MÁSTER UNIVERSITARIO EN SISTEMAS ESPACIALES
POR LA UNIVERSIDAD POLITÉCNICA DE MADRID

POLITÉCNICA

2020.05.05

On-board software laboratory

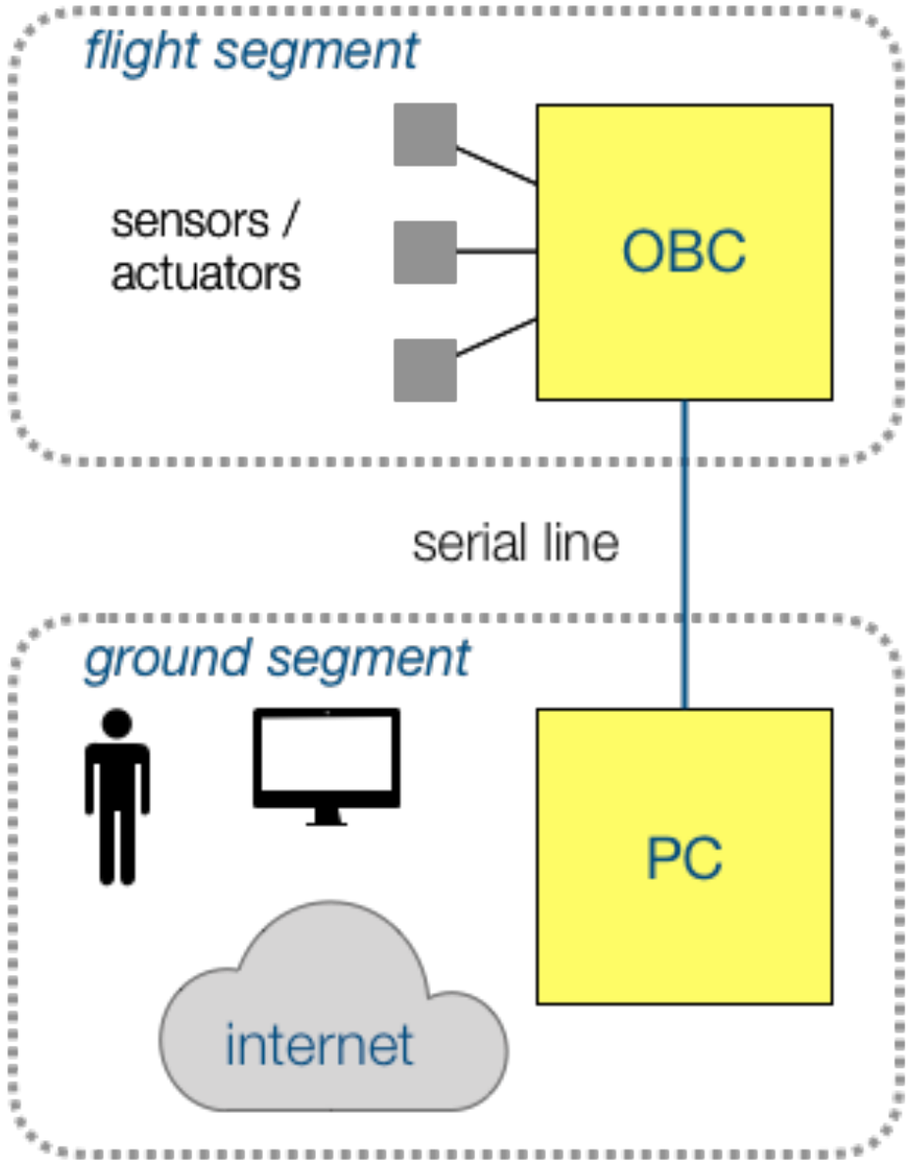
Juan Antonio de la Puente <juan.de.la.puente@upm.es>

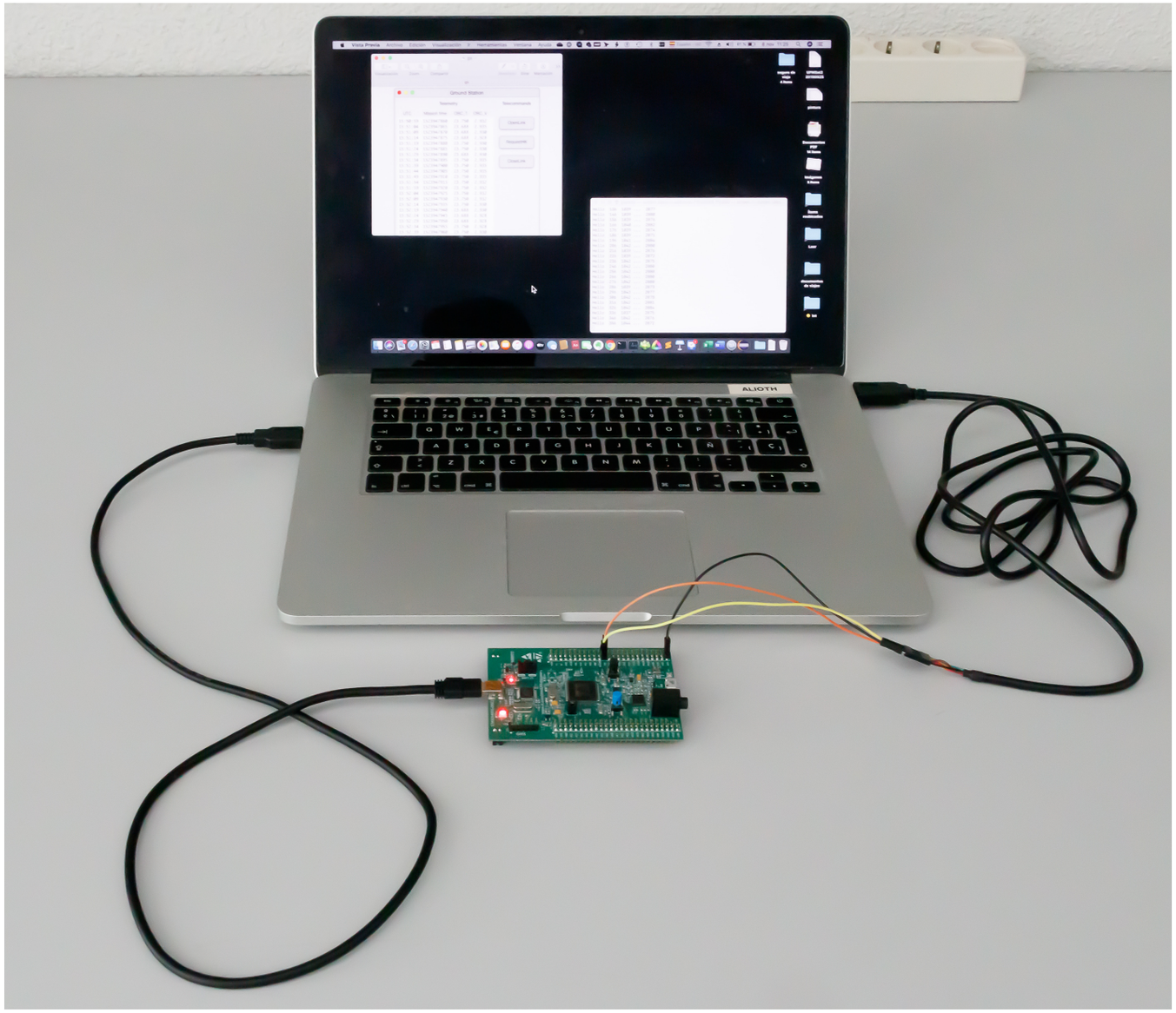
STRAST

Laboratory assignments

1. Installation of a native programming environment.
2. Installation of the cross-platform programming tools.
3. Simple housekeeping program.
4. Tasking program.
5. Distributed program.
6. Real-time program, including temporal analysis

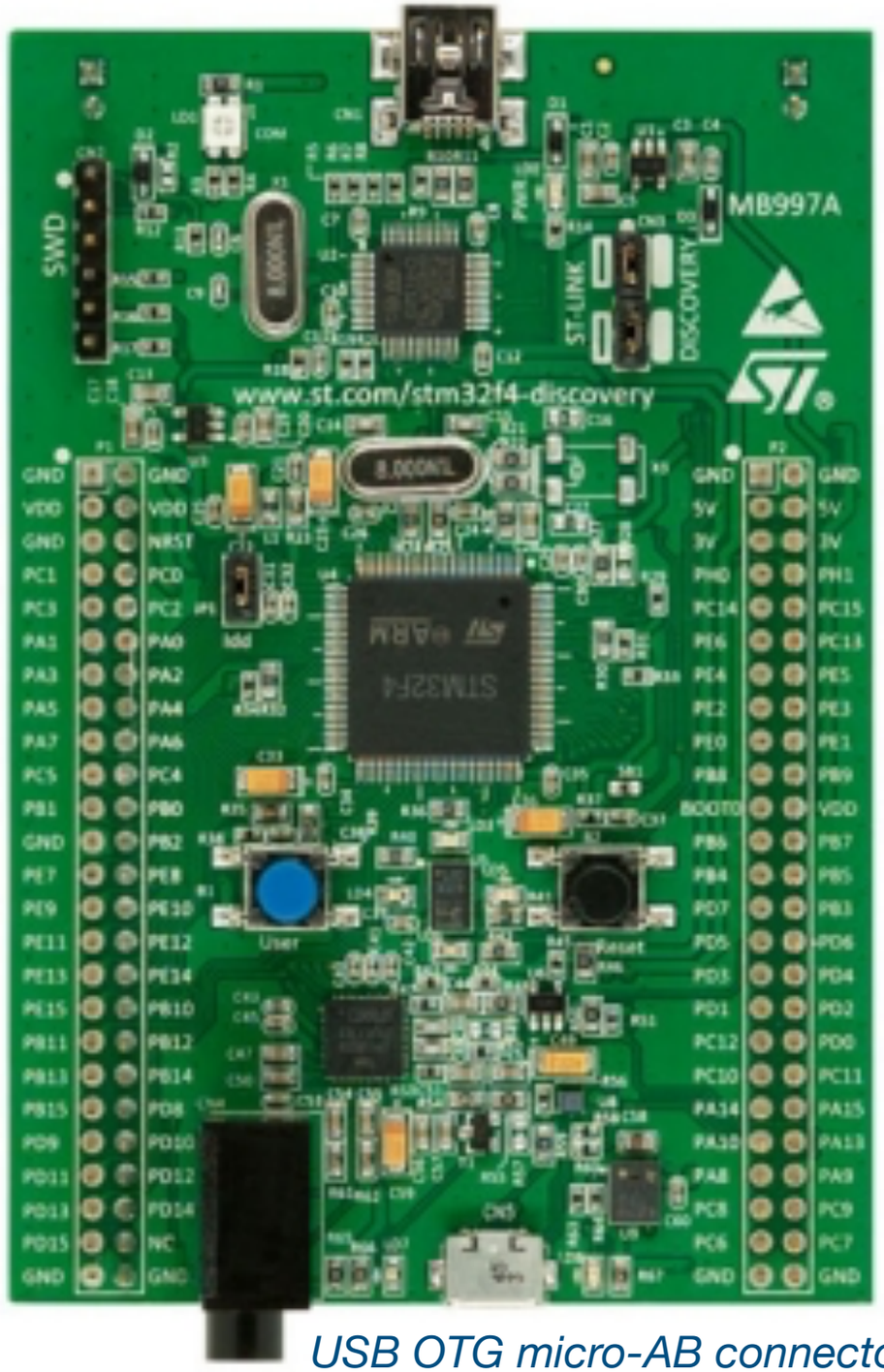
Laboratory kit





Computer board

USB ST-LINK mini-B connector



PB6 & PB7 pins

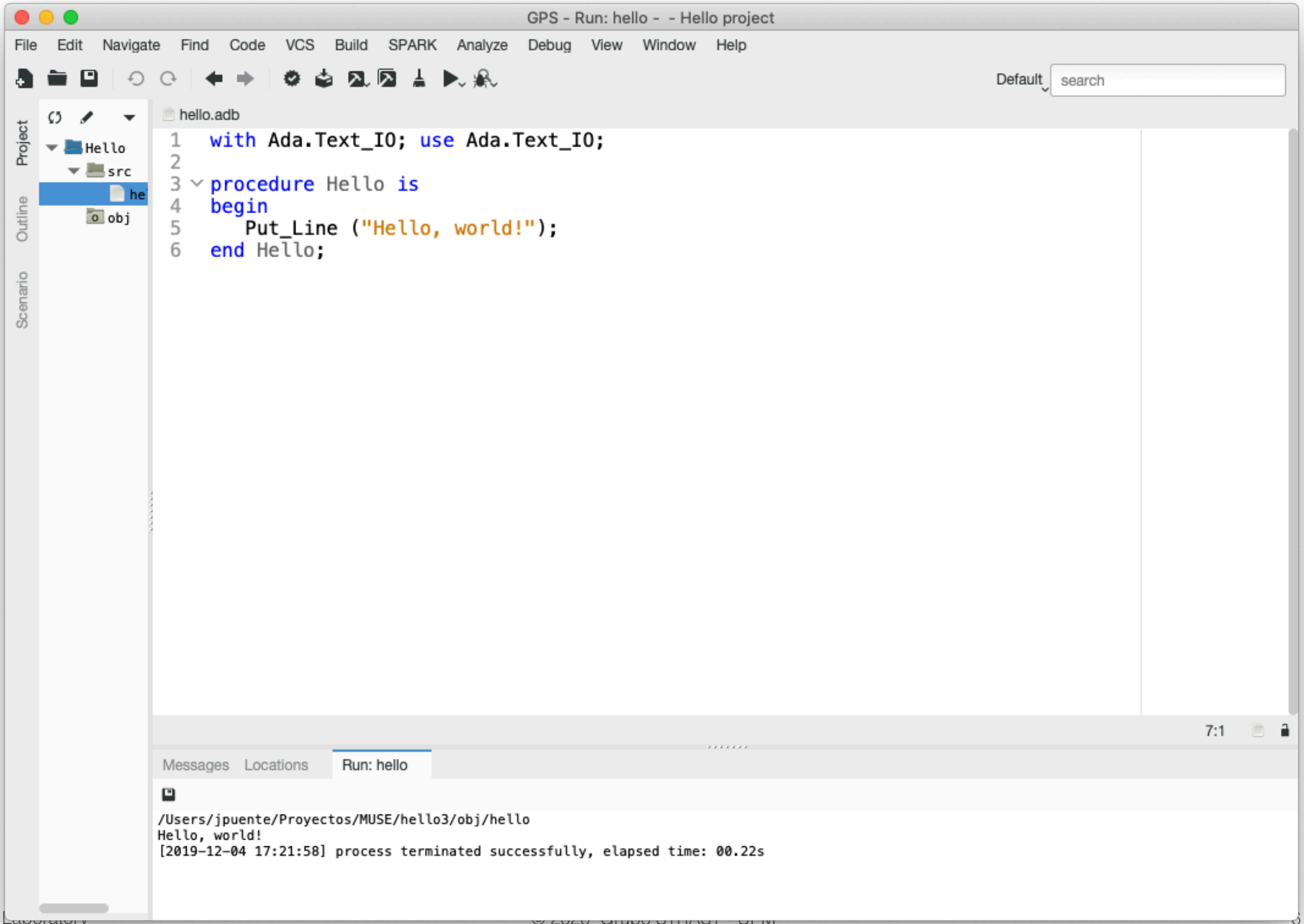
GND pin

USB OTG micro-AB connector

1. Install the native compiler

- Windows:
  - Download and run `gnat-community-2019-20190517-x86_64-windows-bin.exe`
  - Run the GPS application
  - Create a new project and enter the source code for hello.adb
  - Build and run the executable
- MacOS and Linux
  - See laboratory guide

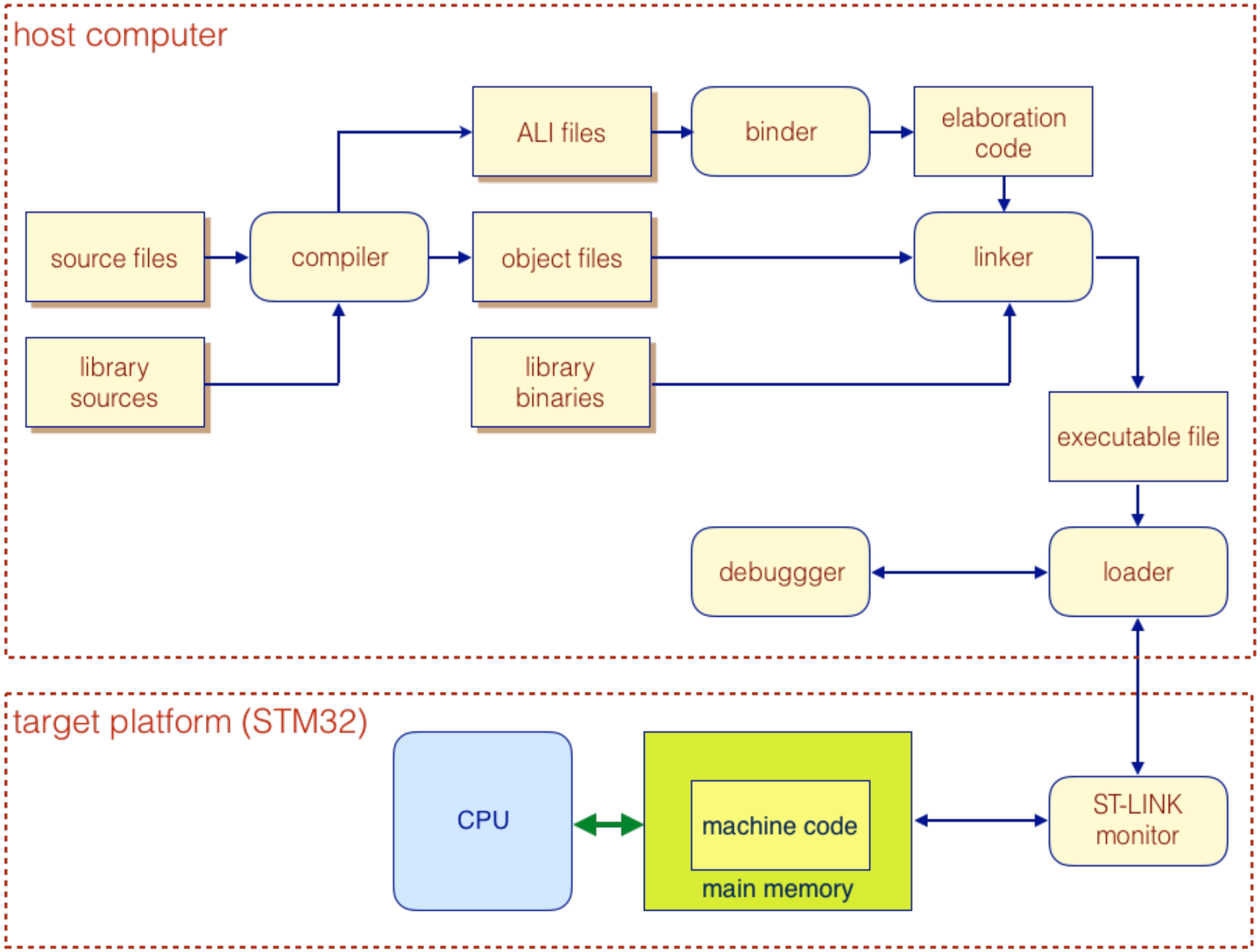
Sample screenshot



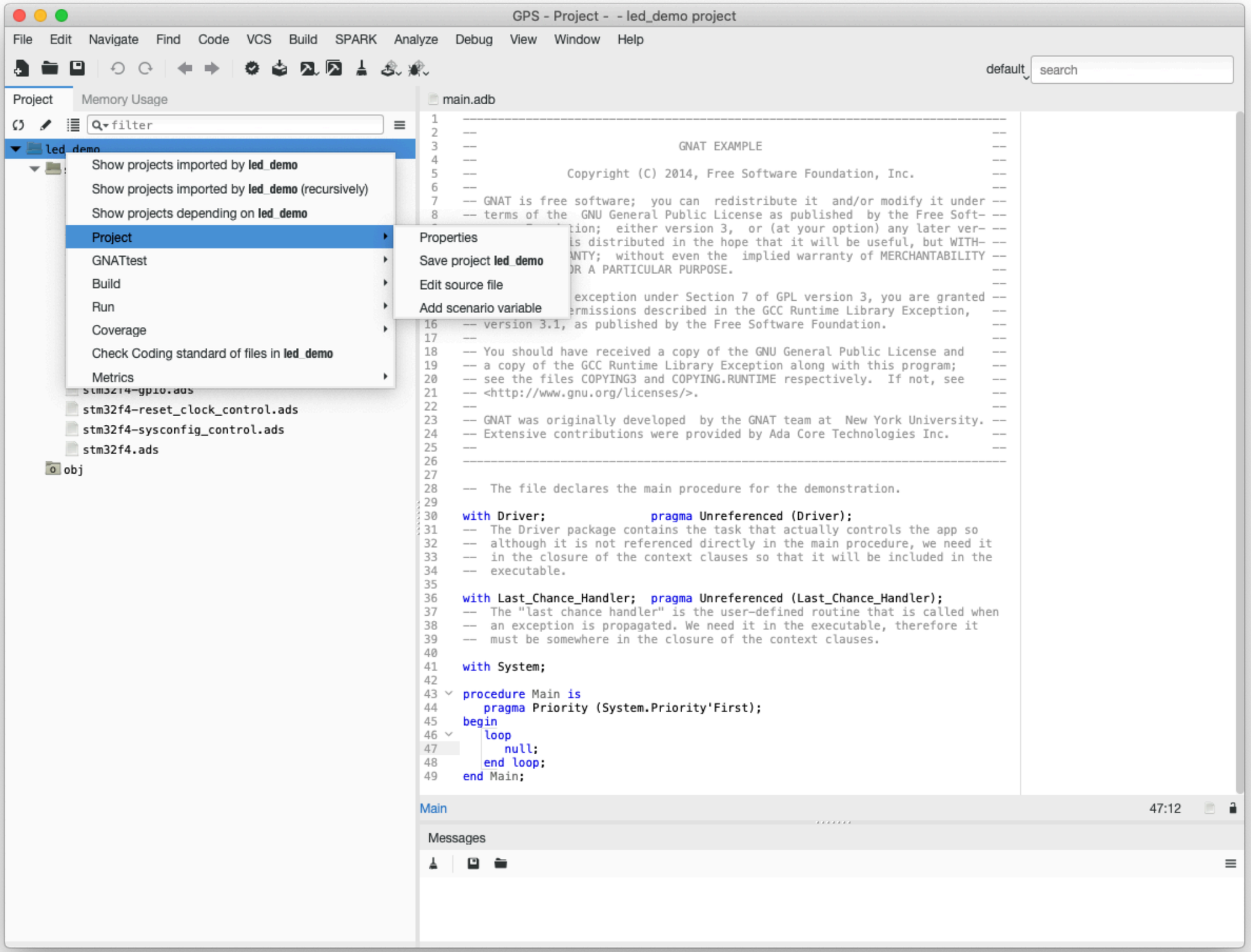
2a. Install cross compiler for STM32F4

- Windows:
  - Download and execute the cross-compiler
    `gnat-community-2019-20190517-arm-elf-windows64-bin.exe`
  - Download and execute the ST-LINK driver
    - follow the instructions in `readme.txt`
  - Start the GPS application
  - Create a new project
    - `File > New Project > STM324F compatible > LED demo project template`
    - Set `Project > Properties > Embedded > Connection tool` to `st-util`
  - Build and run the executable
  - Connect the board to the host computer and flash the executable to the board

Cross-compilation tools



Sample screenshot



2b. Install the Ada Drivers Library

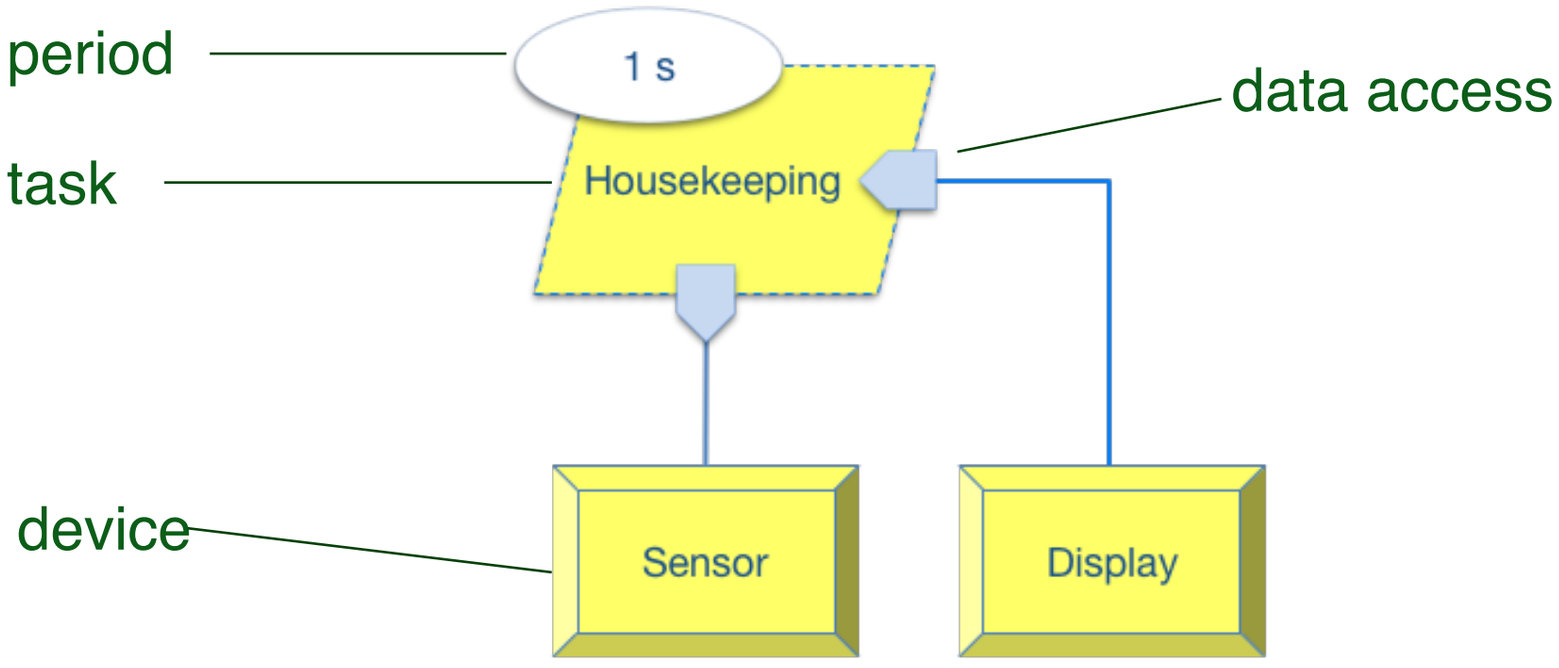
- All platforms:
  - Download zip file from `https://github.com/AdaCore/Ada_Drivers_Library`
  - Unzip the archive and move the resulting folder to your laboratory folder
  - Rename the folder to `Ada_Drivers_Library`
  - Open project `…/Ada_Drivers_Library/examples/STMF4_DISCO/blinky_f4disco.gpr`.
  - Build the executable and flash it to the board

3. Simple housekeeping system

- Read a single sensor (OBC_T) and show the reading on the screen
  - the OBC board has no screen
    - use the debugger (for now) - *semi-hosting*
  - use the Ada Drivers Library (ADL) to read the sensor
    - STM32.Device
    - STM32.ADC

Software architecture

AADL : Architecture Analysis & Design Language



AADL components implemented as Ada packages

- main procedure: OBDH
- additional packages for data definitions & others as necessary

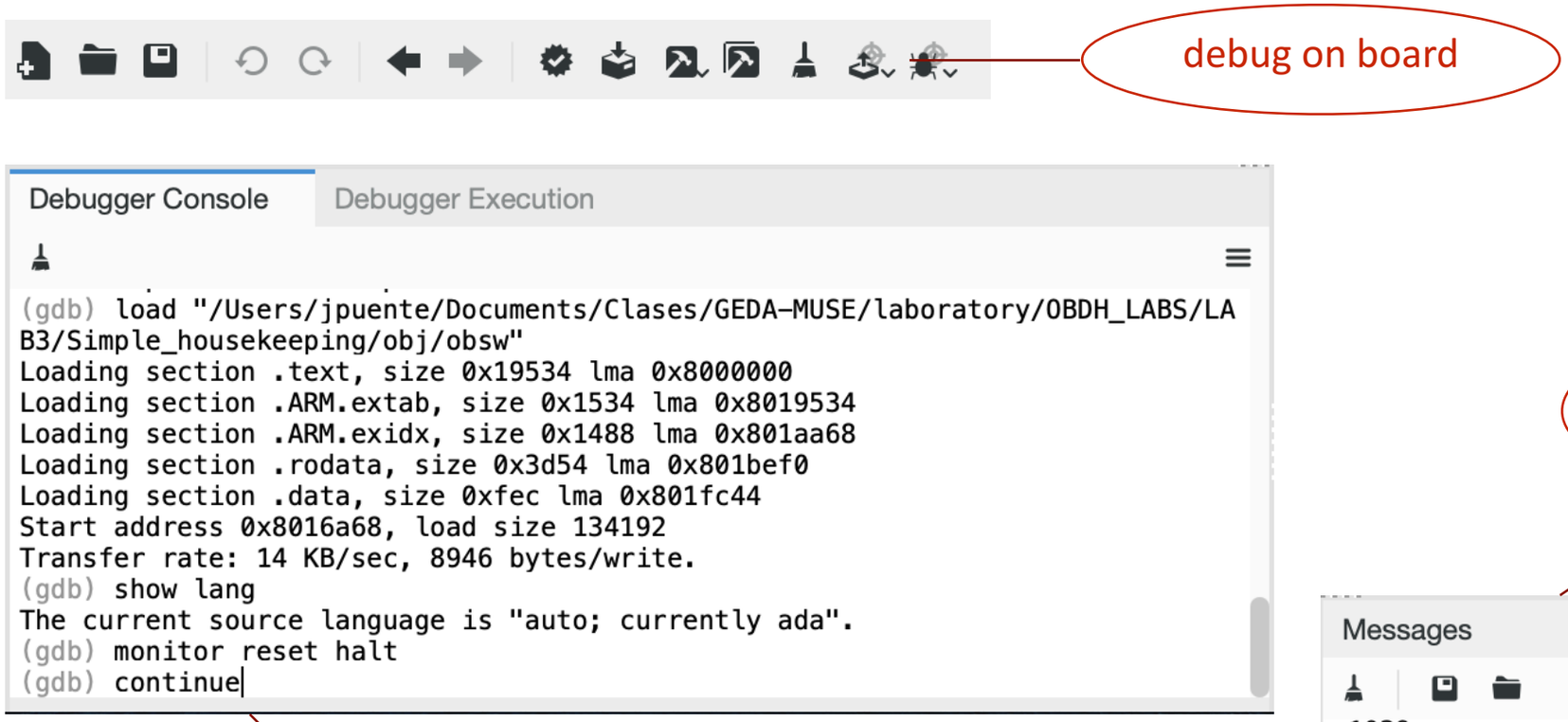
Temperature sensor

- OBC_T : MCU internal temperature sensor
  - connected to ADC1_IN16 input
  - range -40 .. +125 ºC
  - 14-bit reading VSENSE: 0 .. 4095
  - Temperature (in °C) = {(VSENSE – V25) / Avg_Slope} + 25

Symbol	Parameter	Min	Typ	Max	Unit
T_L(1)	V_{SENSE} linearity with temperature	-	±1	±2	°C
Avg_Slope(1)	Average slope	-	2.5		mV/°C
V_{25}(1)	Voltage at 25 °C	-	0.76		V
t_{START}(2)	Startup time	-	6	10	µs
T_{S_temp}(2)	ADC sampling time when reading the temperature (1 °C accuracy)	10	-	-	µs

Display

- Emulate a display with *semi-hosting*
  - run with the debugger
  - `Ada.Text_IO` output is redirected to the debugger



Study the code and make changes

- All platforms:
  - Download zip file from moodle or `https://github.com/STR-UPM/OBDH_LABS`
  - Unzip the archive and move the resulting folder to your laboratory folder
  - Open project `.../LAB3/Simple_housekeeping/simple_housekeeping.gpr`
  - Build the executable and debug on the board

Additional tasks

- Include the conversion to Celsius in the `Display.Put` procedure.
- Add the following statement to the main loop in the `Housekeeping Run` procedure:

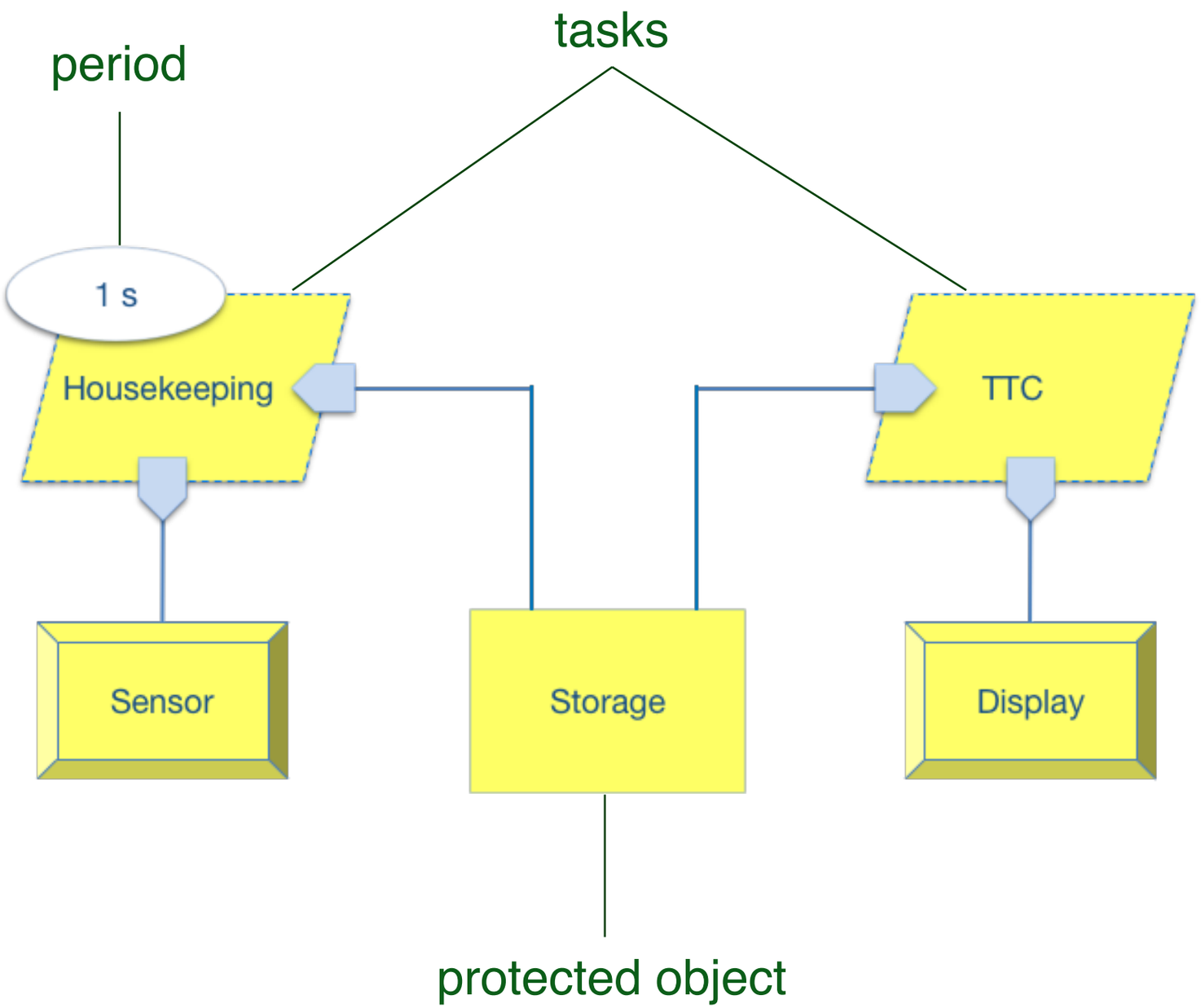
```
delay until Clock + Milliseconds (1000);
```

(see the code in the Blinky_LEDS program)

4. Tasking housekeeping system

- Add two tasks for better structuring
  - Housekeeping
    - read a sensor (OBC_T)
  - TTC
    - display the sensor value
- Add a protected object for inter-task communication
  - Storage
    - the housekeeping task puts the sensor value in the storage
    - the TTC task gets the value from the storage

Software architecture



Study the code and make changes

- All platforms:
  - Download zip file from moodle or `https://github.com/STR-UPM/OBDH_LABS`
  - Unzip the archive and move the resulting folder to your laboratory folder
  - Open project `.../LAB4/Tasking_housekeeping/tasking_housekeeping.gpr`
  - Build the executable and debug on the board

Additional tasks

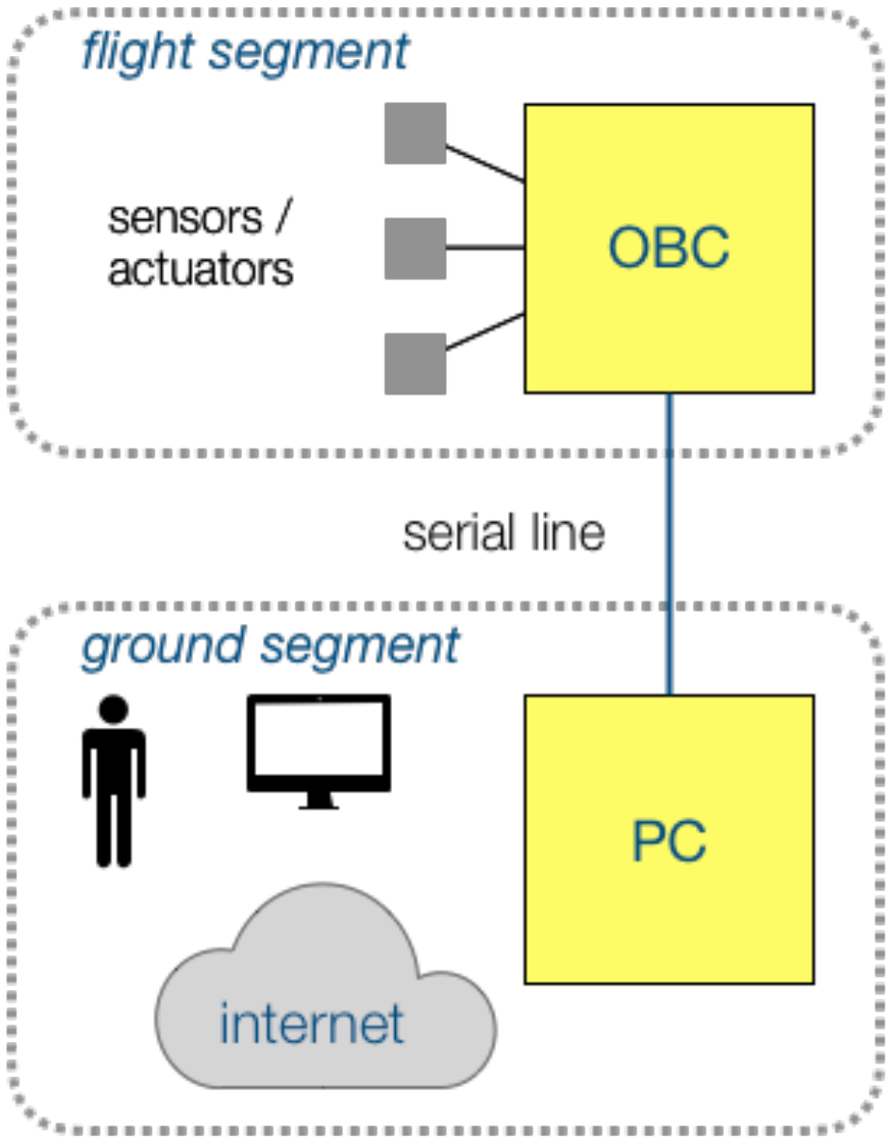
- Include the conversion to Celsius in the `Display.Put` procedure.
- Add the following statement to the main loop in the `Housekeeping Run` procedure:

```
delay until Clock + Milliseconds (1000);
```

(see the code in the Blinky_LEDS program)

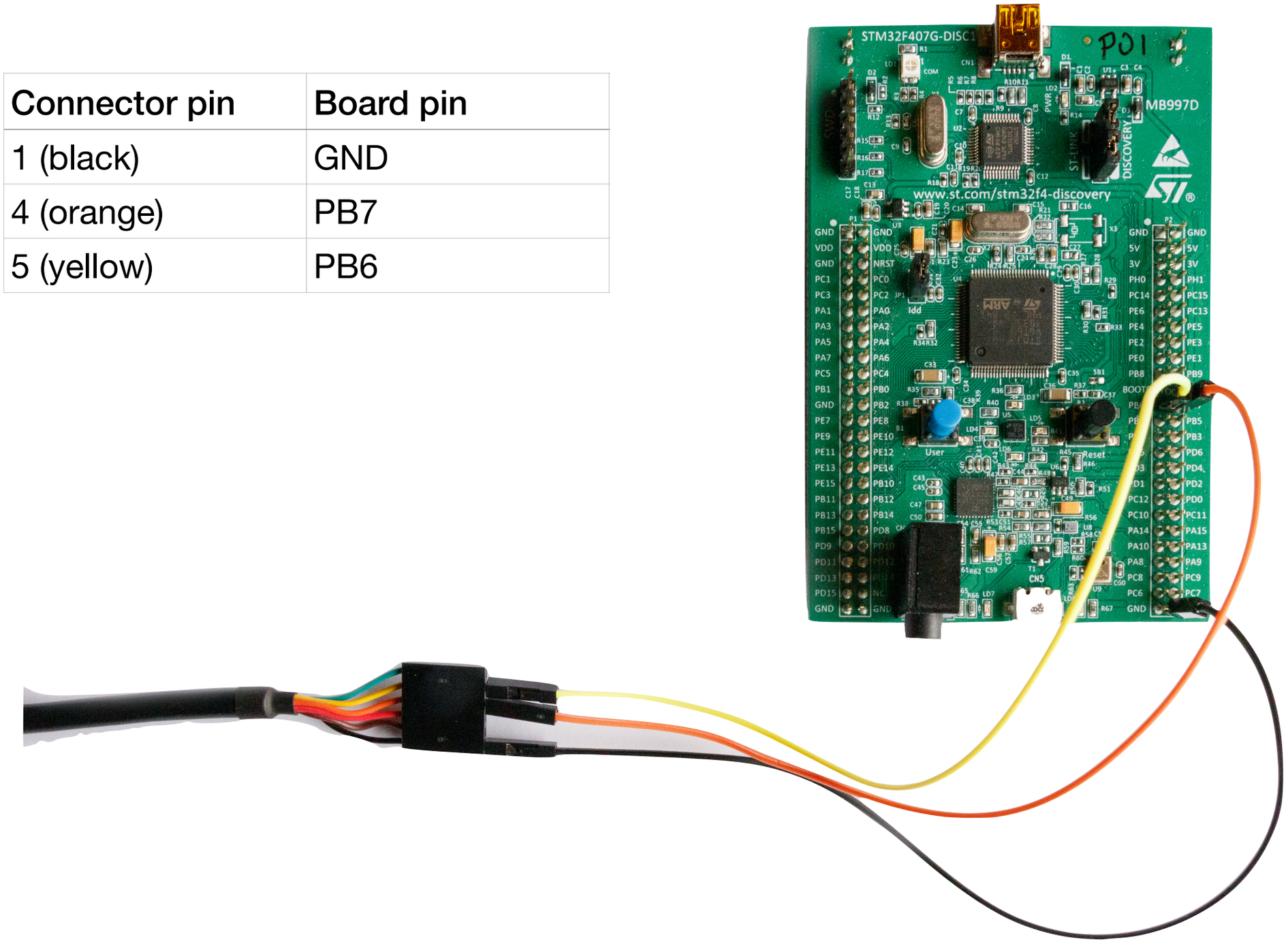
5. Distributed housekeeping system

- Add a serial interface connected to the host PC
  - simulate a radio link between flight and ground systems



Hardware

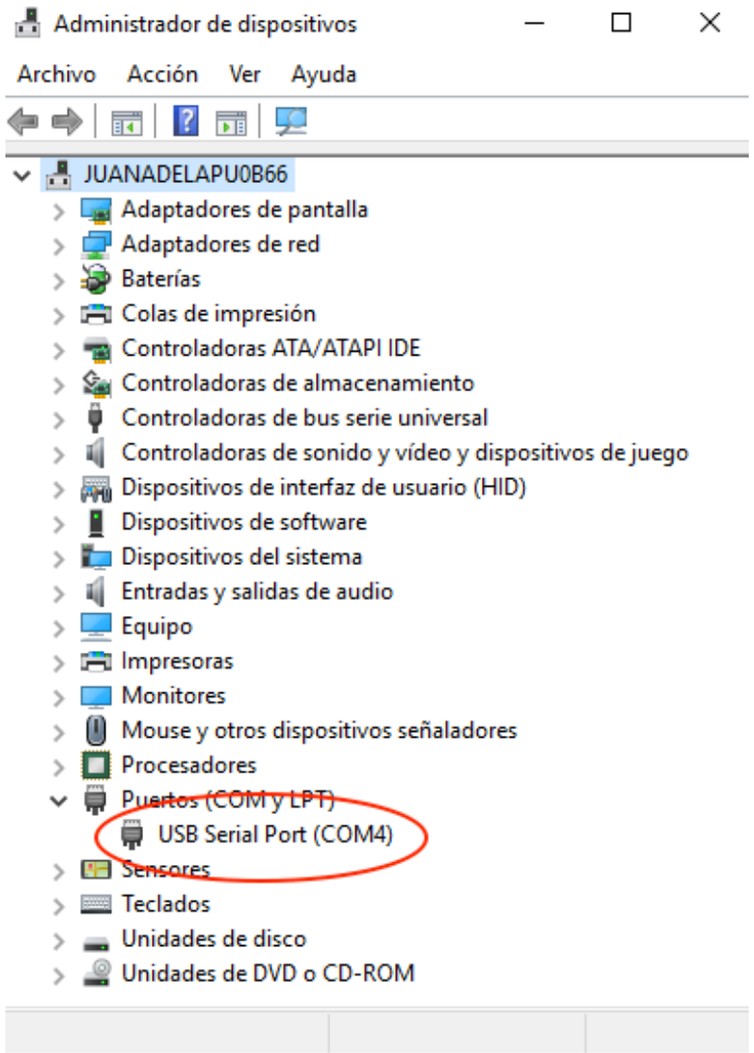
Connector pin	Board pin
1 (black)	GND
4 (orange)	PB7
5 (yellow)	PB6

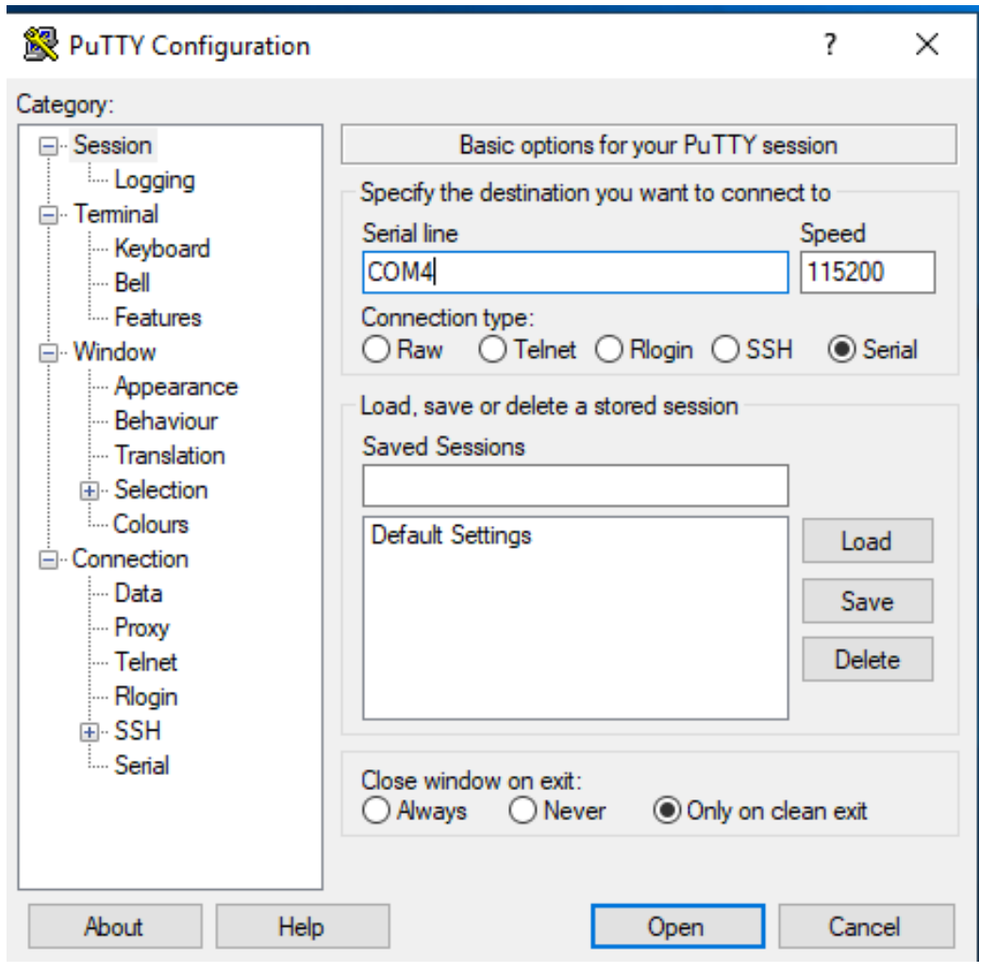


Host terminal

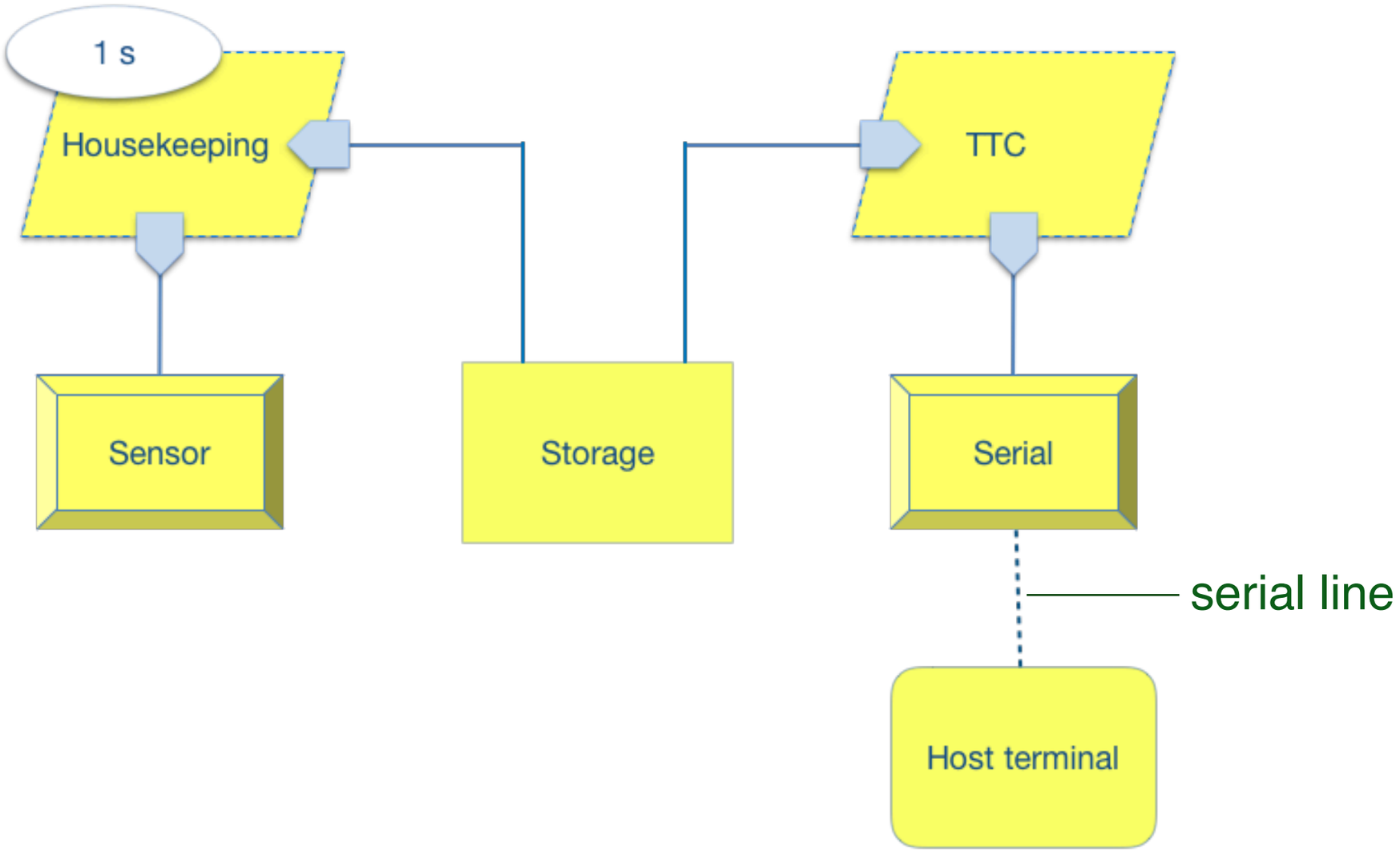
(Windows — for other platforms see manual)

- Download and install https://www.putty.org
- Find the COM port in device manager (e.g. COM4)
- Open PutTTY and configure as serial connection,115200 bps, 8N1





Software architecture

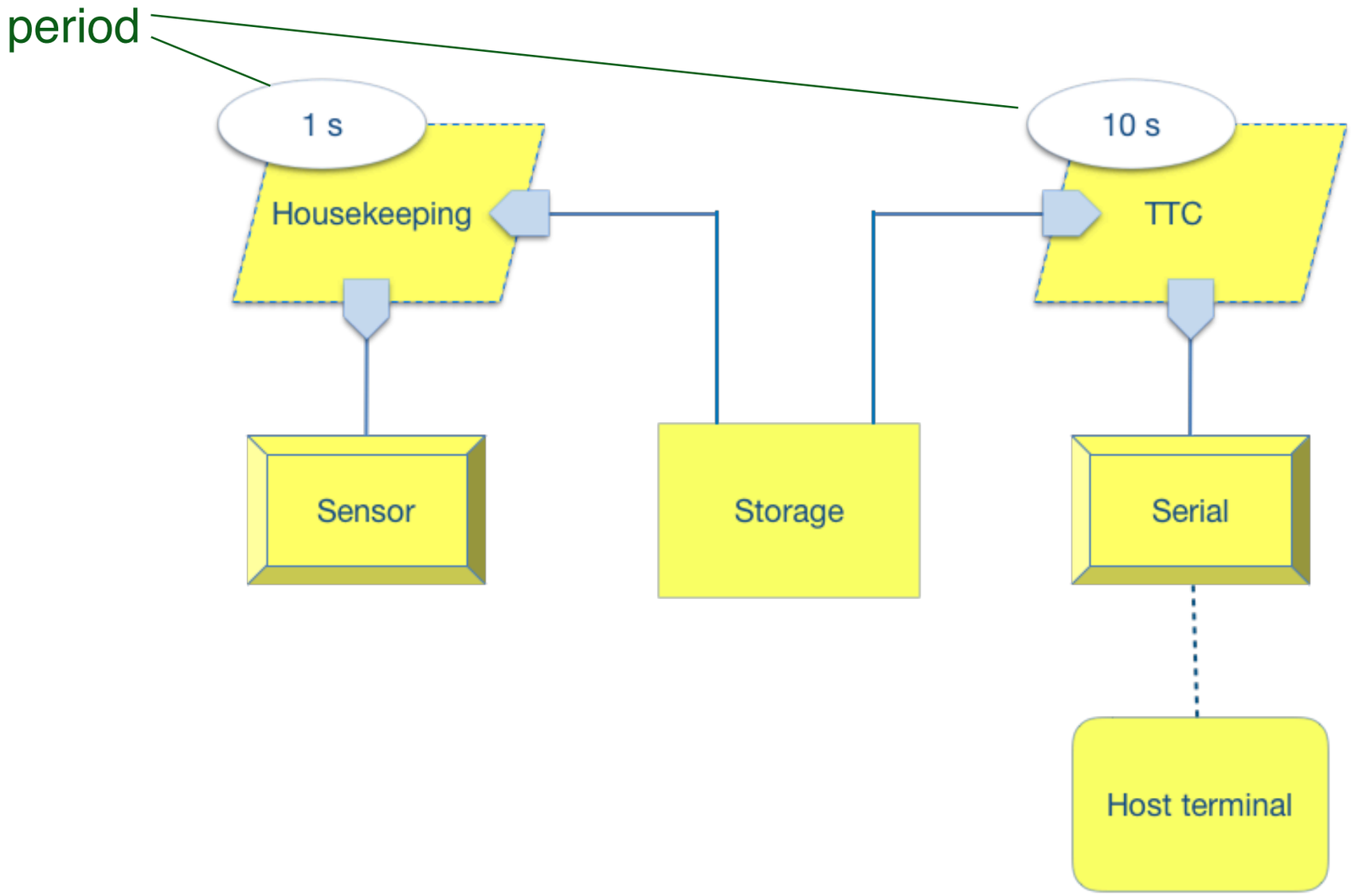


Study the code and make changes

- All platforms:
  - Download zip file from moodle or `https://github.com/STR-UPM/OBDH_LABS`
  - Unzip the archive and move the resulting folder to your laboratory folder
  - Open project `.../LAB5/Distributed_housekeeping/distributed_housekeeping.gpr`
  - Build the executable and flash to board

6. Real-time housekeeping system

- Add real-time requirements and priorities to tasks
- Add timestamps to measurements



Real-time requirements

Task	Period	Deadline	Priority
Housekeeping	1,0	1,0	20
TTC	10,0	2,0	10
Storage Buffer	—	—	20

```
package Housekeeping is
   Period      : Time_Span := Milliseconds (1000);  -- 1s
   HK_Priority : constant System.Priority := 20;
   ...
private
   Start_Delay : Time_Span := Milliseconds (1000); -- 1s
   task Housekeeping_Task
     with Priority => HK_Priority;
end Housekeeping;
```

Implementation

```
task body Housekeeping_Task is
   Data      : State;
   Next_Time : Time := Clock + Start_Delay;
begin
   loop
      delay until Next_Time;
      ...
      Next_Time := Next_Time + Period;
   end loop;
end Housekeeping_Task;
```

Timestamps

```
package Housekeeping_Data is
   ...
   type Mission_Time is new Uint64;
   --  Seconds since system startup

   type State is
      record
         Data      : Analog_Data;
         Timestamp : Mission_Time;
      end record;

end Housekeeping_Data;
```

Housekeeping

```
procedure Get (S : out State) is
    OBC_T : Analog_Data;
    SC    : Seconds_Count;
    TS    : Time_Span;
 begin
    Sensor.Get (OBC_T);
    Split (Clock, SC, TS);
    S.Data := OBC_T;
    S.Timestamp := Mission_Time (SC);
end Get;
```

Storage

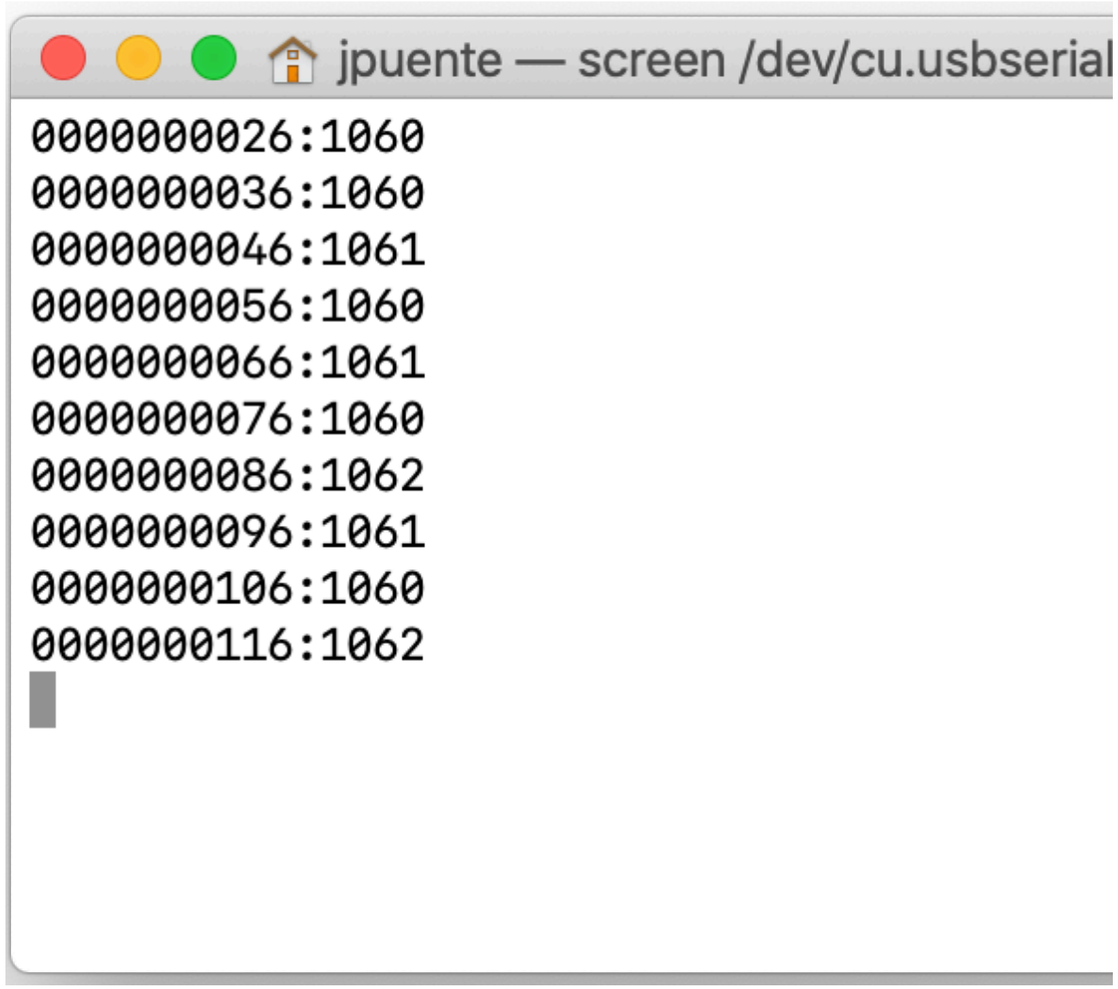
```
package Storage is

   procedure Put (Data : State);
   procedure Get (Data : out State);
   ...
   -- Real-time attributes
   ST_Priority : constant System.Priority := 20;

private

   protected Buffer
     with Priority => ST_Priority
   is ...
...
end Storage;
```

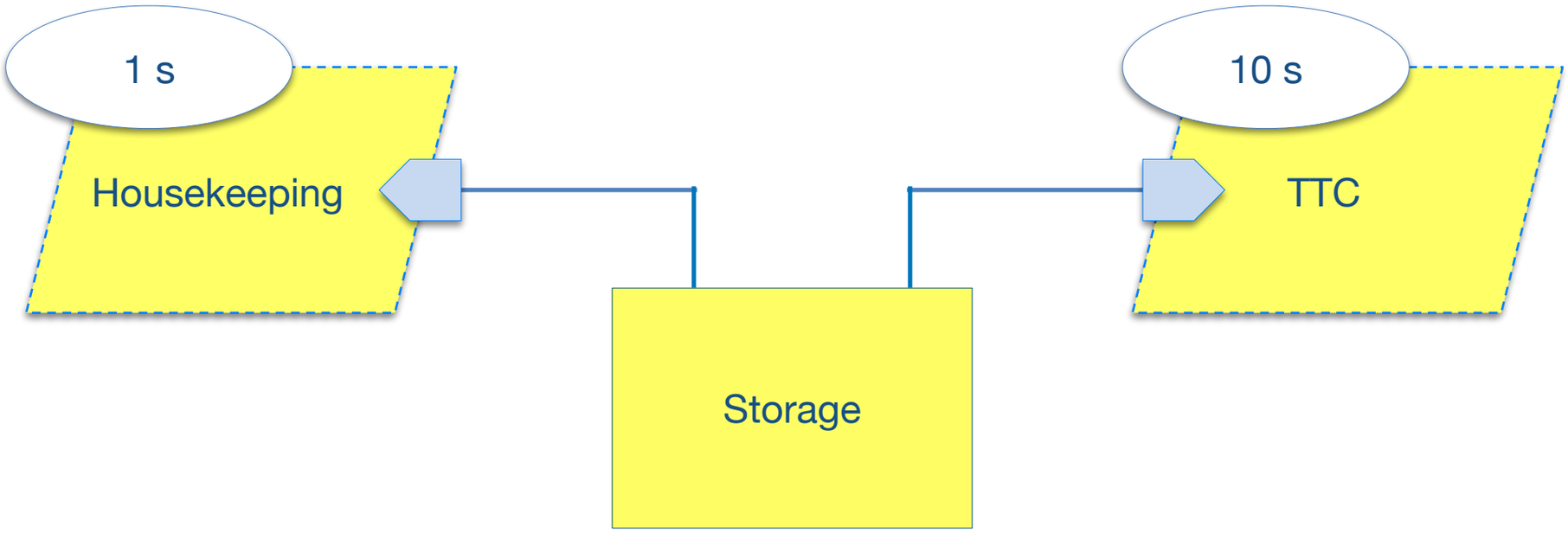
Laboratory work



- Open project
  `.../LAB6/RT_housekeeping/real_time_housekeeping.gpr`
- Build the executable and flash to board
- Make sure the TTY cable is connected
- Telemetry messages appear on the ground terminal

Real-time analysis

- Task structure



- Real-time attributes

Task	T	C	B	D	R	P	C_{Put}	C_{Get}
HK	1,0			1,0		20		
TTC	10,0			2,0		10		
						CP	20	20

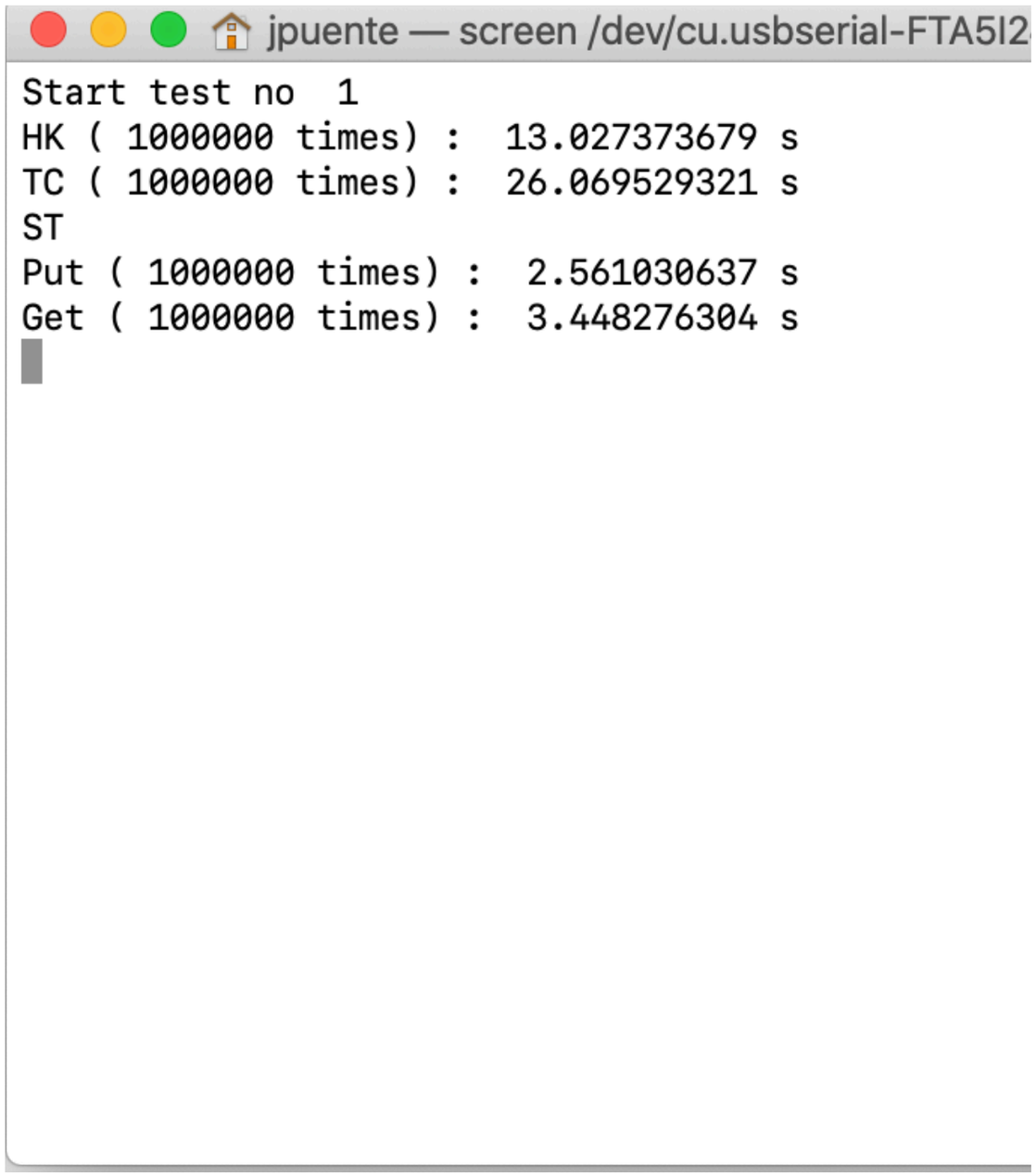
WCET measurement

- A simple dynamic measurement technique will be used in the laboratory
- Execute task bodies a number of times and measure times with the board real-time clock
  - e.g. for HK task

```
T1 := Clock;
for I in 1..N loop
   Get(Data);
   Storage.Put (Data);
   T := T + Period;
end loop;
T2 := Clock;
C := T2 - T1;
```

  - the estimated WCET in seconds is C/N

Laboratory work



- Open project `.../LAB6/RT_housekeeping/wcet_meter.gpr`
- Study the source code
- Build the executable and flash to board
- Make sure the TTY cable is connected
- Analysis results appear on the ground terminal

Real-time analysis with measured values

Task	T	C	B	D	R	P	C_{Put}	C_{Get}
HK	1,0	$13 \cdot 10^{-6}$	$4 \cdot 10^{-6}$	1,0		20	$3 \cdot 10^{-6}$	—
TTC	10,0	$26 \cdot 10^{-6}$	—	2,0		10	—	$4 \cdot 10^{-6}$
						CP	20	20

$$R_i = C_i + B_i + \sum_{j \in \mathrm{hp}(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Task	T	C	B	D	R	P	C_{Put}	C_{Get}
HK	1,0	$13 \cdot 10^{-6}$	$4 \cdot 10^{-6}$	1,0	$17 \cdot 10^{-6}$	20	$3 \cdot 10^{-6}$	—
TTC	10,0	$26 \cdot 10^{-6}$	—	2,0	$39 \cdot 10^{-6}$	10	—	$4 \cdot 10^{-6}$
						CP	20	20

All deadlines are guaranteed