise

- Connect the Cellular Concept Board to the PC
- Validate communication with the board
- Detect ICCID (SIM ID)
- Detect IMEI (HW ID)
- Detect cellular networks
- Print the result of every step



Cellular Concept Board

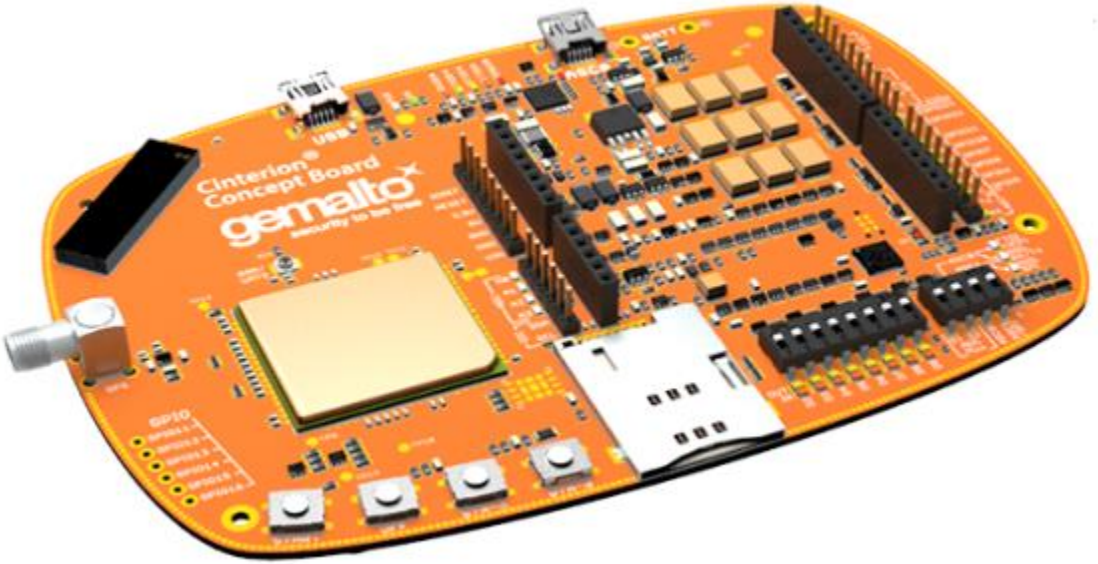
- Connect the Cellular Concept Board to the PC and make sure it works:
  - There are two USB ports on the board, use port “**ASC0**”
  - There are two USB connectors on the cable, use only one of them (e.g.: the black one)
  - Default baud rate: **115200 (8N1)** – 8 data bits, no parity, 1 stop bit
- Power up the modem by pressing the ‘Start’ button
- Wait for the “**^SYSSTART**” and **“+PBREADY”** URCs* to appear



*URC - **unsolicited result code**. Not a reply to AT command but information that can be thrown by the module at any time

Cellular Concept Board

- Write **AT,** hit enter, and see that you get **OK** as a response
- Note: there is no “backspace”, you can’t delete – everything you type, goes directly to the modem
- Can’t see what you are typing, but you do get an **OK** response? It’s OK! (Change echo mode by entering ATE0 or ATE1)
- No response at all? Start over! (the “**Off**” button can help)
- Check the Modem Concept Board Startup Guide (In Moodle)



Guidance

- **Useful Commands (case insensitive):**
- AT
- ATE0 | ATE1
- AT+CCID
- AT+GSN
- AT+CREG?
- AT+COPS?
- AT+COPS=? (takes 1-2 minutes to execute!)
- *Check Cinterion® EHS6 AT Command Set in Moodle*

```
^SYSLOADING

^SYSSTART

+PBREADY
at
OK
at+creg?
+CREG: 0,4

OK
at+cops=?
+COPS: (1,"Cellcom IL","Cellcom","42502",2),(1,"Orange IL","OrangeIL","42501",2)
,(1,"IL Pelephone","PCL","42503",2),(3,"","","42507",2),(3,"","","42508",2),(1,"
Orange IL","OrangeIL","42501",0),(1,"Cellcom IL","Cellcom","42502",0)

OK
at+cops?
+COPS: 0,0,"Cellcom IL",2

OK
```

Guidance cont’d

- Create a serial_io.h file with the following functions:
  - `int SerialInit(char* port, unsigned int baud);`
  - `int SerialRecv(unsigned char *buf, unsigned int max_len, unsigned int timeout_ms);`
  - `int SerialSend(unsigned char *buf, unsigned int size);`
  - `void SerialFlushInputBuff(void);`
  - `int SerialDisable(void);`
- (serial_io.h is available in Moodle)
- Implement these functions in a file called serial_io_linux.c
- OK to assume a single thread, also OK to use global/static variables
- Test your code with the Cellular Concept Board
  - E.g.: send “AT\r\n” and receive a response from the modem

Guidance cont’d

Create a program (main.c - no need to use args) that:

- Uses the implemented serial_io.h interface
- Initializes the Cellular Concept Board – The modem will be connected before starting the program, but turned on only after the program had started
  - You may want to wait for +PBREADY
  - You may want to implement a compile-time debug mode, in which the modem is turned on before the program is started, and then there is no need to wait for +PBREADY or anything like that
- Verifies communication by sending AT (and receiving OK) and prints the response of the following commands:
  - AT+CCID
  - AT+GSN
  - AT+CREG?
  - AT+COPS?
  - AT+COPS=?

Guidance cont’d

- Note: cellular modem doesn’t work well indoors - best by a window
- Tip: First, try to do it yourself via a serial port (putty, screen, tio, etc.)
- Tip 2: If anything does not work, copy to some text editor the commands that you planned to send, and run them manually (works best when copied from text editors, new-line included)
- Tip 3: Timing is important. Some commands need time to complete. Some commands give different response after a while. Try running **AT+CREG?** for 1-3 minutes right after the modem is turned on and see how it changes. Same with **AT+COPS?**

Useful links

- Serial Programming HOWTO
- Linux Serial Ports Using C/C++
- Serial Programming Guide for POSIX Operating Systems

Exercise #3

- Work & submit in pairs
- 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: 9.11.2020, 23:59
- Your SIM cards are limited to 5**MB**. Any additional **byte** above 5MB equals -1 point in the final score.
- 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@ayyeka.com
- Samyon Ristov – samyon.ristov@mail.huji.ac.il