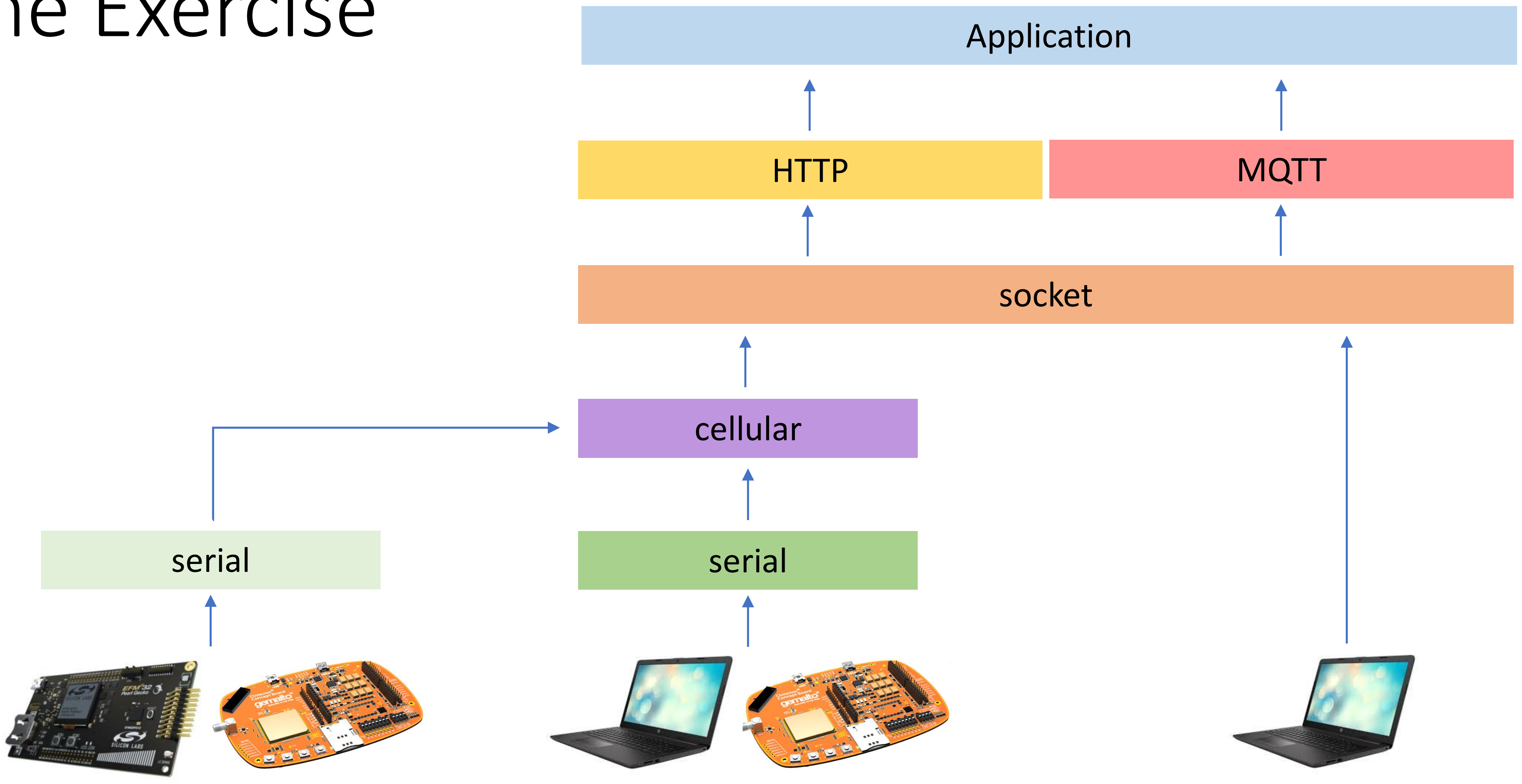
WORKSHOP ON **INTERNET OF THINGS** 67612

Exercise 4

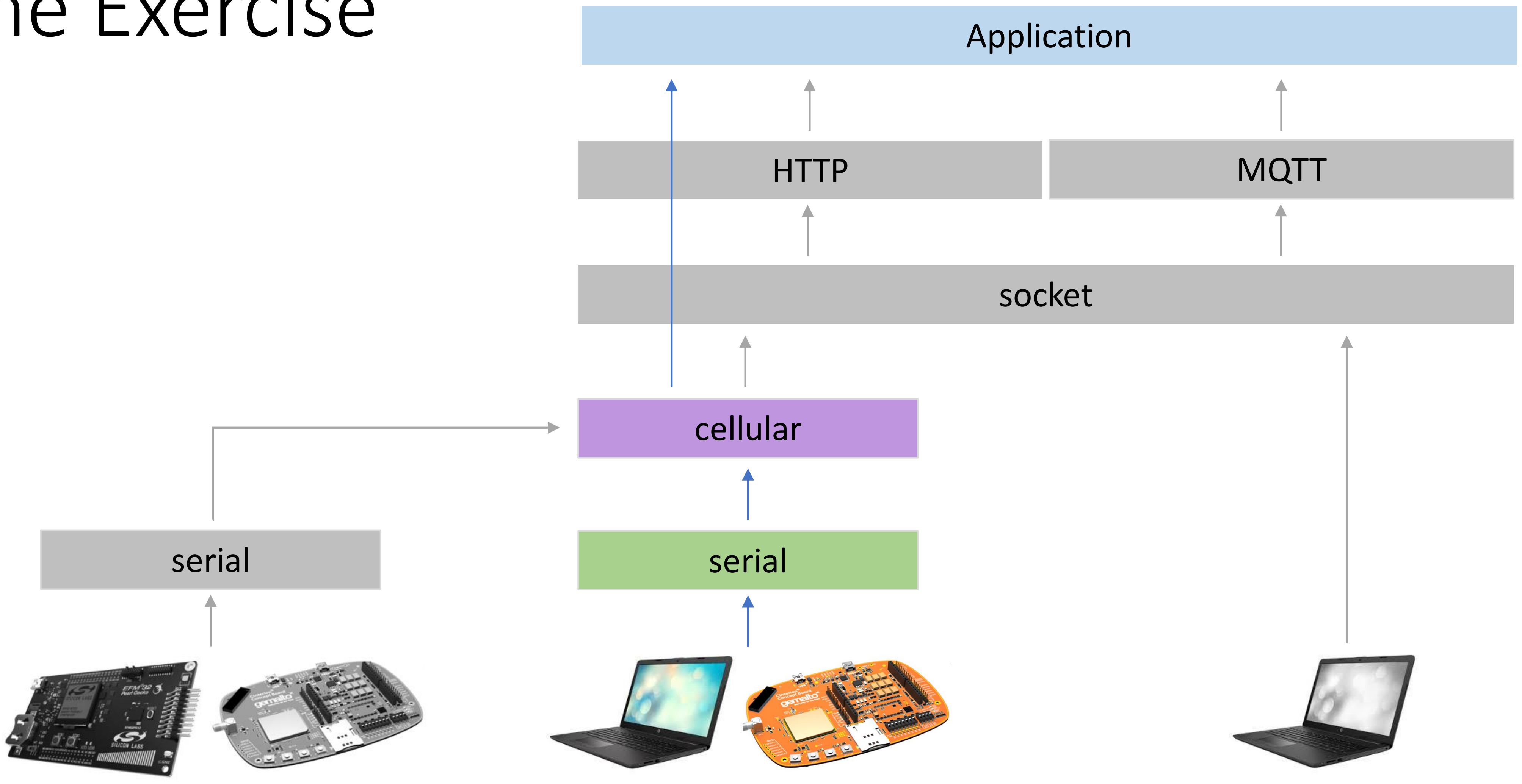
REGISTERING TO A CELLULAR NETWORK

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

The Exercise

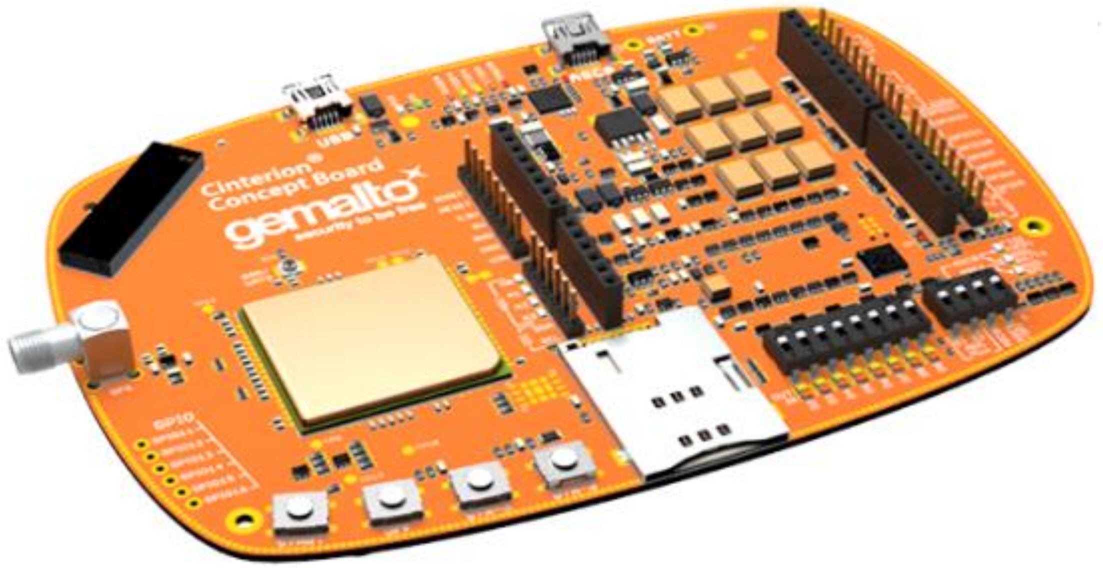


The Exercise



The Exercise

- Validate communication with the board
- Detect ICCID (SIM ID)
- Detect IMEI (HW ID)
- Detect cellular networks
- Register to a cellular network
- Detect signal strength/quality
- Print the result of every step



Guidance

- Declare the following function in “cellular.h”:

Guidance

```
/*
 * Initialize whatever is needed to start working with
 * the cellular modem (e.g. the serial port).
 * Returns 0 on success, and -1 on failure
 */
int CellularInit(char *port)

/*
 * Deallocate / close whatever resources CellularInit() allocated
 */
void CellularDisable(void);
```

Guidance

```
/*
 * Checks if the modem is responding to AT commands
 * Return 0 if it does, returns -1 otherwise
 */
int CellularCheckModem(void); ← OPTIONAL

/*
 * Checks if the modem is responding to AT commands
 * Return 0 if it does, returns -1 otherwise
 * Tries multiple times until succeeds or fails
 */
int CellularWaitUntilModemResponds(void);
```

Guidance

```
/*
 * Returns -1 if the modem did not respond or respond with an error.
 * Returns 0 if the command was successful and the
 * registration status was obtained from
 * the modem. In that case, the status parameter will
 * be populated with the numeric value
 * of the <regStatus> field of the "+CREG" AT command
 */
int CellularGetRegistrationStatus(int *status); ← OPTIONAL
```

Guidance

```
/*
 * Returns -1 if the modem did not respond or
 * respond with an error or couldn't register.
 * Returns 0 if the command was successful and the
 * registration status was obtained from
 * the modem and the modem is registered (1 or 5).
*/
int CellularWaitUntilRegistered(void);
```

Guidance

```
/*
 * Forces the modem to search for available operators
 * (see “+COPS=?” command). Returns -1
 * if an error occurred or no operators found. Returns 0 and populates
 * opList and opsFound if * the command succeeded.
 * opList is a pointer to the first item of an array of type
 * OPERATOR_INFO, which is allocated
 * by the caller of this function. The array contains a total of
* maxops items. numOpsFound is
 * allocated by the caller and set by the function.
 * numOpsFound will contain the number of
 * operators found and populated into the opList
 */
int CellularGetOperators(OPERATOR_INFO *opList, int maxops, int
*numOpsFound);
```

Guidance

```
typedef enum __OP_STATUS {
    UNKNOWN_OPERATOR = 0,
    OPERATOR_AVAILABLE = 1,
    CURRENT_OPERATOR = 2,
    OPERATOR_FORBIDDEN = 3
} OP_STATUS;

typedef struct __OPERATOR_INFO {
    char operator_name[MAX_OPERATOR_NAME]; // Long name. See <format> under
+COPS
    int operator_code; // Short name. See <format> under +COPS
    char access_technology[MAX_TECH_SIZE]; // "2G" or "3G"
    OP_STATUS operator_status;
} OPERATOR_INFO;
```

Guidance

```
/*
 * Forces the modem to register/deregister with a network.
 * Returns 0 if the
 * command was successful, returns -1 otherwise.
 * If mode=0, sets the modem to automatically register
 * with an operator (ignores the * operatorCode parameter).
 * If mode=1, forces the modem to work with a specific operator,
 * given in operatorCode.
 * If mode=2, deregisters from the network
 * (ignores the operatorCode parameter).
 * See the "+COPS=<mode>,…" command for more details
 */
int CellularSetOperator(int mode, int operatorCode);
```

Guidance

```
/*
 * Returns -1 if the modem did not respond or respond with
 * an error (note, CSQ=99 is also an error!)
 * Returns 0 if the command was successful and the signal
 * quality was obtained from
 * the modem. In that case, the csq parameter will be
 * populated with the numeric
 * value between -113dBm and -51dBm
 */
int CellularGetSignalQuality(int *csq);
```

Converting CSQ to dBm

CSQ Command Description

Guidance

```c
/*
 * Returns -1 if the modem did not respond or respond with an error.
 * Returns 0 if the command was successful and the ICCID
 * was obtained from the modem.
 * iccid is a pointer to a char buffer, which is allocated
 * by the caller of this function.
 * The buffer size is maxlen chars.
 * The obtained ICCID will be placed into the iccid buffer
 * as a null-terminated string
 */
int CellularGetICCID(char* iccid, int maxlen);
```

Guidance

```
/*
 * Returns -1 if the modem did not respond or respond with an error.
 * Returns 0 if the command was successful and the IMEI
 * was obtained from the modem.
 * imei is a pointer to a char buffer, which is allocated
 * by the caller of this function.
 * The buffer size is maxlen chars.
 * The obtained IMEI will be placed into the imei buffer
 * as a null-terminated string
 */
int CellularGetIMEI(char* imei, int maxlen);
```

Guidance

BONUS 10 POINTS

```
/*
 * Runs SMONI command. Returns -1 if the modem did not respond, respond with an error,
 * or respond with SEARCH or NOCONN.
 * Returns 0 if the command was successful and the signal info was
 * obtained from the modem.
 * sigInfo is a pointer to a struct, which is allocated by the
 * caller of this function.
 * The obtained info will be placed into the sigInfo struct
 */
int CellularGetSignalInfo(SIGNAL_INFO *sigInfo);

typedef struct __SIGNAL_INFO {
    int signal_power; // In 2G: dBm. In 3G: RSCP. See ^SMONI responses
    int EC_n0; // In 3G only. See ^SMONI responses
    char access_technology[MAX_TECH_SIZE]; // "2G" or "3G"
} SIGNAL_INFO;
```

Guidance cont’d

- Implement “cellular.h” functions in “cellular.c” file. These files are the cellular wrapper of the serial_io.h and serial_io_linux.c. Highly suggested not to use regex.
- Write main.c function that:
  - Connects to the modem by using the cellular.h interface (The modem will be connected before starting the program, but turned on only after the program had started, OR, turned on up to 20 seconds after starting the program)
  - Verifies the connection by sending an AT command
  - Turn echo on/off (whichever you need, but do it explicitly, don’t except it to be pre-set)
  - Extracts the ICCID & IMEI
  - Wait until registered to the network (CREG 0,1 or 0,5)
  - Checks for all available networks
  - Tries to connect to the first network in the list (check registration status)
    - If fails, disconnects, and tries to register to the next network in the list
    - If fails registering all networks, exits the program
  - Assuming registered to a network, gets its signal quality
  - Exits the program
  - Prints the results of every step

Check the following commands

- AT
- ATE
- AT+GSN
- AT+CCID
- AT+CREG
- AT+COPS
- AT+CSQ
- AT^SMONI

Guidance cont’d

- Your cellular.c file shouldn’t be OS/HW dependent code (or almost not OS dependent. All OS/HW dependent stuff are done in serial_io_linux.c, and just a few, in main.c)
- OK to assume a single thread, also OK to use global/static variables.
- Make sure that you can exit the program during its execution (ctrl+c)
- We suggest to use ATE0 to disable the echo. Without that, a response you read from the modem will begin with the initial command sent to the modem.
  - e.g. "AT\r\n" -> "AT\r\nOK\r\n“
- In certain environments, a single ‘\n’ char is automatically translated to “\r\n”.
- There may not be any available networks around you – that’s a totally possible scenario!
- The signal may become stronger or weaker during the connection process
- Some networks may not allow you to register

Exercise #4

- 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: 16.11.2020, 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@mail.huji.ac.il
- Samyon Ristov – samyon.ristov@mail.huji.ac.il