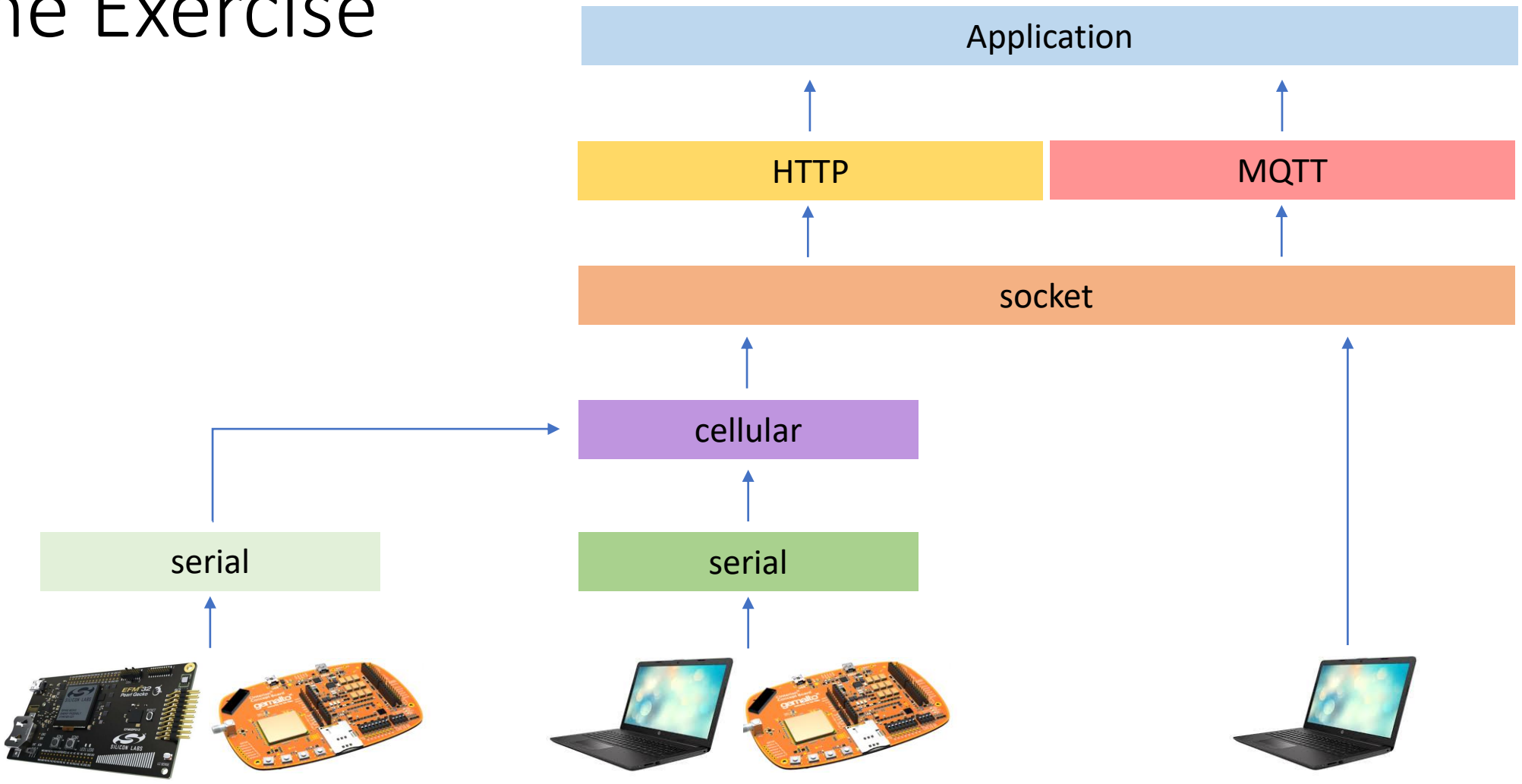
WORKSHOP ON **INTERNET OF THINGS** 67612

Exercise 8

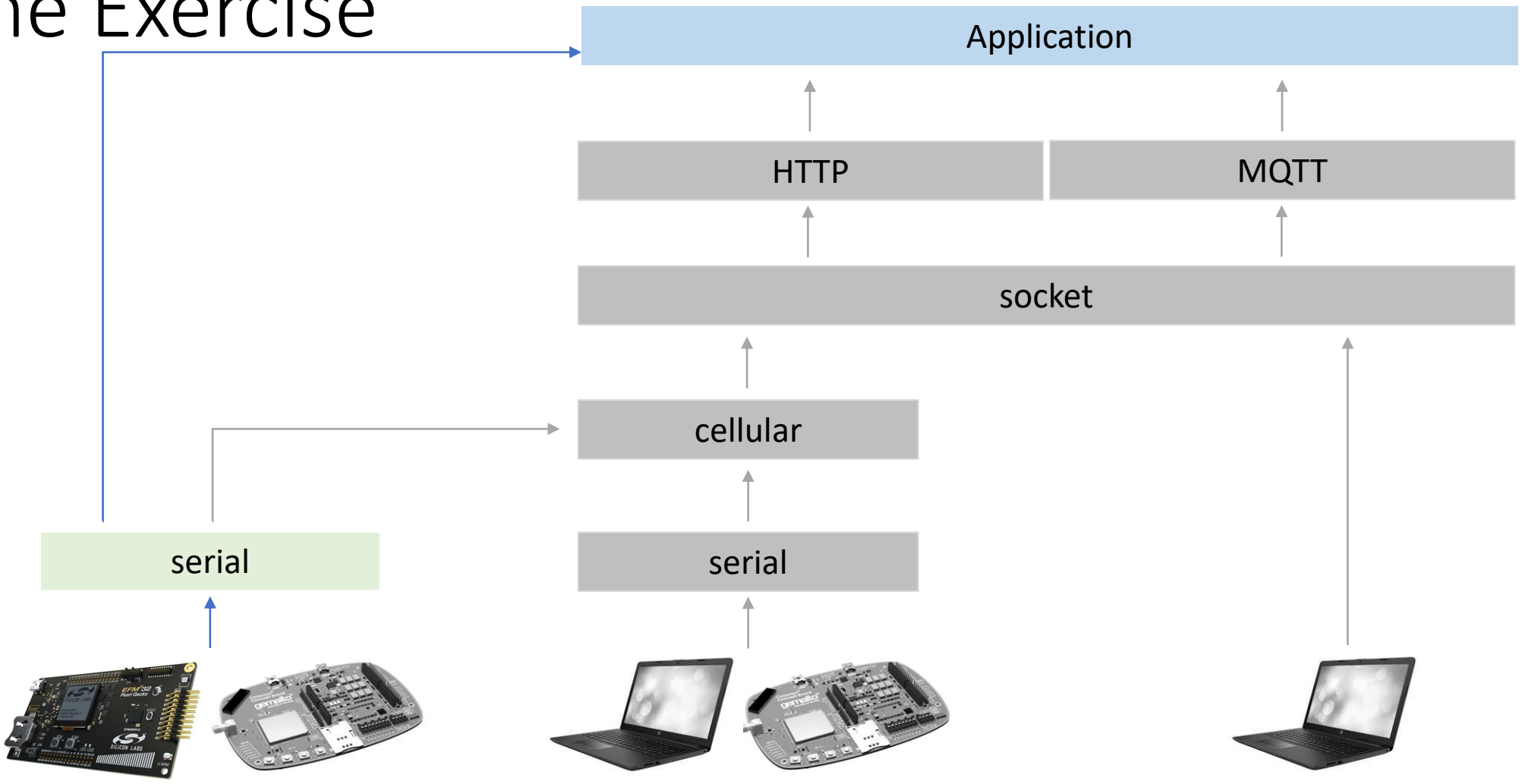
Embedded UART

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

The Exercise



The Exercise



The Exercise

- Send and receive data over UART
- React to BTN0 and BTN1
- React to timeout

Guidance

- **Read** [at least] the following chapters in the Pearl Gecko's Reference Manual:
- **Chapter 2 - System Overview**.
- **Chapters 16 and 17 – USART** and **LEUART**. Make sure you understand which registers does what (this is true for all peripherals!). For example, Register LEUARTn_RXDATA will hold the data that the modem throws at you, at least until the next byte comes in... (BTW, this "n" stands for the peripheral number, in practice, you will use something like "LEUART0->RXDATA").
- **General tip**: use a proper PDF reader (e.g., Adobe Acrobat Reader), and not a browser. Then you can navigate through the chapters/sections easily.

Guidance

- 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 the same as we already had before)
- Implement these functions in a file called serial_io_efm32pg12.c
- OK to assume a single thread (task), also OK to use global/static variables

Guidance cont’d

- Write code that does the following:
  - Runs an infinite loop
  - Every 10 seconds, prints on the screen the number of BTN0 clicks for the last 10 seconds
  - Clicking on BTN1 sends over UART the current-BTN0-clicks-count as a string: “number of clicks: <the-current-BTN0-clicks-count>”
  - The count is 0 if there were no clicks for the last 10 seconds
  - When something is received over UART, print: “received: <the-received-content>” (assume a reception of a string).
  - A new count starts every 10 seconds
  - The strings should be null terminated

  - “Send” and “Receive” are implemented with interrupts
  - The idea is to send data to ourselves (or to someone else) and print the received data
  - baud rate: **115200 (8N1)**
  - Use one of the **USART** peripherals (not LEUART) (Use it in an asynchronous mode, so it’s a UART, not U**S**ART)

Guidance cont’d

- Use Simplicity Studio v5. Note that it generates code that is very different from Simplicity Studio v4
- The following example-projects can help you:
  - Platform - I/O Stream USART Bare-metal (*use mostly this example*)
  - Platform - SLSTK3402A EFM32PG12B LEUART Echo (*this one has less abstraction than the previous example, so it can give you a better idea of how things work. Although, you can get all the interformation you need from the first example as well – it’s just a bit more work to get there*)

Guidance cont'd

- Read the EFM32 Pearl Gecko Starter Kit User's Guide and find the right pins. Be careful – connection to wrong pins can kill the hardware.

  Connect the TX (PA6) to RX (PA7)

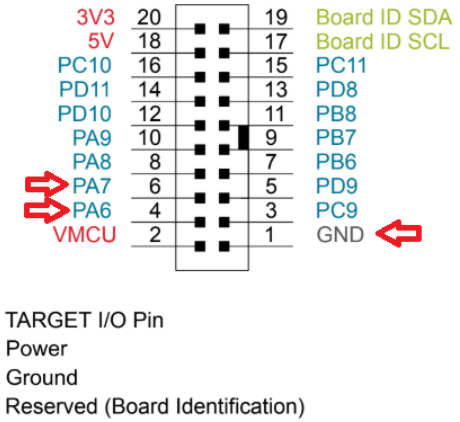


Figure 4.2. Expansion Header

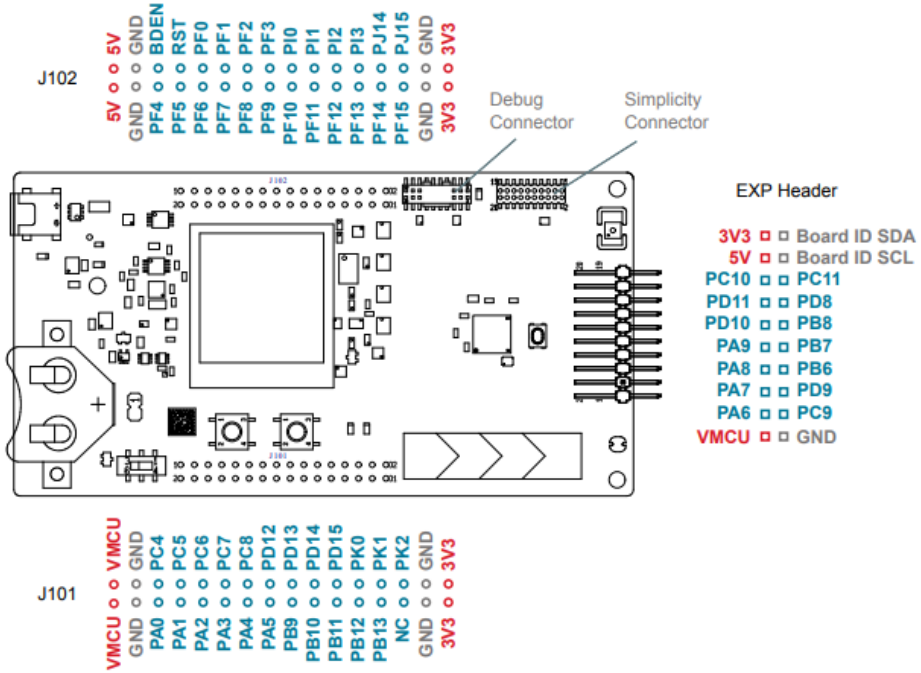


Figure 4.1. Breakout Pads and Expansion Header

Pin	Connection	EXP Header function	Shared feature	Peripheral mapping
20	3V3	Board controller supply		
18	5V	Board USB voltage		
16	PC10	I2C_SDA	SENSOR_I2C_SDA	I2C0_SDA #15
14	PD11	UART_RX		LEU0_RX #18
12	PD10	UART_TX		LEU0_TX #18
10	PA9	SPI_CS		USART2_CS #1
8	PA8	SPI_SCLK		USART2_CLK #1
6	PA7	SPI_MISO		USART2_RX #1
4	PA6	SPI_MOSI		USART2_TX #1
2	VMCU	EFM32 voltage domain, included in AEM measurements.		
19	BOARD_ID_SDA	Connected to Board Controller for identification of add-on boards.		
17	BOARD_ID_SCL	Connected to Board Controller for identification of add-on boards.		
15	PC11	I2C_SCL	SENSOR_I2C_SCL	I2C0_SCL #15
13	PD8	GPIO		USART3_CS #29

Exercise #8

- Work & submit in pairs
- Deliverables:
  - Provide .sls and .bin files (project source code and definitions, and the compiled version)
  - 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: 28.1.22, 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