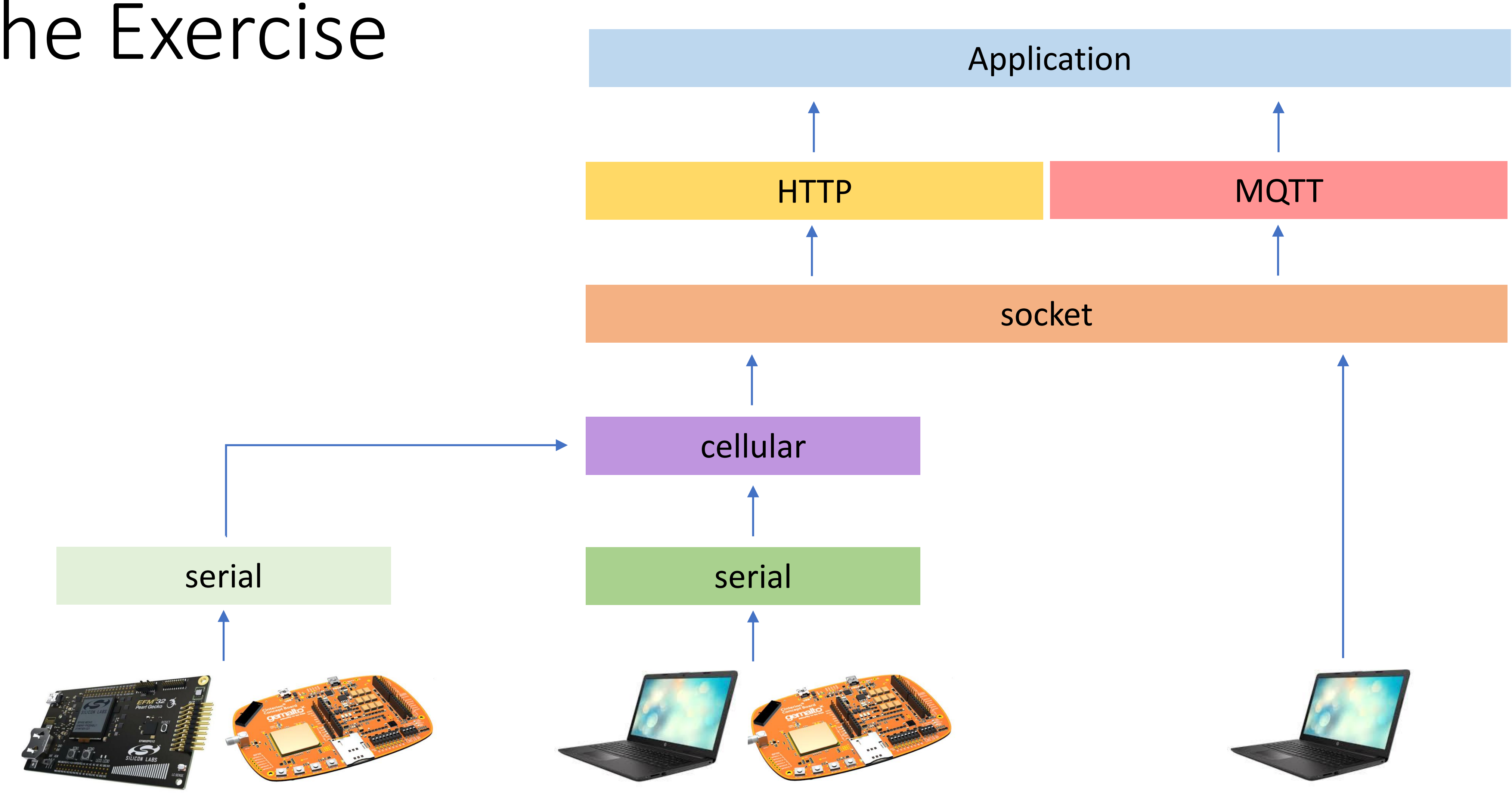
WORKSHOP ON **INTERNET OF THINGS** 67612

Exercise 9

IoT

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

The Exercise



Application

HTTP

MQTT

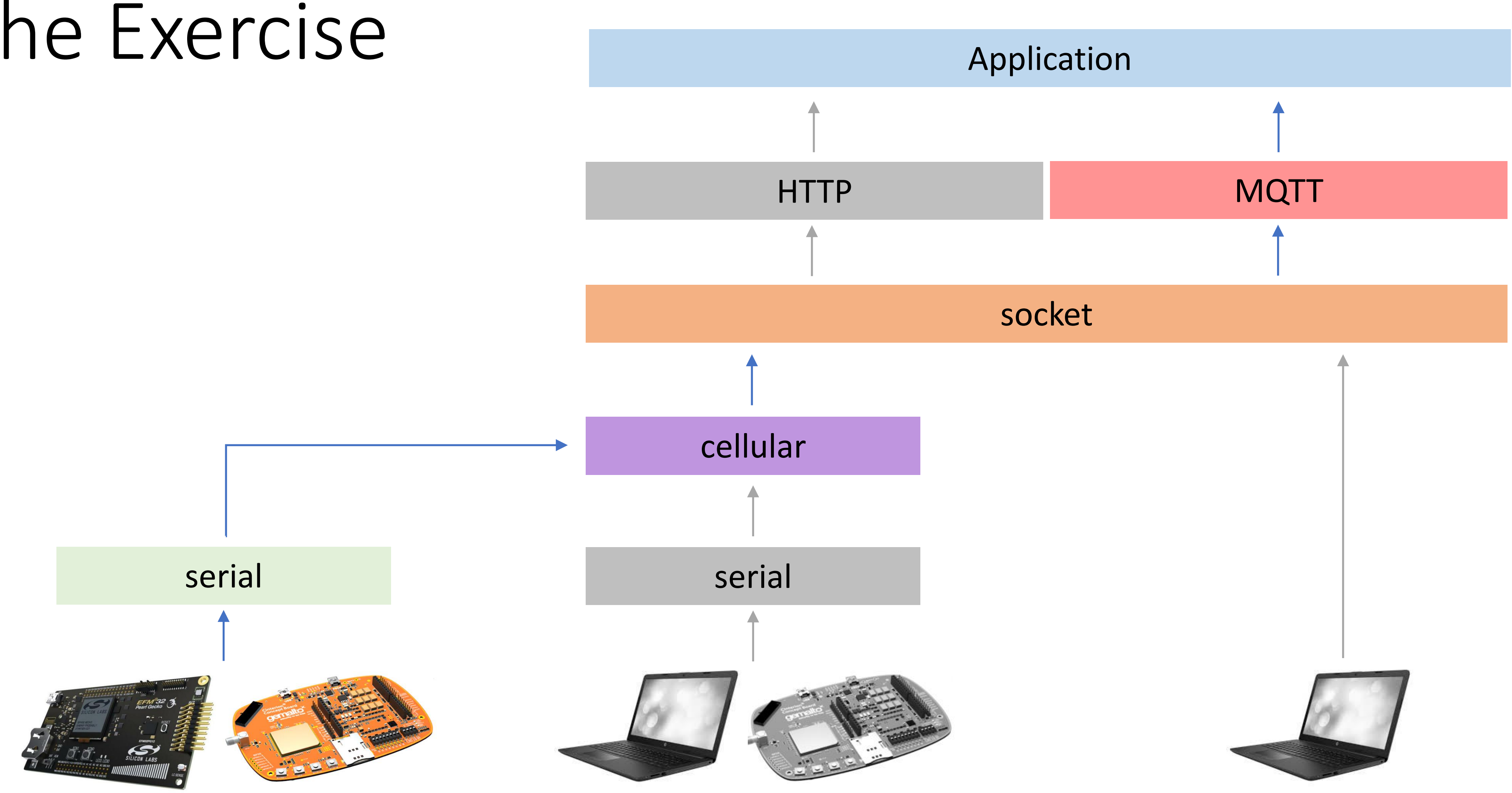
socket

cellular

serial

serial

The Exercise



Application

HTTP

MQTT

socket

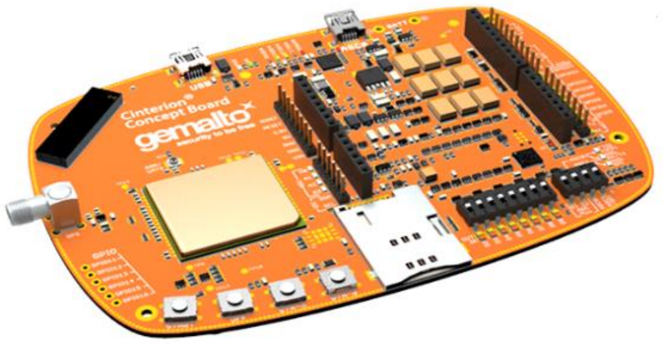
cellular

serial

serial

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

- Tie it all together
- Write code the does the following (Basically, the same steps as in ex6. See a note about the hostname in the next slides):
  - Runs an infinite loop
  - 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 (start over)

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
    - ***Note that since we don’t resolve DNS, you’ll have to use the IP address***
  - 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

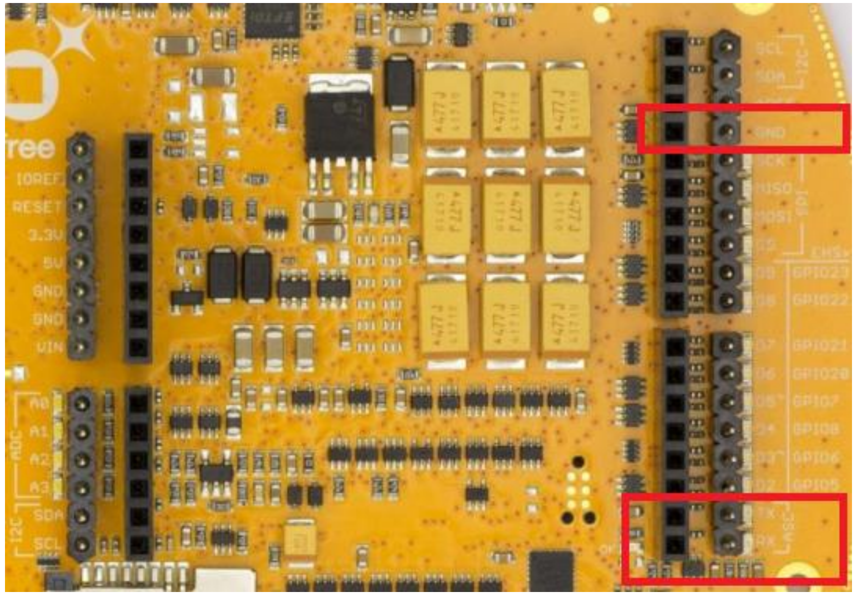
- Connect:
  - No username and password
  - Keep alive = 60
  - Clean session = 0
  - Generate any client-id
  - Use LWT
- LWT:
  - QoS = 1
  - No retain message
- Publish:
  - QoS = 1
  - No retain message
- Subscribe:
  - QoS = 1

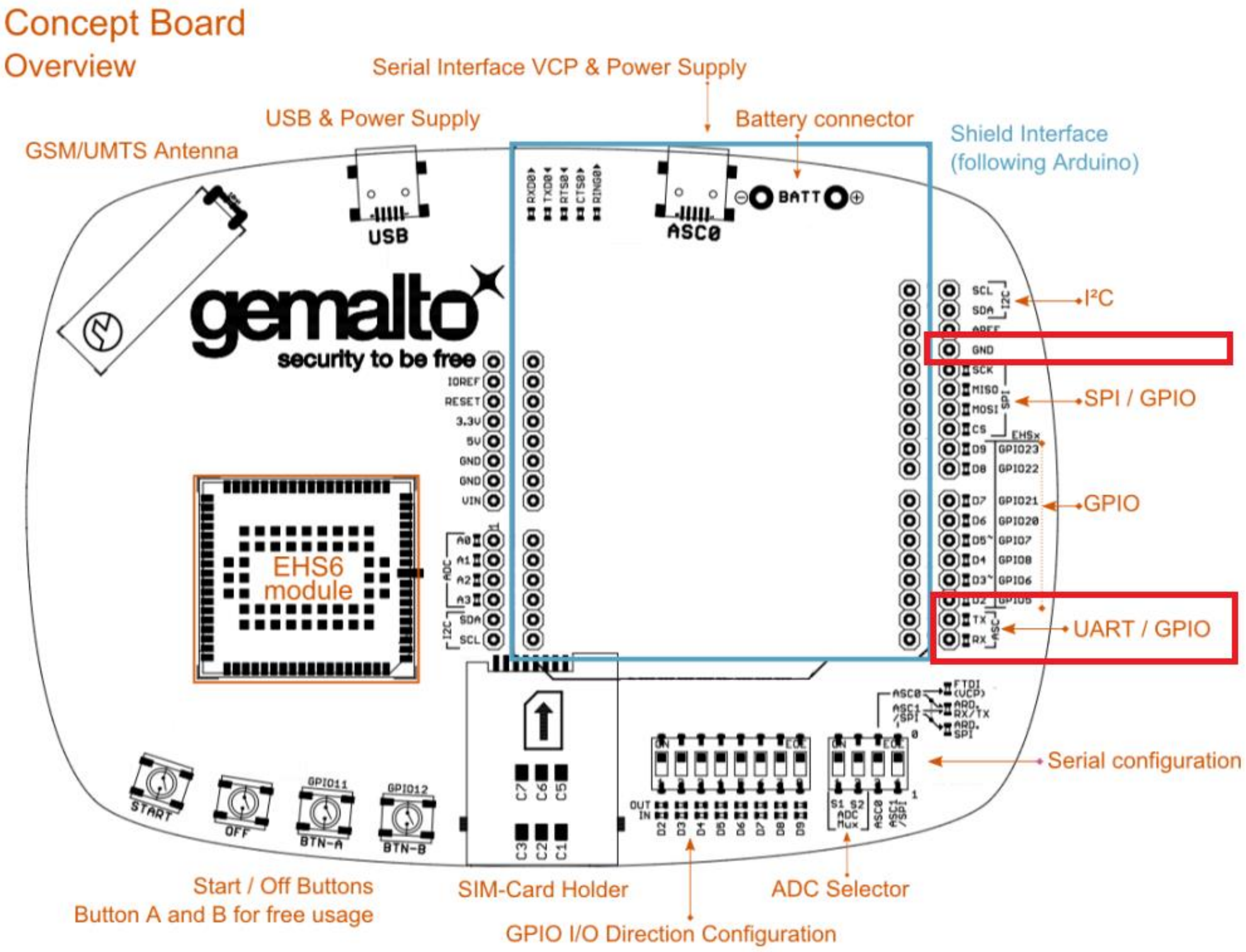
Guidance cont’d

- This time we’ll need three wires: RX, TX, GND.
    - Note: these are not just wires! (more about it in the next slides)

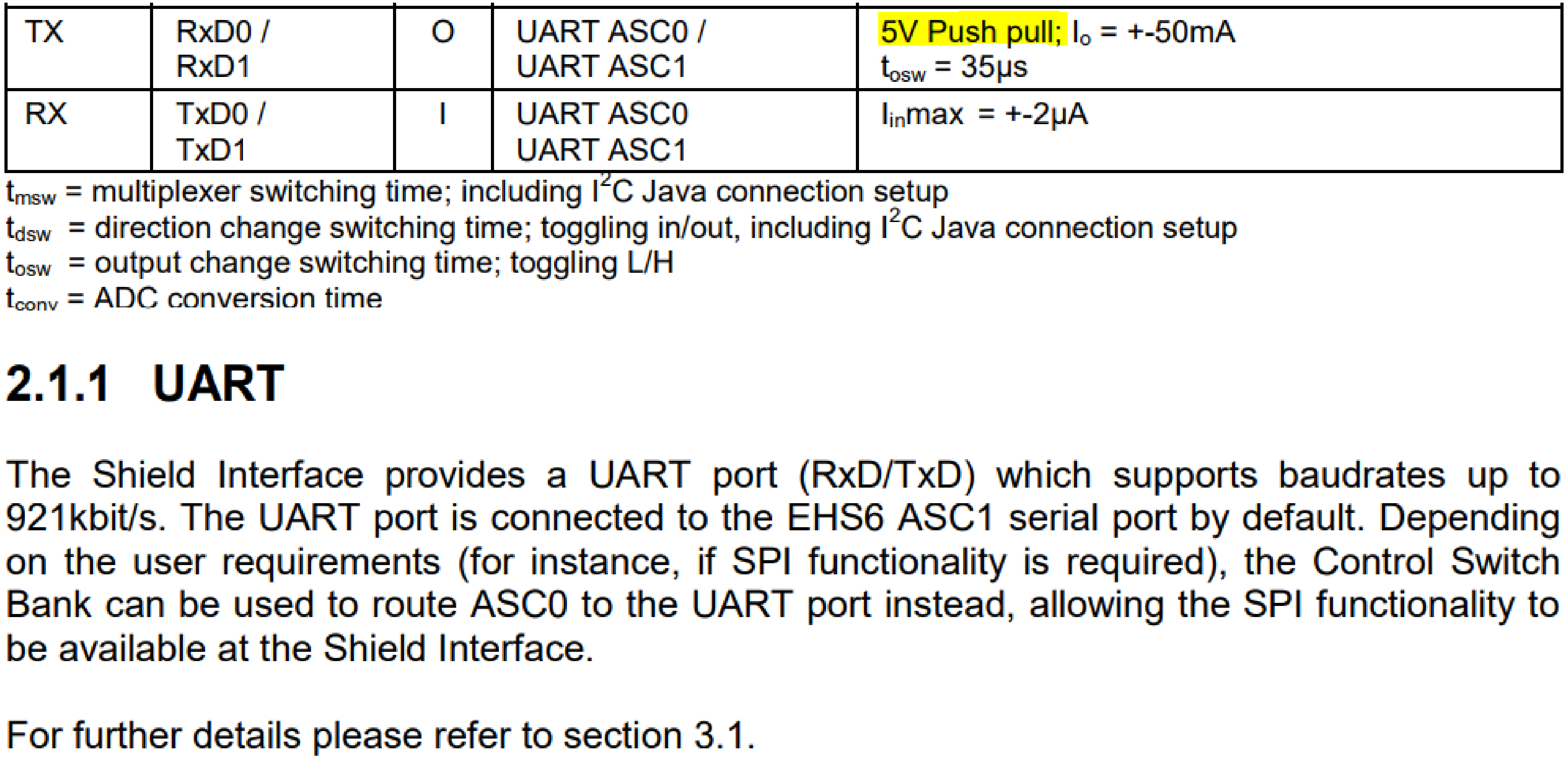
Guidance cont’d

Read the Cinterion Concept Board Hardware Interface Description and Cinterion Concept Board Start-up Guide and find the right pins. Be careful – connection to wrong pins can kill the hardware.





Guidance cont’d



TX	RxD0 / RxD1	O	UART ASC0 / UART ASC1	5V Push pull; I_o = +-50mA t_{osw} = 35µs
RX	TxD0 / TxD1	I	UART ASC0 UART ASC1	I_{in}max = +-2µA

t_{msw} = multiplexer switching time; including I^2C Java connection setup
t_{dsw} = direction change switching time; toggling in/out, including I^2C Java connection setup
t_{osw} = output change switching time; toggling L/H
t_{conv} = ADC conversion time

2.1.1 UART

The Shield Interface provides a UART port (RxD/TxD) which supports baudrates up to 921kbit/s. The UART port is connected to the EHS6 ASC1 serial port by default. Depending on the user requirements (for instance, if SPI functionality is required), the Control Switch Bank can be used to route ASC0 to the UART port instead, allowing the SPI functionality to be available at the Shield Interface.

For further details please refer to section 3.1.

Guidance cont’d

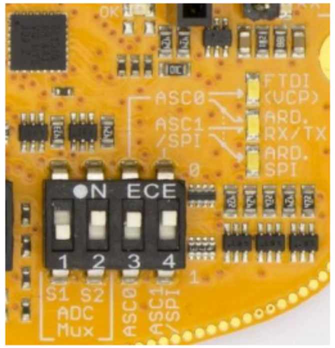
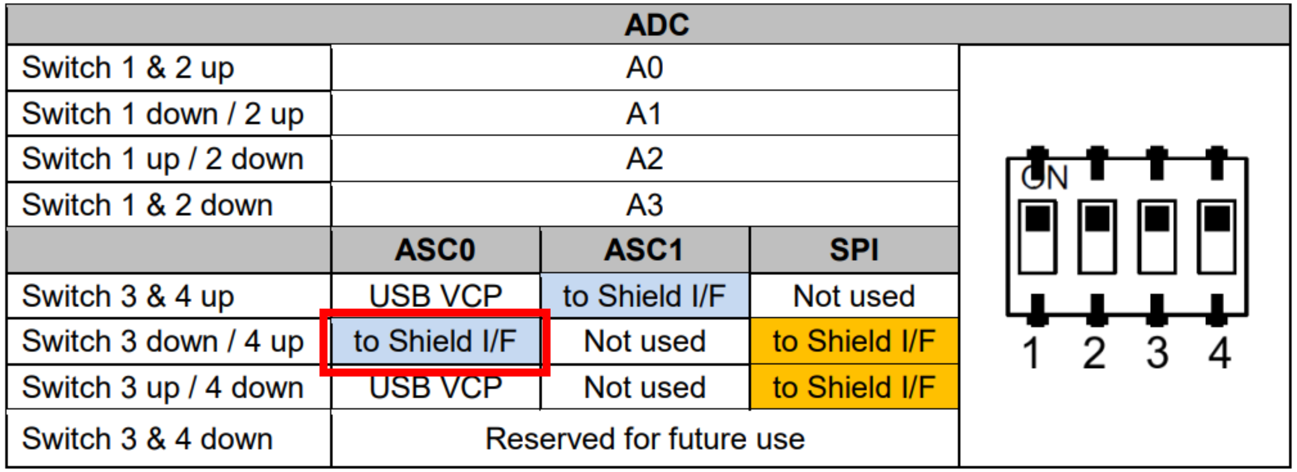


Figure 5: configuration switch bank and corresponding LEDs

Table 5: Concept Board configuration for ASC/SPI and ADC



ADC				
Switch 1 & 2 up	A0			
Switch 1 down / 2 up	A1			
Switch 1 up / 2 down	A2			
Switch 1 & 2 down	A3			
	ASC0	**ASC1**	**SPI**	
Switch 3 & 4 up	USB VCP	to Shield I/F	Not used	
Switch 3 down / 4 up	to Shield I/F	Not used	to Shield I/F	
Switch 3 up / 4 down	USB VCP	Not used	to Shield I/F	
Switch 3 & 4 down	Reserved for future use			

Guidance cont’d

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

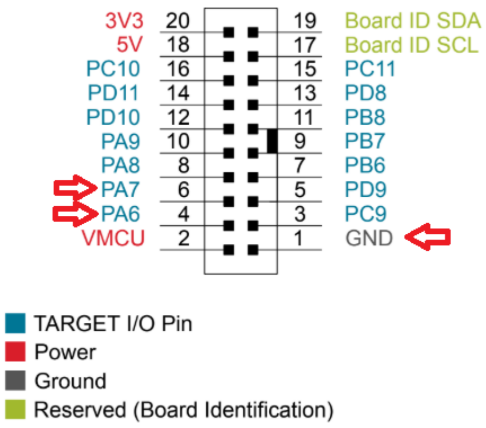
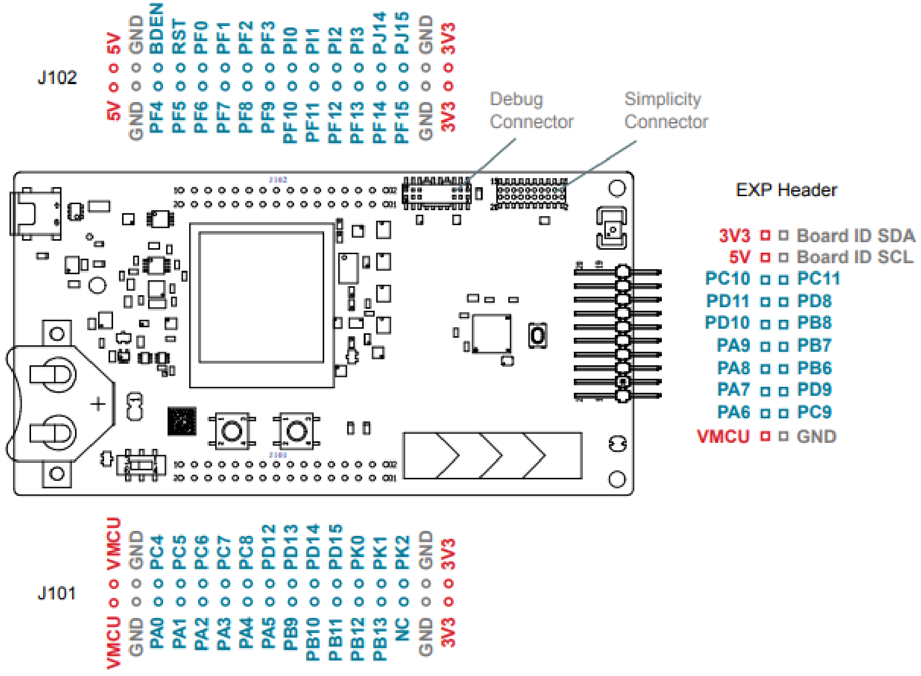
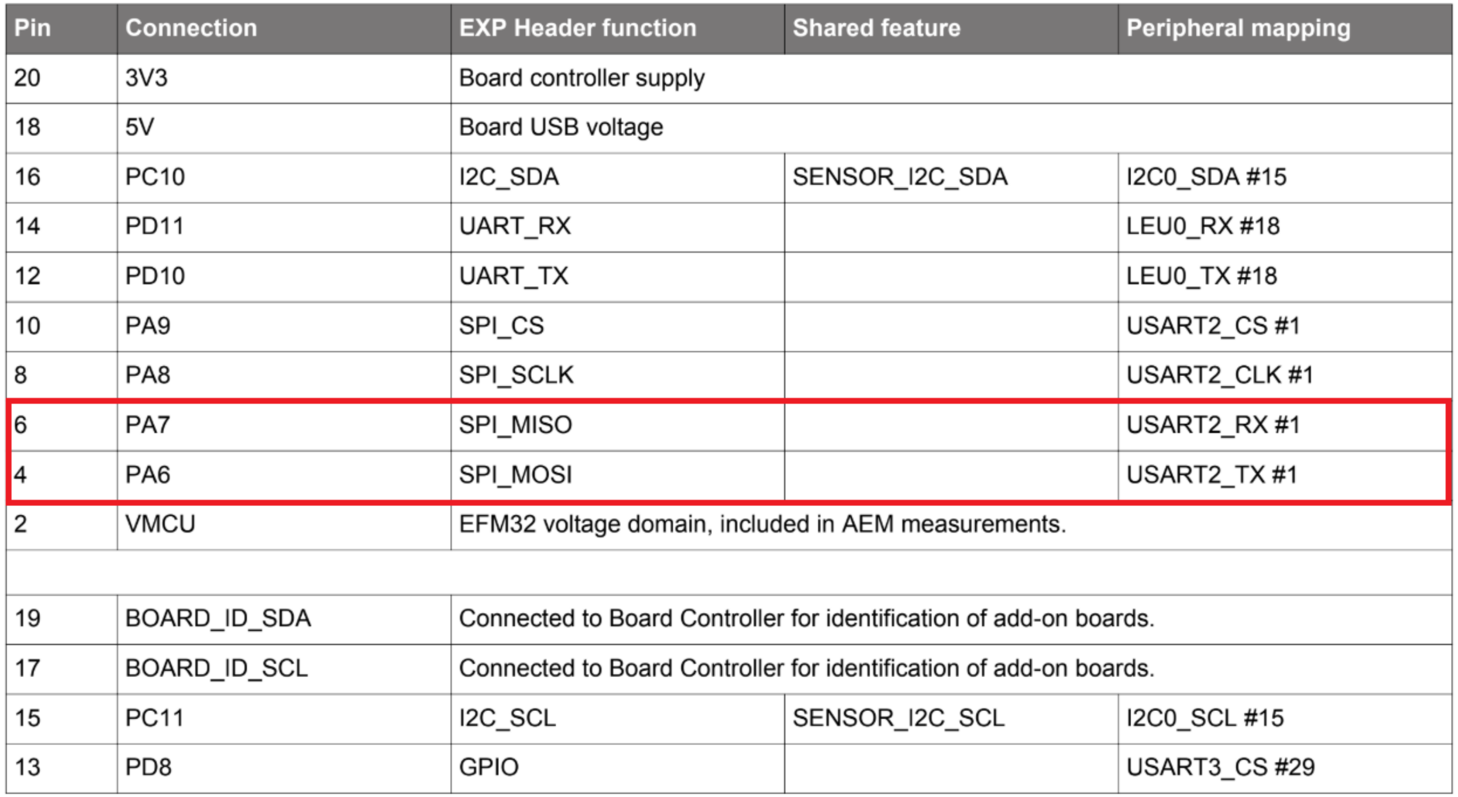


Figure 4.2. Expansion Header



J102
5V GND BDEN RST PF0 PF1 PF2 PF3 PI0 PI1 PI2 PI3 PJ14 PJ15 GND 3V3
5V GND PF4 PF5 PF6 PF7 PF8 PF9 PF10 PF11 PF12 PF13 PF14 PF15 GND 3V3
Debug Connector
Simplicity Connector
EXP Header
3V3 Board ID SDA
5V Board ID SCL
PC10 PC11
PD11 PD8
PD10 PB8
PA9 PB7
PA8 PB6
PA7 PD9
PA6 PC9
VMCU GND
J101
VMCU GND PC4 PC5 PC6 PC7 PC8 PD12 PD13 PD14 PD15 PK0 PK1 PK2 GND 3V3
VMCU GND PA0 PA1 PA2 PA3 PA4 PA5 PB9 PB10 PB11 PB12 PB13 NC GND 3V3

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

Guidance cont’d

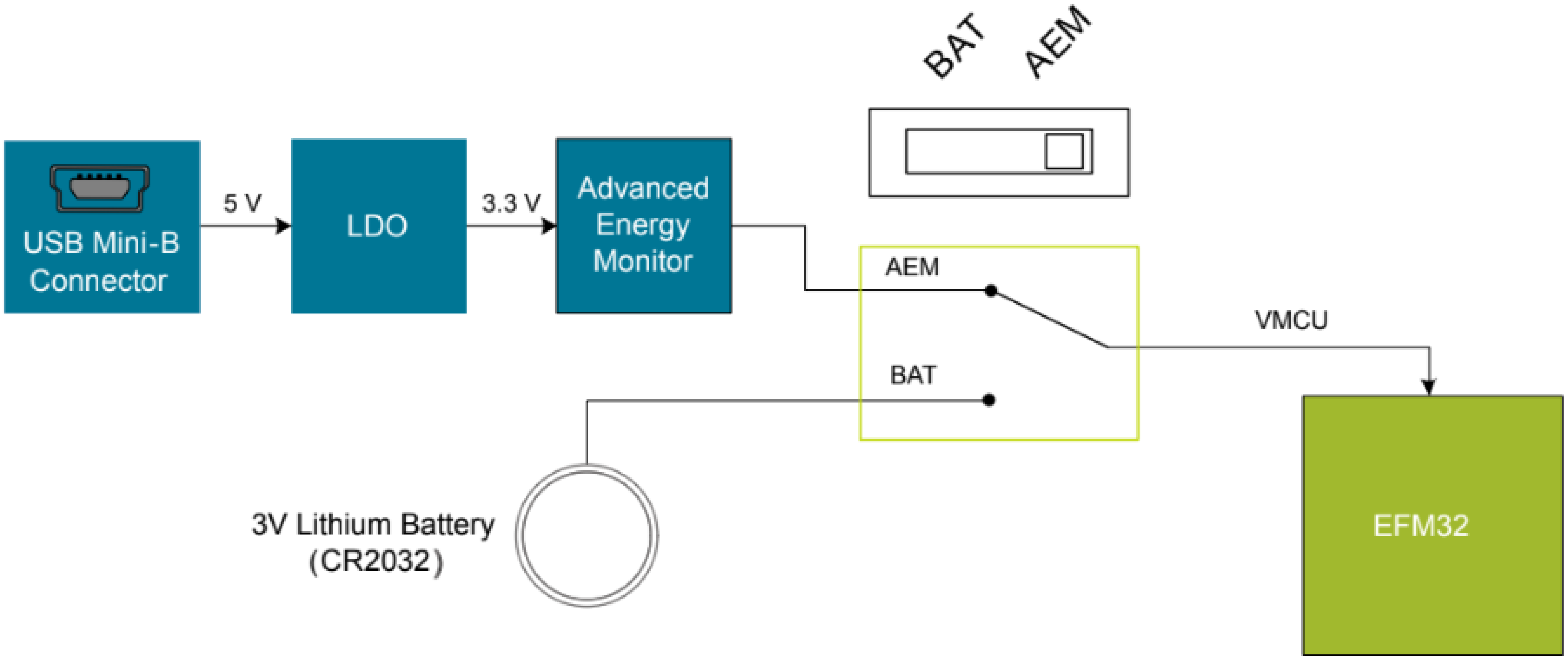
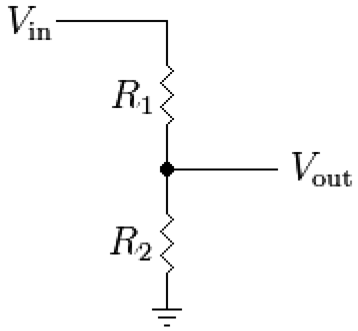


Figure 5.1. Power Switch

Guidance cont’d

- Modem TX (5V) → STK RX (3.3V)
- A voltage divider is needed.
- A voltage divider is a passive linear circuit that produces an output voltage (Vout) that is a fraction of its input voltage (Vin). Voltage division is the result of distributing the input voltage among the components of the divider
- Ohms law: $I = \frac{V}{R}$ or $V = IR$ or $R = \frac{V}{I}$
- In our circuit, the current between Vin and GND is: $I = \frac{V_{\text{in}}}{R_1 + R_2}$
- So the voltage between Vout and GND is: $V_{\text{out}} = V_2 = I_2 \cdot R_2 = (\frac{V_{\text{in}}}{R_1 + R_2}) \cdot R_2 = V_{\text{in}} \cdot \frac{R_2}{R_1 + R_2}$
- In short: $V_{\text{out}} = V_{\text{in}} \cdot \frac{R_2}{R_1 + R_2}$

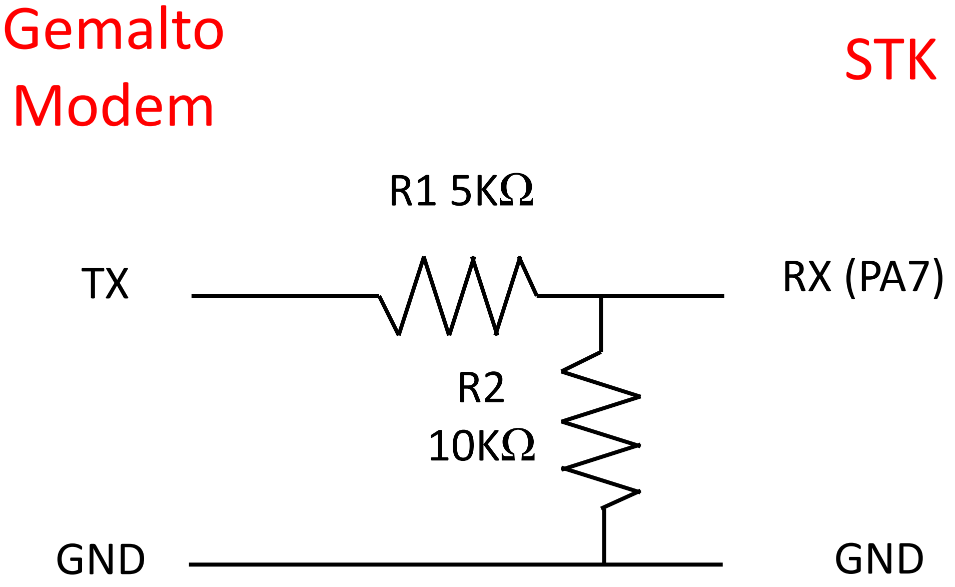


Guidance cont'd

- Modem TX (5V) → STK RX (3.3V)

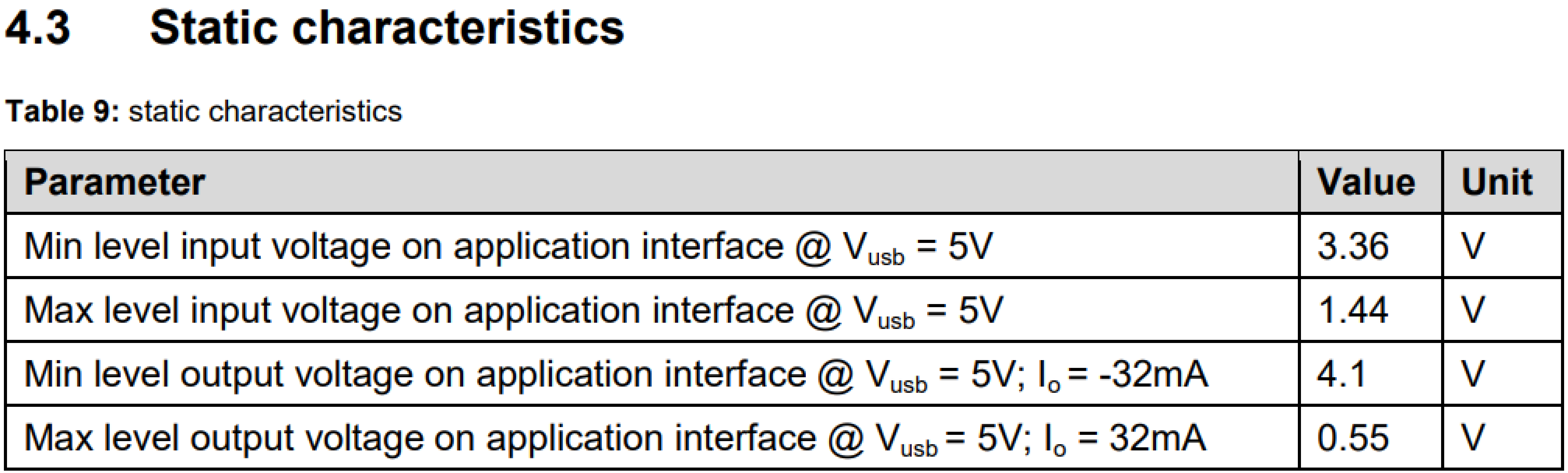
$$V_{\text{out}} = V_{\text{in}} \cdot \frac{R_2}{R_1 + R_2}$$

$$5 \cdot \frac{10K}{5K + 10K} = 5 \cdot \frac{10K}{15K}$$
$$= 5 \cdot \frac{2}{3} = \frac{10}{3} = 3\frac{1}{3} = 3.333V$$



Guidance cont’d

- Modem RX (5V) ← STK TX (3.3V)



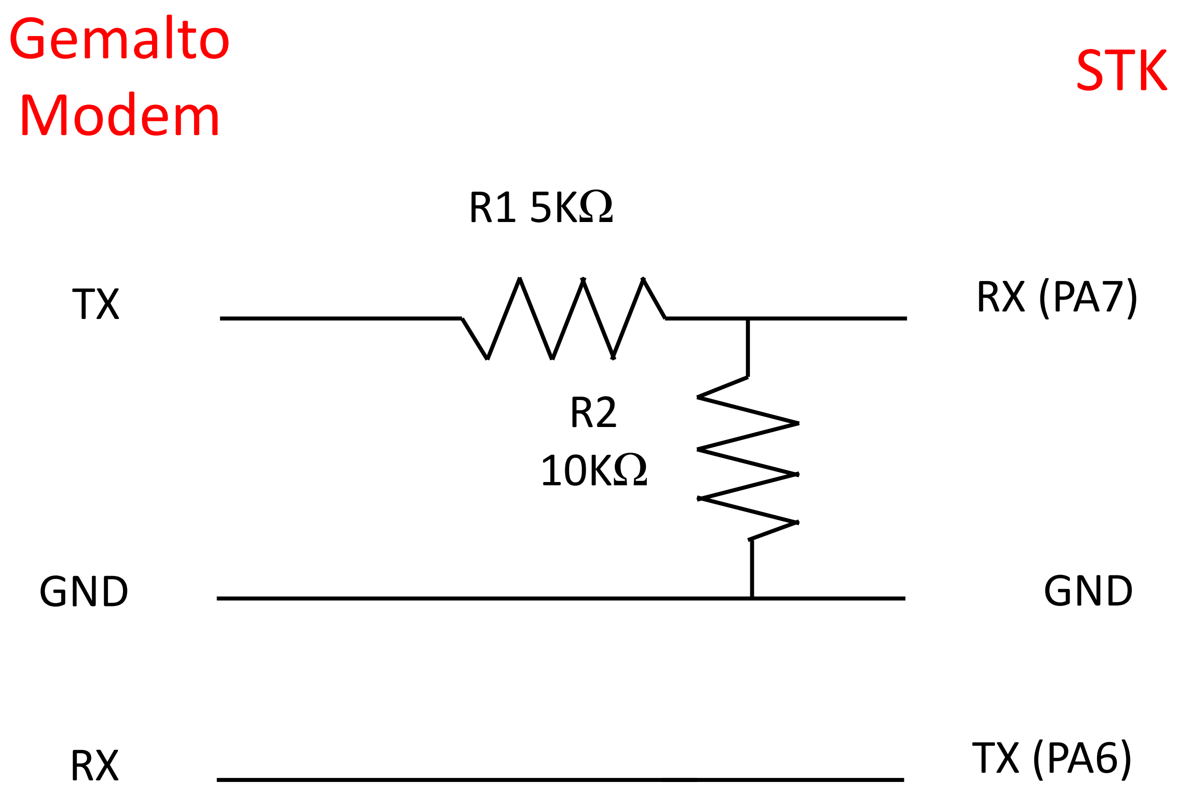
4.3 Static characteristics

Table 9: static characteristics

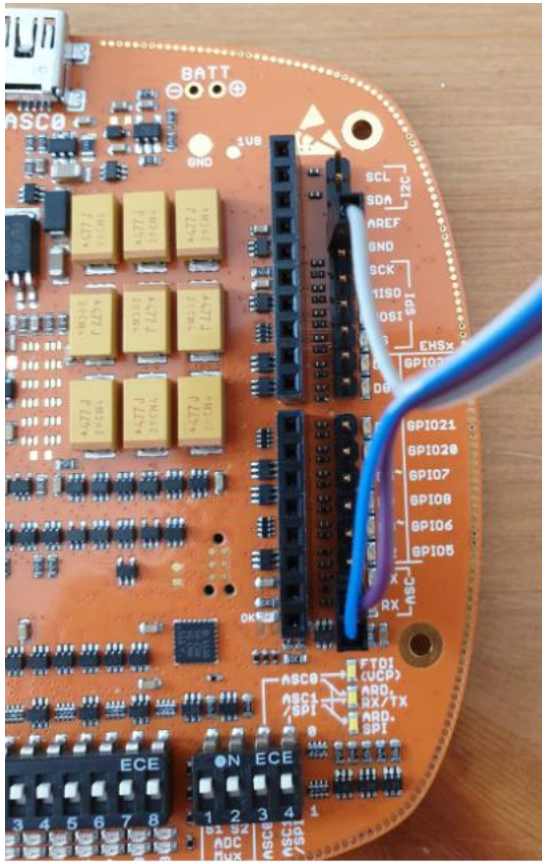
Parameter	Value	Unit
Min level input voltage on application interface @ V_{usb} = 5V	3.36	V
Max level input voltage on application interface @ V_{usb} = 5V	1.44	V
Min level output voltage on application interface @ V_{usb} = 5V; I_o = -32mA	4.1	V
Max level output voltage on application interface @ V_{usb} = 5V; I_o = 32mA	0.55	V

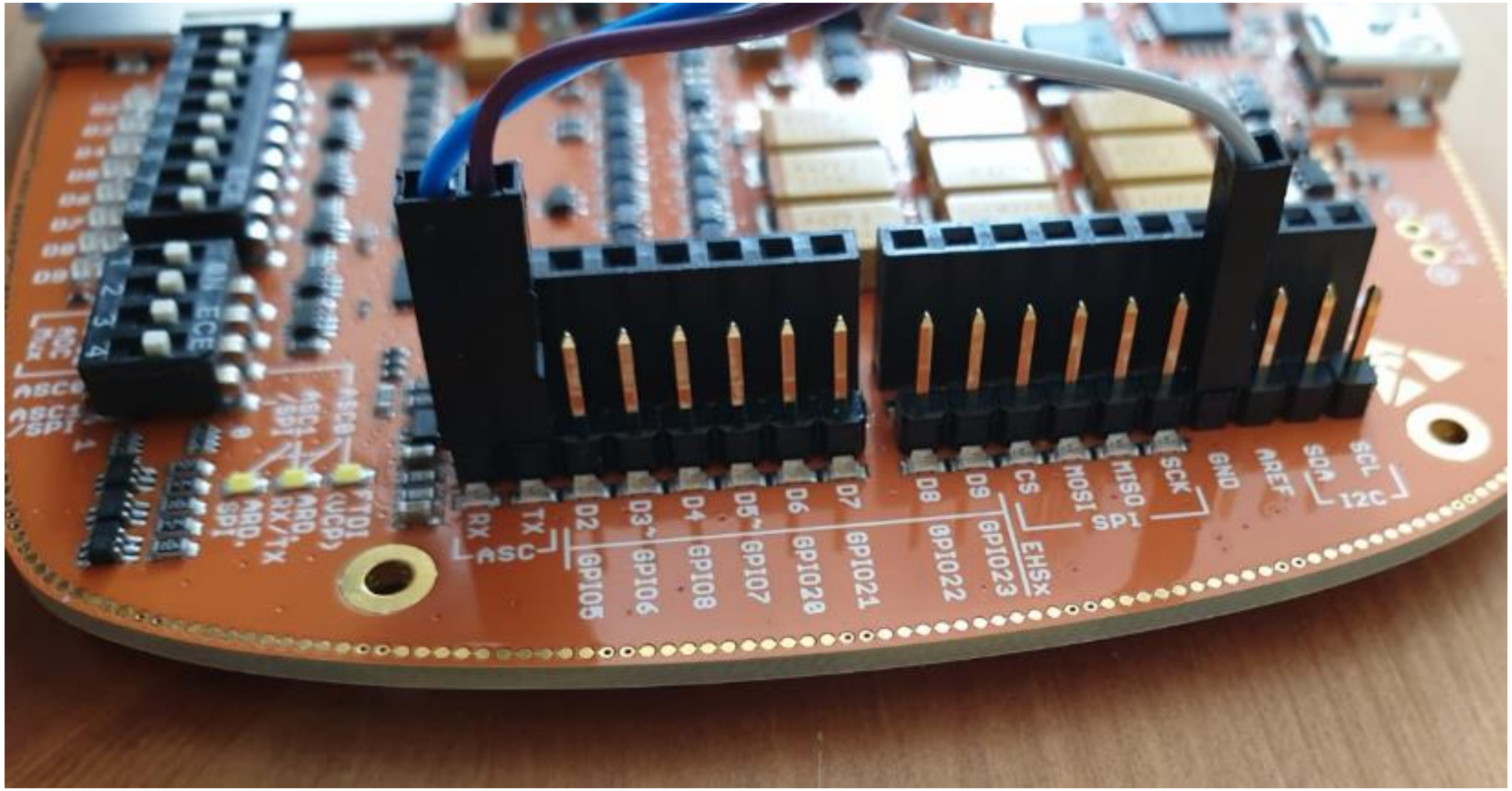
- 3.36V is needed to received a signal but… we got 3.3V.
- Let’s just connect it and see if it works! (it works)

Guidance cont’d

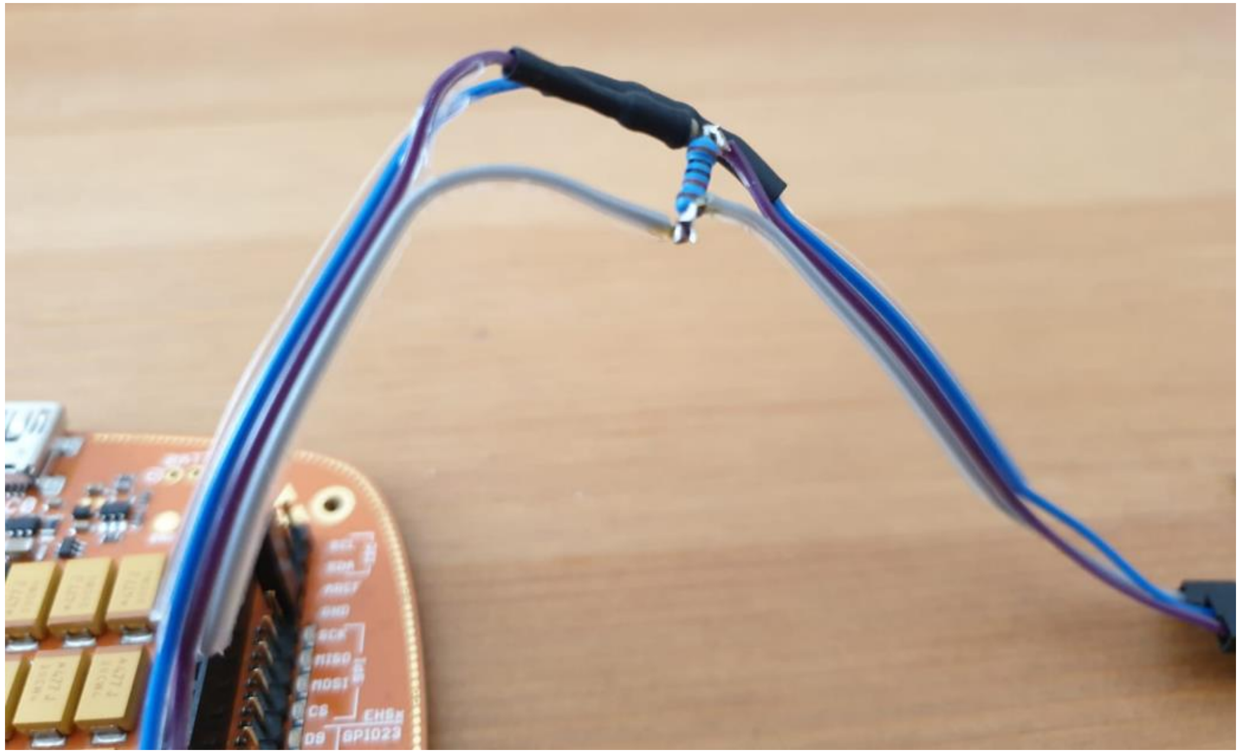


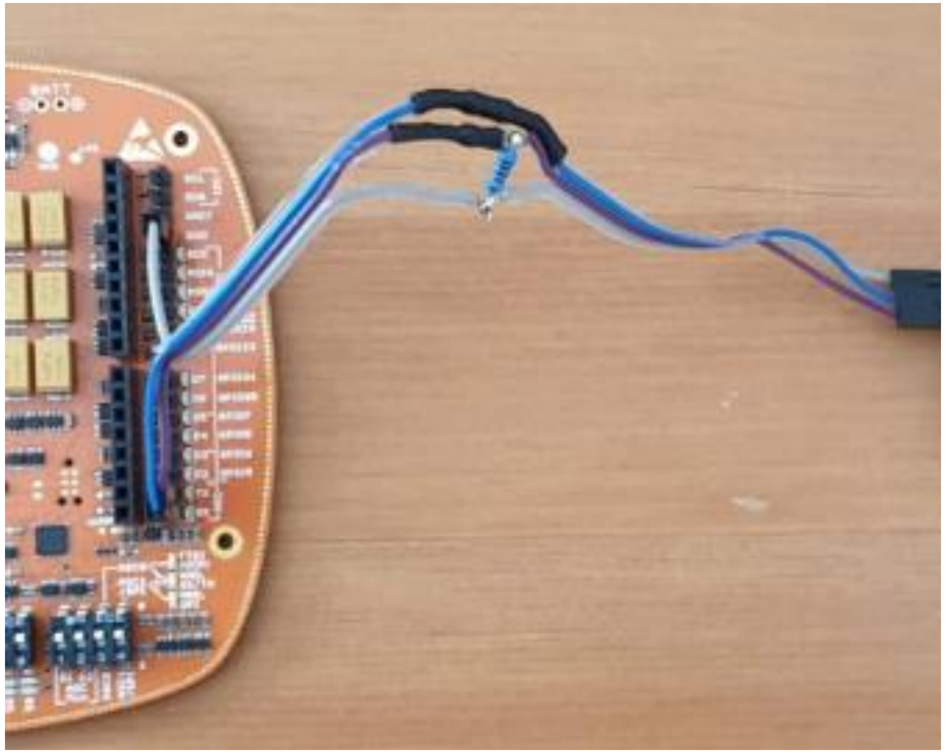
Guidance cont’d



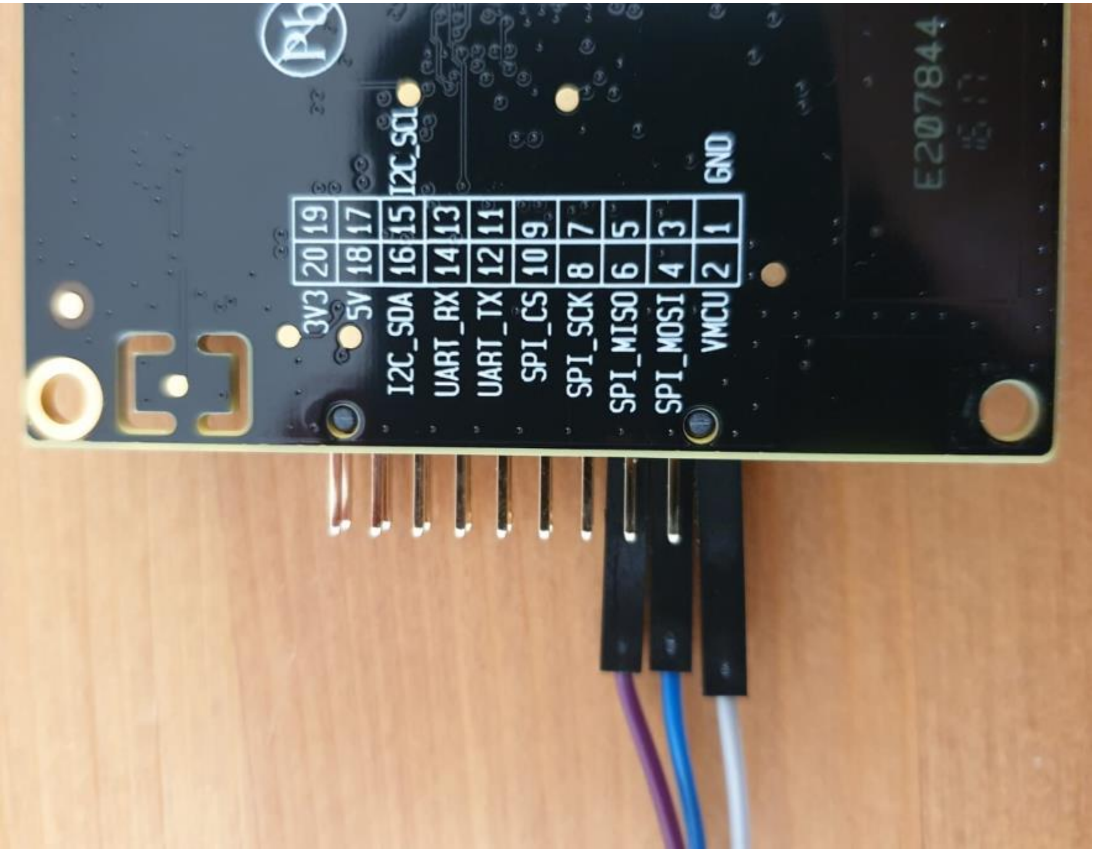


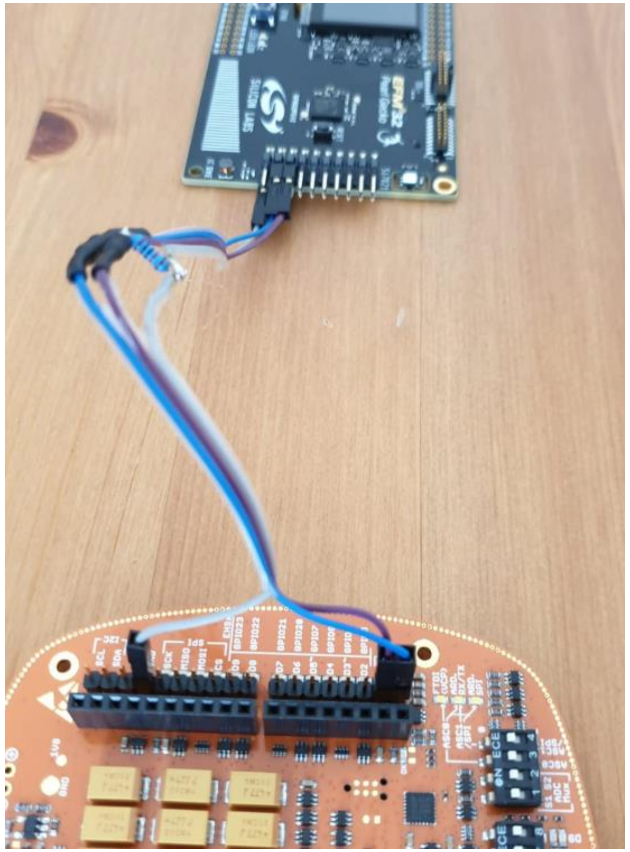
Guidance cont’d

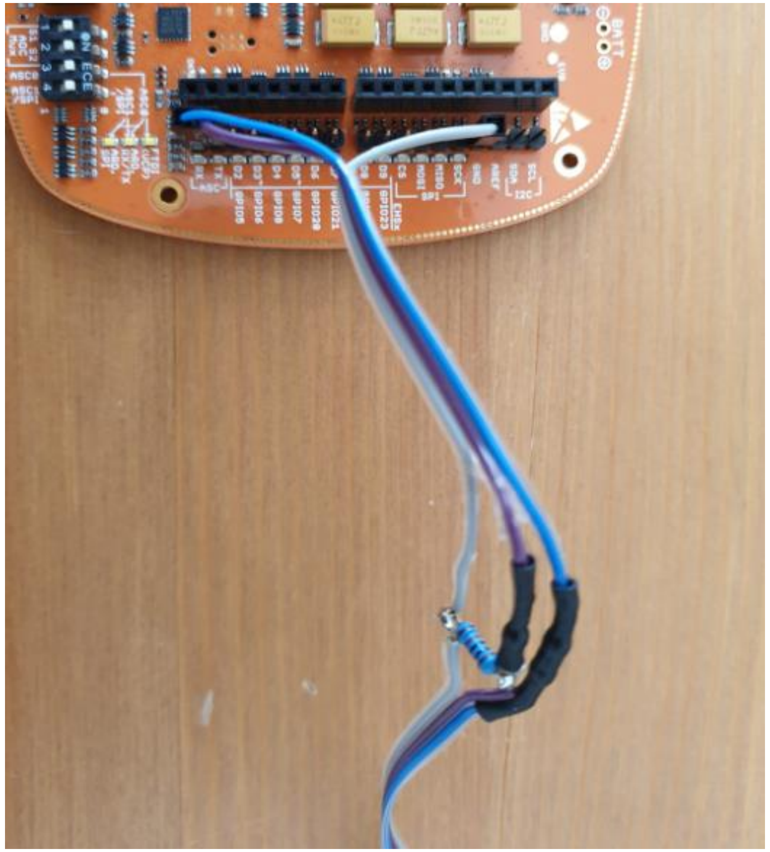




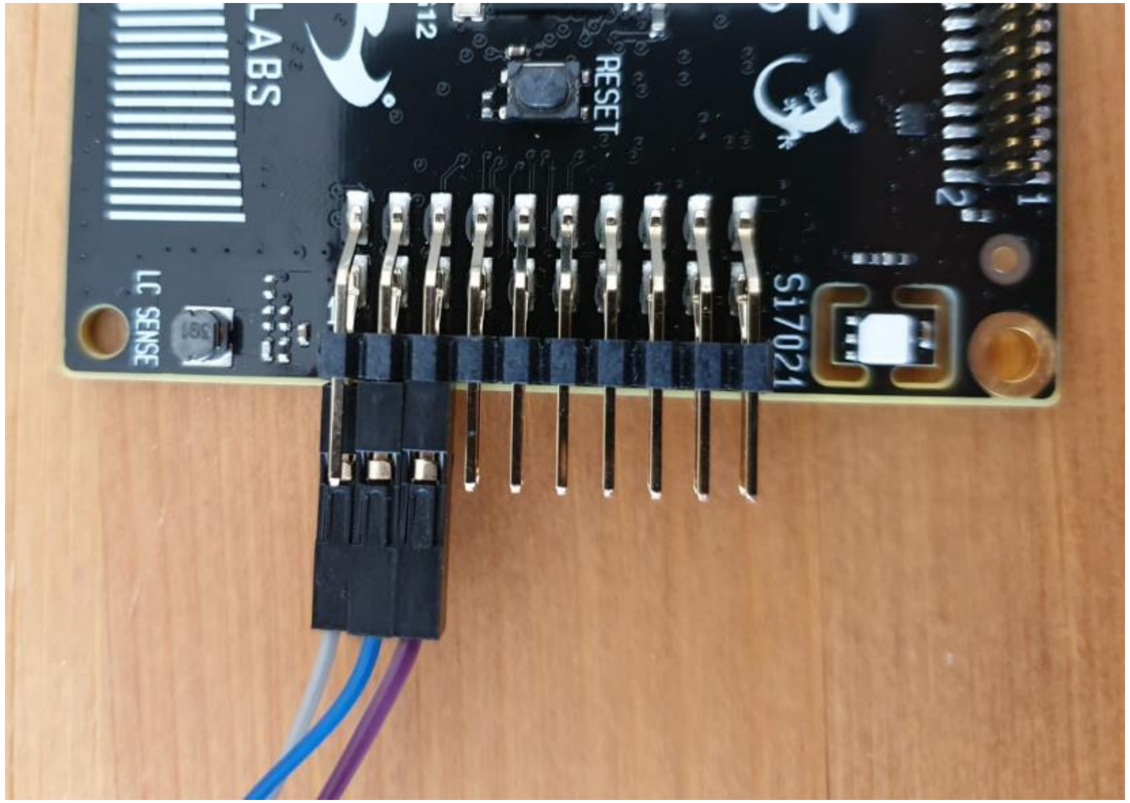
Guidance cont’d

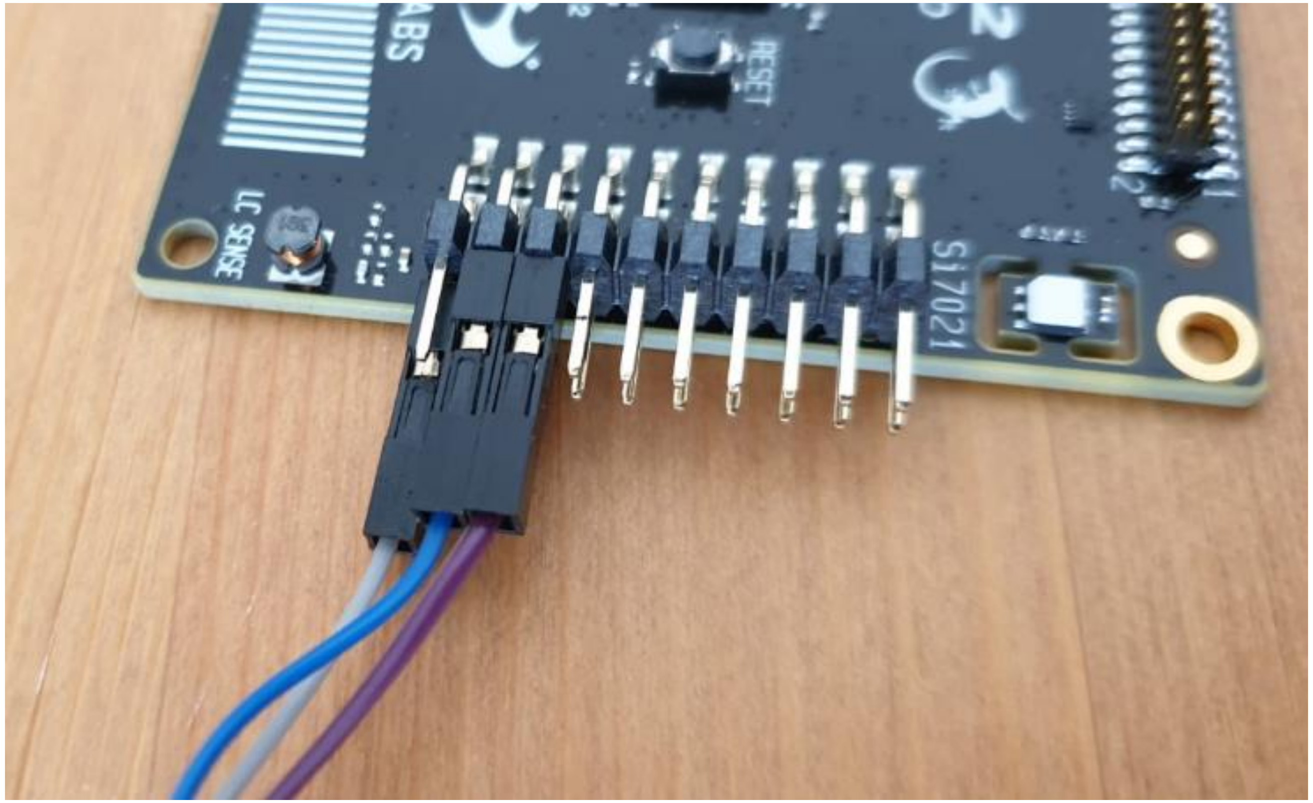




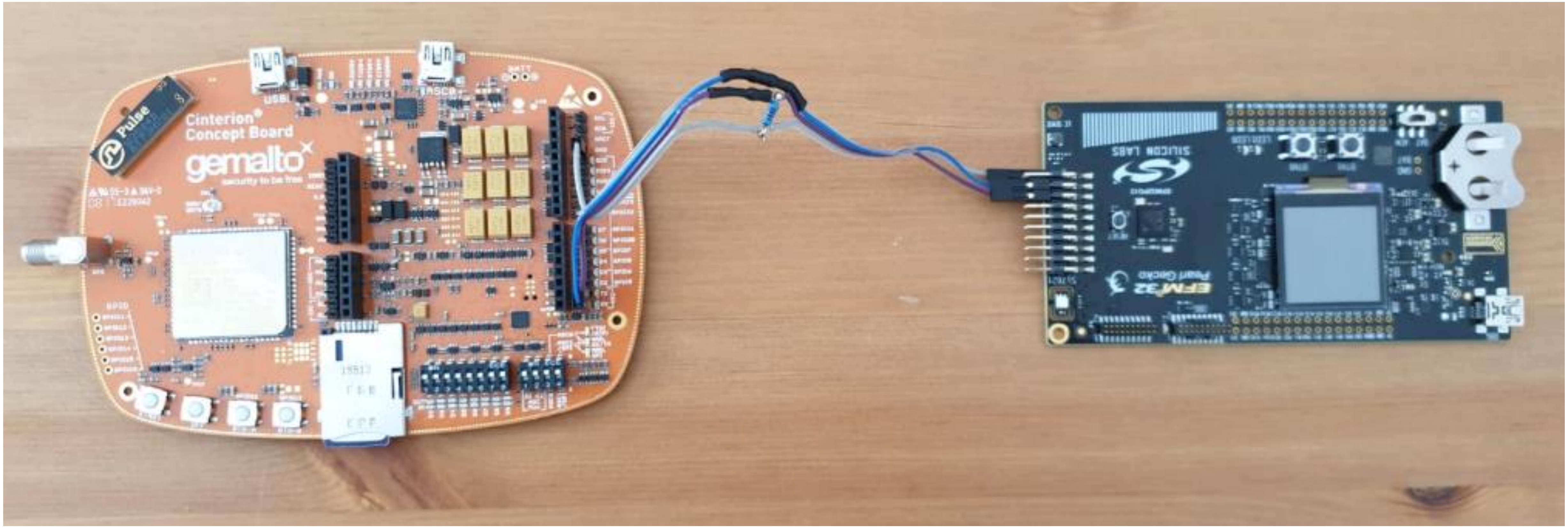


Guidance cont’d





Guidance cont’d



Exercise #9

- 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: 11.1.22, 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