

Introduction to IoT

School Year 2023-2024

Valsalice



Course Structure

1	Introduction and Basics
2	Basic Data Types and Operators
3	Control Structures Pt. 1
4	Control Structures Pt. 2
5	Functions and Scope
6	Arrays
10	Advanced String Usage
11	Custom Data Types

7	Introduction to Contiki-NG and nRF52840			
8	Sensing and Actuating with Contiki-NG			
9	Timers and Concurrency			
12	Basic Communication and Networking	JAN	16	
13	Introduction to the MAC Layer	JAN	23	
14	Introduction to RPL and Network Routing	JAN	30	
15	Advanced Protocols: TSCH and 6TiSCH	FEB	6	
16	Reliable Data Transfer Challenge	FEB	20	FEB 27 MAR 5




= Core Topics



= Optional Topics

Open your Virtual Machines

1. Turn on your Laptops
2. Login to Windows using "User"
3. Open the **Virtual Box** program
4. Add a new Virtual Machine (**Ctrl + A**)
5. Open the **VirtualBox** folder  (**NOT** the .VirtualBox)
6. Select the **nRF52840LAB** file
7. Click **Start**

Prepare the Coding Environment

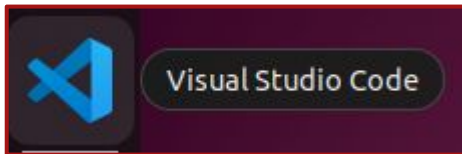
1 Start the Virtual Machine **nRF52840LAB**

2 Log-in using credentials:

Username: **ubuntu**

Password: **ubuntu**

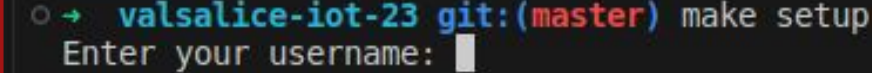
3 Open **Visual Studio Code** (use the App bar on the left)



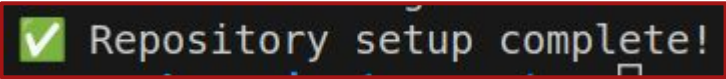
Prepare the Coding Environment

4 From the Terminal:

```
make setup
```

5 
valsalice-iot-23 git:(master) make setup
Enter your username:

6 
Password

7 
✓ Repository setup complete!



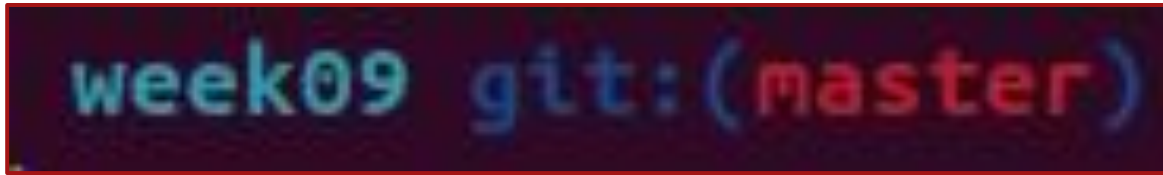
If you see **any (yellow) errors** input the credentials again

Prepare the Coding Environment

- 7 Open the **week12** folder in the terminal

Right click on the left + “**Open in Integrated terminal**”

- 8 You should see the following in the terminal:

A screenshot of a terminal window with a dark background. The text 'week09' is displayed in a light blue font, and 'git:(master)' is displayed in a red font.

Recap: Data Types

C has a number of primitive data types:

int

42

1200

1_200

-3

float

3.14

0.00001

-2.1

char

'A'

'@'

'\n'

bool

true

false

Strings are *NOT* a primitive data type, and have special syntax.

strings

"Hello"

"A"

"I am a full sentence!"

Recap: Variables

A variable is a named container that stores data or values.

```
int x = 42;  
float y = -0.12;  
char w = 'A';  
char z[50] = "Full sentence";
```

Booleans require a custom include statement:

```
#include <stdbool.h>  
bool hello = true;
```


Recap: Boolean Operators

Greater than	>
Greater or equal than	>=
Less than	<
Less or equal than	<=
Equals	==
Not equals	!=
Not	!

Recap: Chaining Comparisons

- **and** (both must be true)

```
true && false
```

```
(5 < 6) && (5 < 10)
```

- **or** (either must be true)

```
true || false
```

```
(5 < 3) || (5 < 10)
```

- **not** (negation)

```
!true
```

```
!(5 < 3)
```

Recap: If-Statement chaining

You can chain multiple conditions with **else if**.

What is the difference between these two snippets of code?

```
int num;
scanf("%d", &num);

if (num < 3) {
    printf("Small number\n");
} else if (num < 10) {
    printf("Medium number\n");
}
```

```
int num;
scanf("%d", &num);

if (num < 3) {
    printf("Small number\n");
}
if (num < 10) {
    printf("Medium number\n");
}
```

Recap: While-Loops

Repeat parts of
your code!

```
int num;  
printf("Input a number greater than 100: ");  
scanf("%d", &num);  
  
while (num <= 100) {  
    printf("Wrong number, try again: ");  
    scanf("%d", &num);  
}  
  
printf("Well done!\n");
```

Recap: For-Loops

Repeat a **specific** amount of times!

```
int x;

for (x = 1; x <= 5; x++) {
    printf("Hello %d\n", x);
}
```

```
int x = 0;

while (x < 5) {
    x += 1;
    printf("Hello %d\n", x);
}
```

Recap: Arrays

Modifiable containers for data.

With **variables**:

```
int num1 = 42;  
int num2 = 100;  
int num3 = 10;  
  
printf("%d\n", num1);  
printf("%d\n", num2);  
printf("%d\n", num3);
```

With a **list**:

```
int array[] = {42, 100,  
10};  
  
for(int i = 0; i < 3; i++)  
{  
    printf("%d\n",  
array[i]);  
}
```

Recap: Accessing Array Elements

To access array elements you can use the **[index]** operator.

NOTE: List indices start from **0**

index:	0	1	2	3	4
<code>int array[] = {</code>	<code>17,</code>	<code>28,</code>	<code>33,</code>	<code>56,</code>	<code>6};</code>

```
printf("%d\n", array[0]);
```

```
printf("%d\n", array[3]);
```

Recap: Assigning Array Elements

To assign array elements you can use the **[index]** operator on the left-hand-side of a statement (like a variable)

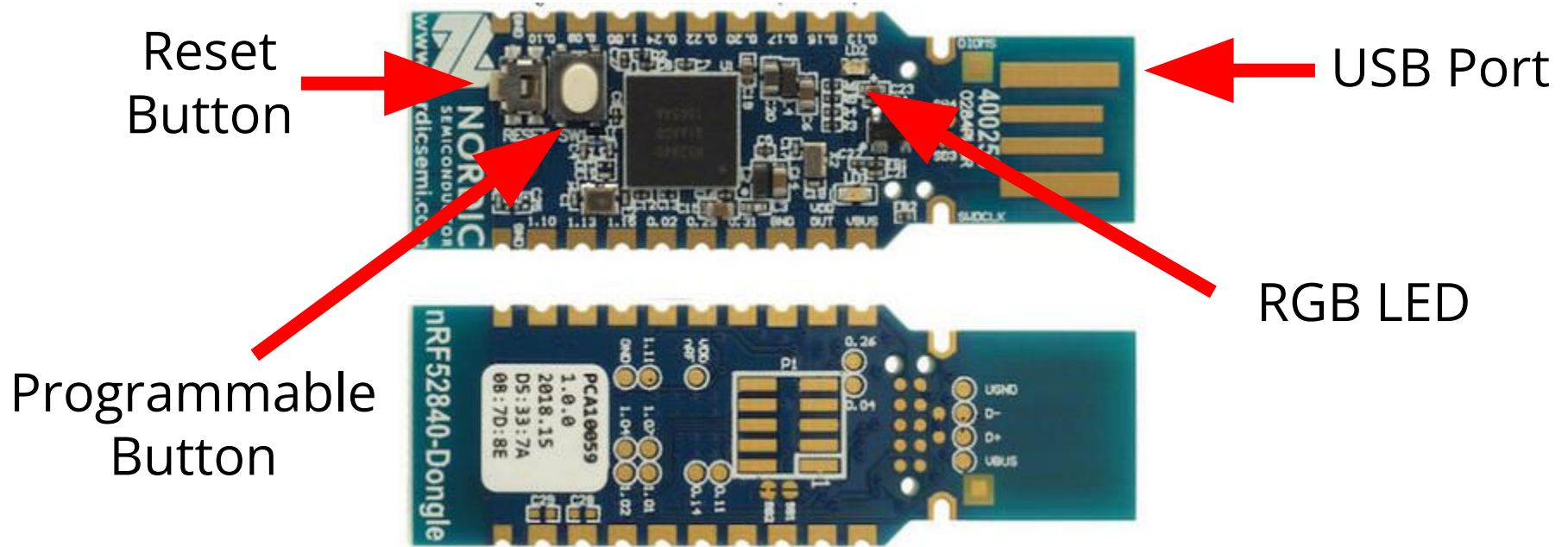
```
int array[] = {17, 28, 33, 56, 6};  
array[3] = 100;  
array[2] = -7;
```

```
printf("%d\n", array[0]);
```

```
printf("%d\n", array[3]);
```



Recap: What is the nRF52840?



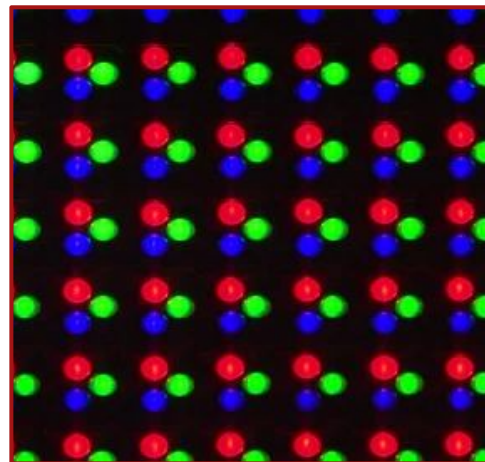
Recap: Anatomy of a Contiki-NG Program

```
1 PROCESS_THREAD(button_hal_example, ev, data) {  
2     PROCESS_BEGIN();  
3     while (1) {  
4         PROCESS_YIELD();  
  
5         if (ev == button_hal_press_event) {  
6             printf("Button pressed!\n");  
7         }  
8     }  
9     PROCESS_END();  
10 }
```

Recap: RGB LEDs

LEDs are **actuators**, they allow the device to act on the outside world. RGB LEDs have **three configurable color** channels:

1. **Red**
2. **Green**
3. **Blue**



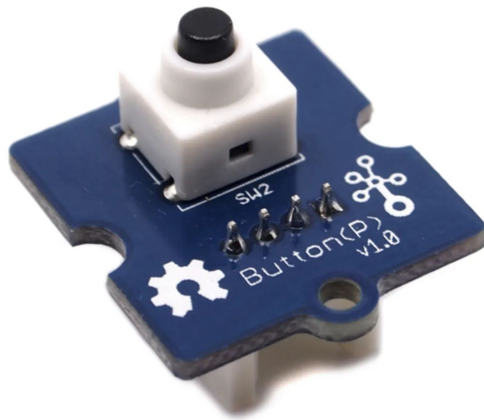
LED displays (such as those of PCs) work the same way

Recap: The LED Library

```
#define RGB_LED_RED      1
#define RGB_LED_GREEN   2
#define RGB_LED_BLUE    4
#define RGB_LED_MAGENTA (RGB_LED_RED | RGB_LED_BLUE)
#define RGB_LED_YELLOW  (RGB_LED_RED | RGB_LED_GREEN)
#define RGB_LED_CYAN    (RGB_LED_GREEN | RGB_LED_BLUE )
#define RGB_LED_WHITE   (RGB_LED_RED | RGB_LED_GREEN | RGB_LED_BLUE)
/*-----*/
void rgb_led_off(void);
void rgb_led_set(uint8_t colour);
```

Recap: Buttons

Buttons allow the device to **“sense”** the world around them.
The button allows the device to **receive input and react** to actions in the world around them.



Recap: Button Library

```
/* Event generated when a button gets pressed */
extern process_event_t button_hal_press_event;
/* Event generated when a button gets released */
extern process_event_t button_hal_release_event;
/* Event generated every second the button is kept pressed */
extern process_event_t button_hal_periodic_event;

/*-----*/
#define BUTTON_HAL_STATE_RELEASED 0
#define BUTTON_HAL_STATE_PRESSED 1
/*-----*/

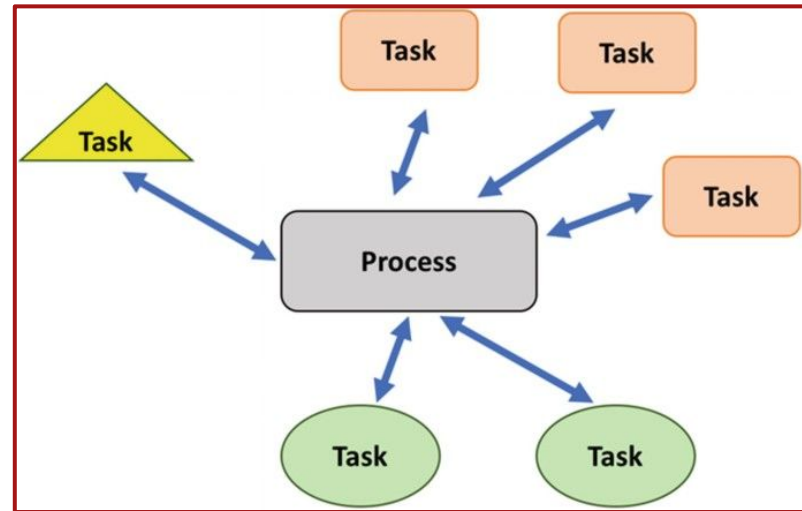
void button_hal_init(void);
uint8_t button_hal_get_state(button_hal_button_t *button);
```

Recap: Processes

Contiki-NG **Processes** allow for the execution of **multiple tasks** at the same time (i.e. **concurrently**).

We have seen:

1. **PROCESS_BEGIN**
2. **PROCESS_END**
3. **PROCESS_YIELD**



Recap: The E-Timer Library

```
/* Event generated when a timer expires */
#define PROCESS_EVENT_TIMER          0x88
/*-----*/
/* Set the amount of time on the timer. Also start the timer */
void etimer_set(struct etimer *et, clock_time_t interval);
/* Restart the timer with the previously set amount of time */
void etimer_restart(struct etimer *et);
void etimer_stop(struct etimer *et);
/*-----*/
/* Check if the timer has completed */
bool etimer_expired(struct etimer *et)
```


Recap: Programming the nRF52840

1 Attach the **nRF52840** chip to your laptops

⚠ Ensure the device is in **bootloader mode** (blinking red light)

Reset
Button



3 Program the firmware

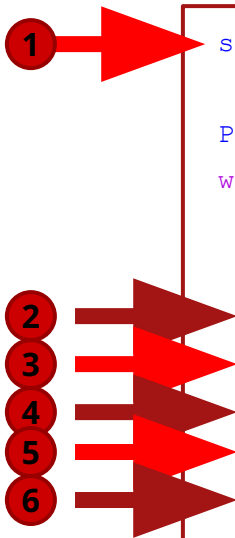
```
make simple_timer.dfu-upload
```

Recap Exercise

Change the code in
(**simple_timer.c**)

1) Pressing the button
turns the LED Green.

2) After 2 seconds the
LED is turned off



1 → static struct etimer blink_timer;

2 → PROCESS_BEGIN();

3 → while (true) {

4 → PROCESS_YIELD();

5 → if (ev == button_hal_press_event) {

6 → rgb_led_set(RGB_LED_GREEN);

etimer_set(&blink_timer, 2 * CLOCK_SECOND);

} else if (ev == PROCESS_EVENT_TIMER) {

if (etimer_expired(&blink_timer)) {

rgb_led_off();

}

}

}

To flash: make simple_timer.dfu-upload

For console: make login 

Save remotely your Changes

1

```
make save
```

2

```
|Password
```

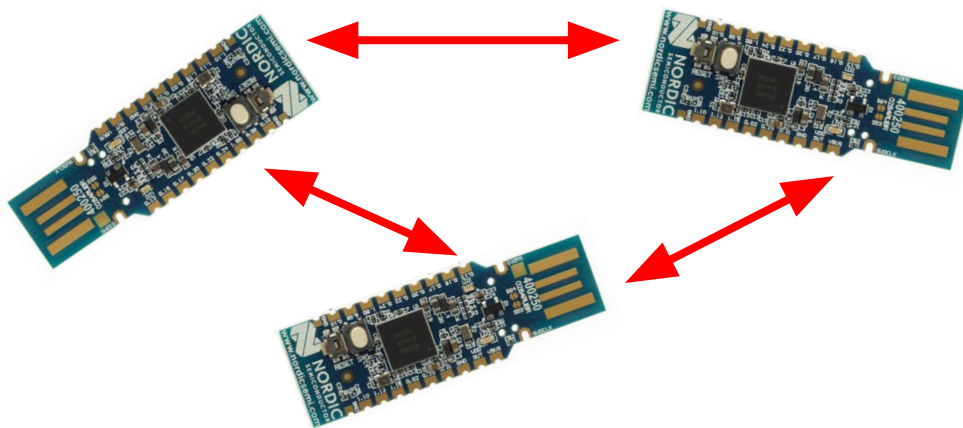
```
Git: https://aspina@git.spina.me (Press 'Enter' to confirm or  
'Escape' to cancel)
```

3

```
✓ Changes committed and pushed. All done!
```

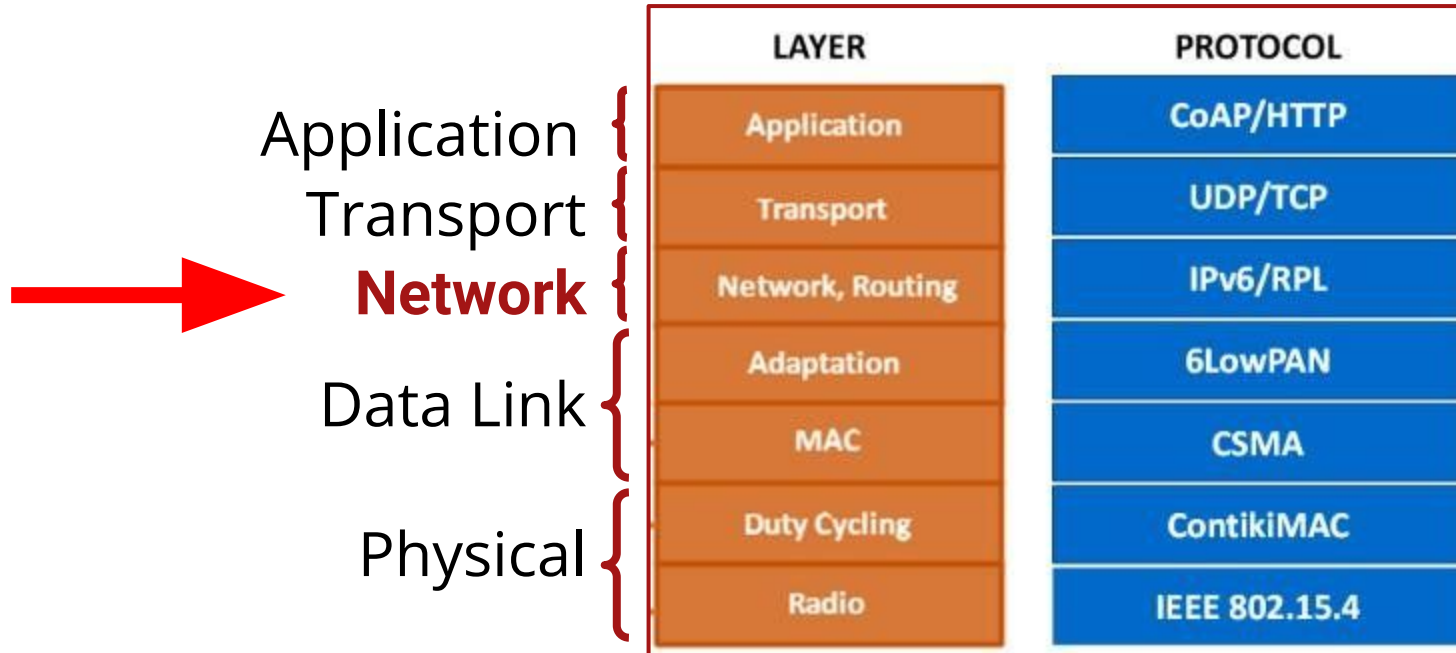
Networking

- **Networking** is the practice of connecting computers and other devices to **share information**.
- Involves **transmitting data** over various types of media, like wireless signals.



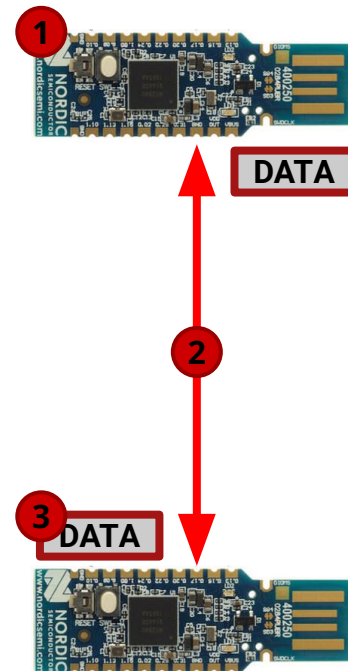
Networking

The Network layer is part of the **OSI standard**. We disable the Application and Transport layers in Contiki-NG using “**Nullnet**”.



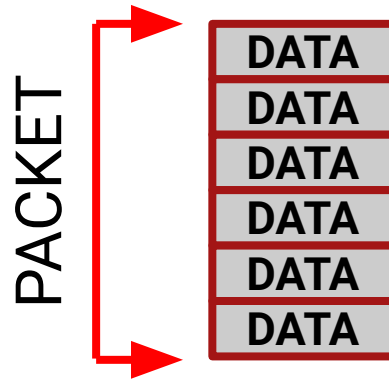
Networking Concepts

1. **Nodes:** Individual devices in a network (like computers, sensors).
2. **Links:** Connections between nodes (wired or wireless).
3. **Data Packets:** Small units of data sent over a network.



Networking Packets

- **Packets** are small chunks of **data** sent over the network.
- Packets can have **arbitrary length** (up to a maximum)



Receiving Packets

We provide a helper function to simplify **receiving packets** over nullnet:

receive_nullnet_data

You can add functionality **inside the function body**.

```
/* Helper function to receive data over nullnet */
void receive_nullnet_data (
    const void *bytes,
    uint16_t len,
    const linkaddr_t *src,
    const linkaddr_t *dest)
{
    int data;
    memcpy(&data, bytes, len);

    printf("Data received: %d\n", data);
}
```


Exercise

To flash: `make receiver.dfu-upload`

For console: `make login`

Change the code in
(receiver.c)

1) Set your LED to the color that you receive from the network

2) Use the **blink_timer** to turn off the LED after one second of receiving the network data

HINT: Restart the timer in the **receive_nullnet_data** function

```
void receive_nullnet_data (...) {
    int data;
    memcpy(&data, bytes, len);
    printf("Color received: %d\n", data);

    // TODO (1): Set the RGB to the receiver color!

    // TODO (2a): Reset the timer to blink!
}

PROCESS_THREAD(receiver_process, ev, data) {
    PROCESS_BEGIN();
    while (true) {
        PROCESS_YIELD();

        // TODO (2b): Turn LEDs off
    }
    PROCESS_END();
}
```

Exercise Solution

```
void receive_nullnet_data (...)
{
    int data;
    memcpy(&data, bytes, len);

    printf("Color received: %d\n",
data);
    rgb_led_set(data);

    etimer_reset(&blink_timer);
}
```

```
while (true) {
    PROCESS_YIELD();

    if (etimer_expired(&blink_timer)) {
        etimer_reset(&blink_timer);

        rgb_led_off();
    }
}
```

Save remotely your Changes

1

```
make save
```

2

```
|Password
```

```
Git: https://aspina@git.spina.me (Press 'Enter' to confirm or  
'Escape' to cancel)
```

3

```
✓ Changes committed and pushed. All done!
```

Sending Packets

We provide a helper function to simplify **sending packets** over nullnet:

send_nullnet_data

You can **call** this function but you **should NOT edit** it.

```
/* Helper function to send data over nullnet */  
void send_nullnet_data (int data) {  
    printf("Sending data: %d\n", data);  
    nullnet_buf = (uint8_t *)&data;  
    nullnet_len = sizeof(data);  
  
    NETSTACK_NETWORK.output(NULL);  
}
```

```
send_nullnet_data(200);  
  
variable = 42;  
send_nullnet_data(variable);
```

Exercise

To flash: `make ping.dfu-upload`
For console: `make login`

Change the code in (**ping.c**)

1) Set your LED to **GREEN** when you receive a message

2) Use **send_nullnet_data** to respond to network messages doubling the data value

3) Use the **blink_timer** to turn off the LED after one second of receiving the network data

```
1 // TODO (1): Turn LED GREEN on message reception
2 // TODO (2): Use `send_nullnet_data` to reply
3 // TODO (3a): Reset the timer to blink!

void receive_nullnet_data (...) {
    int data;
    memcpy(&data, bytes, len);
    printf("Data received: %d\n", data);

    // TODO (1): Turn LED GREEN on message reception
    // TODO (2): Use `send_nullnet_data` to reply
    // TODO (3a): Reset the timer to blink!
}

PROCESS_THREAD (ping_process, ev, data) {
    PROCESS_BEGIN ();
    while (true) {
        PROCESS_YIELD ();

        // TODO (3b): Turn LEDs off
    }
    PROCESS_END ();
}
```

Exercise Solution

```
void receive_nullnet_data (...)
{
    int data;
    memcpy(&data, bytes, len);

    printf("Data received: %d\n", data);

    rgb_led_set (RGB_LED_GREEN);
    send_nullnet_data (data * 2);

    etimer_reset (&blink_timer);
}
```

```
while (true) {
    PROCESS_YIELD ();

    if (etimer_expired (&blink_timer)) {
        etimer_reset (&blink_timer);

        rgb_led_off ();
    }
}
```

Save remotely your Changes

1

```
make save
```

2

```
|Password
```

```
Git: https://aspina@git.spina.me (Press 'Enter' to confirm or  
'Escape' to cancel)
```

3

```
✓ Changes committed and pushed. All done!
```

Quiz Time!

ahaslides.com/DN2A2

End of Class

See you all next week!