ON-BOARD SPACECRAFT SOFTWARE

PROJECT 1 (Part B)
Arduino Module

Tutor: Javier Fernández Muñoz

(jfmunoz@inf.uc3m.es)

Description

The objective is to build a microcontroller subsystem that has a direct access to the sensors and actuators using Arduino. This subsystem is part of a prototype for reduced onboard software for a satellite, together with a main computing hardware system that controls the main tasks. Both subsystems communicate using a master/slave message protocol defined for this project.

1 Characteristics of the microcontroller subsystem:

The microcontroller is in charge of handling the hardware sensors and actuators while executing the commands received from the main system. Some of the tasks were developed in part A of this project. Specifically, these tasks are the following:

- Get the average temperature of the satellite. (This task is going be simulated instead of using a real sensor).
- Get the current position of the satellite on the orbit. (This task is going be simulated instead of using a real sensor).
- Receive commands from the main system, execute them and send back the response.

Also there are new tasks that should be implemented this subsystem, such as:

- Get the status of the sunlight onto the satellite. (This task is going to use a light sensor).
- Set the satellite heater on and off (This task is going to use a led as the heater and will be done on the next part of the project).

The hardware for this subsystem will include:

- One Arduino microcontroller
- One Light resistor (as the sunlight sensor).
- One Led (as the heater activator)

The connection between both subsystems is done using a USB-UART serial connection. Specifically connecting the main computer whith the Arduino.

The source code required for this microcontroller subsystem should be contained in one source file:

- ***arduino_code.ino***: This is the source file where each of the former tasks should be implemented with one function. Also the state of the systems is stored using global variables.

Apart from the source code file, two extra files are needed to act as temporal main computer software in order to perform the test. The first will be a minimal main computer software file to be used on tinkercad.com with another Arduino. The second is a different version of the minimal main computer software file to be used on real hardware with the i386. The files are:

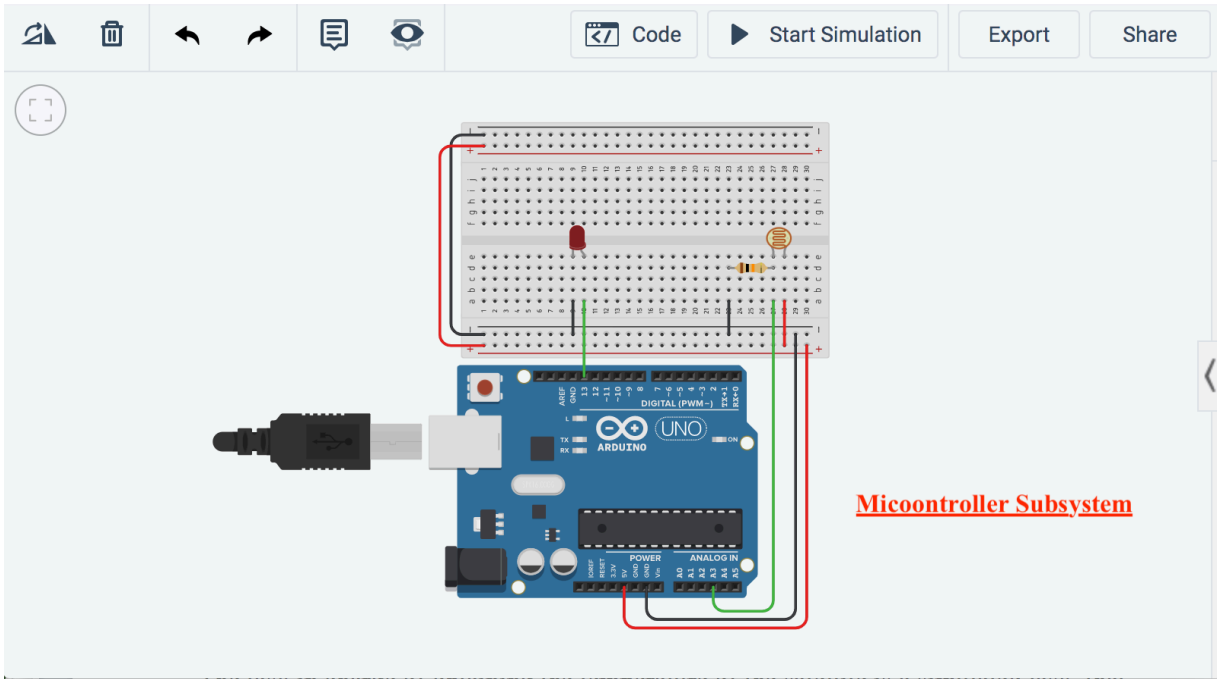
- ***arduino_msg_router.ino***: This is the source file for the minimal main computer software file to be used on tinkercad.com with another Arduino.
- ***rtems_msg_router.c***: This is the source file for the minimal main computer software file to be used on real hardware using a i386 PC.

2 Hardware description of the Arduino subsystem.

This subsystem is composed of an Arduino and a breadboard that contains the sensors and actuators. The sensor and actuators to place on the breadboard are the following:

- ***Sensor for sunlight***: This sensor is designed as an analog input device (See Arduino laboratory) that includes a light resistor.
- ***Led for heater state***: This actuator is designed as a digital output device (See Arduino laboratory) that includes a led.

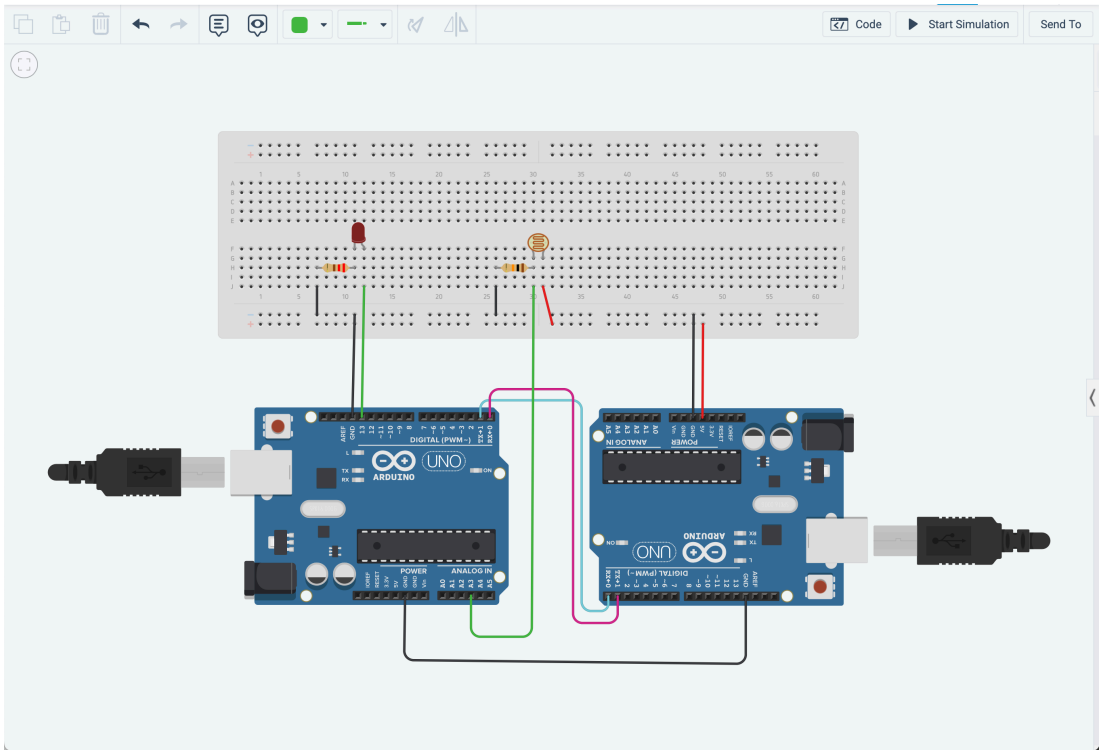
The following Figure shows the connection schema of this subsystem;



3 Testing using tinkercad.

To test the module using tinkercad we need to include another Arduino board connected with the original one. The connection is performed using the UART serial connections (pins 1 and 0) and ground. The new Arduino board acts as the main computer sending periodically all the possible messages and getting its responses. This Arduino should run the ***arduino_msg_router.ino*** code file. The serial monitor on this board will show a printout off all the commands sent and their corresponding responses.

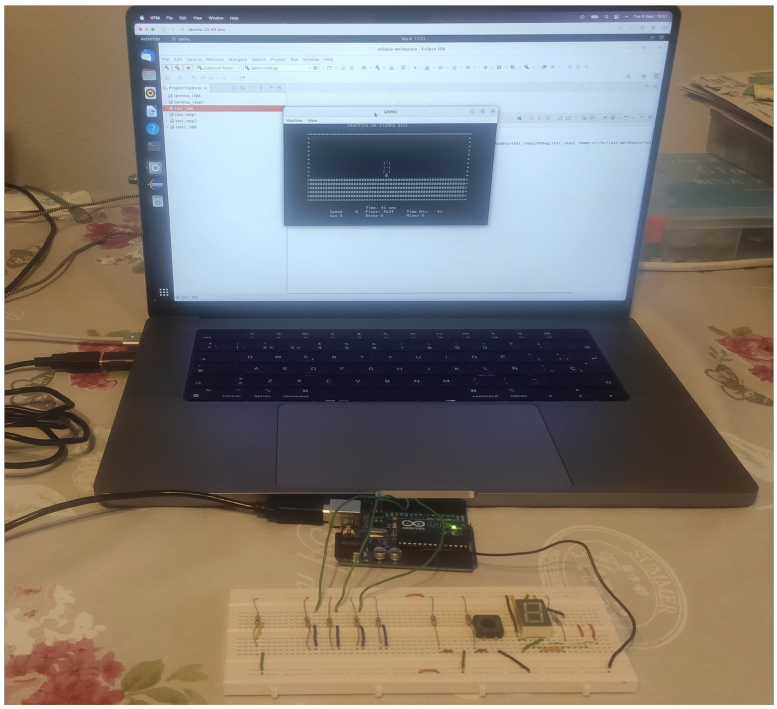
The following Figure shows the connection schema of the two Arduino boards and the rest of the system.



4 Testing using the i386 PC and RTEMS.

To test the module using the i386 PC and RTEMS we need the PC running the virtual machine with Ubuntu 22.04. This virtual machine should have installed the RTEMS development framework (as shown in laboratory 3). We also need a real implementation of the system using an actual Arduino board an breadboard. Both systems should be connected using a USB cable. This i386 PC should run the ***rtems_msg_router.c*** code file. The PC screen will show a Menu to select among all the commands to be sent and to get their corresponding responses.

The following Figure shows the connection schema of the i386 PC with Arduino board and the rest of the system.



5 Installing and using the Aduino IDE on Ubuntu 20.04

First, Install the Arduino IDE as following:

```
$ sudo snap install arduino
```

Change your **username** (example: **str**) to be include in the **dialout** group and reboot the machine:

```
$ sudo usermod -a -G dialout str
$ sudo reboot
```

Once the machine is booted. Open the Arduino program using the Graphical interface. Open the file called ***arduino_code.ino***

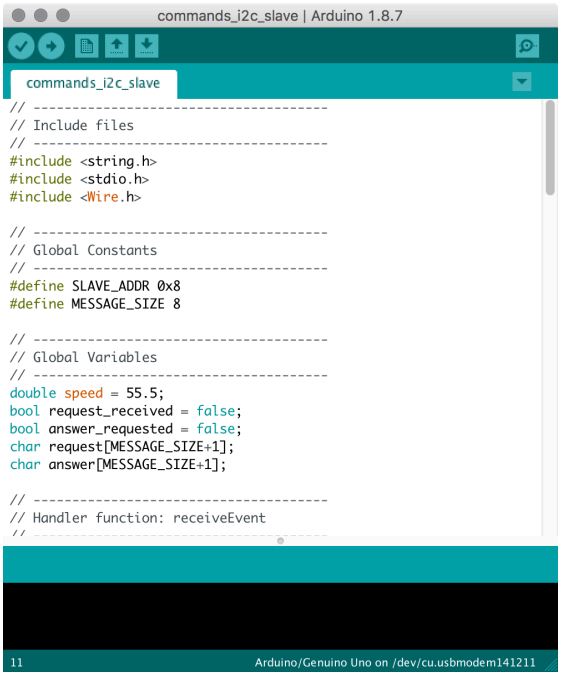


Figure 1: Arduino IDE example

To load the program onto the Arduino Hardware, Connect the Arduino UNO to the PC using the USB cable.

Then, configure the Arduino IDE to use the Arduino board (Tools -> board -> Arduino/Genuino UNO).

Also, configure the Arduino IDE to use the correct USB port (Tools -> port -> ....)

Finally, Compile the example code (Sketch -> Verify/Compile) and upload the example code to the Arduino (Sketch -> Upload)

6 Description of the software for the microcontroller subsystem.

The software of the microcontroller system is based on the Arduino code from the first part of the project with the necessary functions to include the USB-UART communication functions.

The whole code is comprised on a single file: ***arduino_code.ino***

This file includes the stubs for all the tasks of the subsystem and for the loop function. It also includes several pre-made functions:

- Function ***getClock ()***: This function returns a double value that represents the current time measured in seconds.
- Function ***delayClock ()***: This function sleeps an number of seconds expressed with a double value.
- Function ***comm_server ()***: This function receives the next command message and sends the next response massage.

The student should codify all these functions as following:

- Function ***get_temperature()***: This function should get the average temperature of the satellite. The student should use the version of this function developed on the first part of the project.
- Function ***get_position()***: This function should get the current position of the satellite on the orbit of the satellite. The student should use the version of this function developed on the first part of the project.
- Function ***exec_cmd_msg ()***: This function should receive commands from the main system, execute them and send back the response. The student should use the version of this function developed on the first part of the project.
- Function ***read_sun_sensor ()***: This function should get the status of the sunlight onto the satellite. The student should perform this function by reading the light resistor from the breadboard.
- Function ***set_heater ()***: This function should set the satellite heater on and off. The student should perform this function by turning on and off the led from the breadboard.

The student should also codify the main function as following:

- Function ***loop ()***: This function should implement a cycling scheduler for all the tasks mentioned above. This cycling scheduler should implement the following requirements.
  NOTE: The execution times are supposed to be 10 milliseconds ore less in all cases.
  NOTE 2: The memory should contain the design and validation of this cycling scheduler.

Tasks	Periodo/ Deadline (milisec.)	Execution time (milisec)
A: comm_server	200	10
B: exec_cmd_msg	200	10
C: get_position	100	10
D: get_temperature	100	10
E: read_sun_Sensor	50	10
F: set_heater	50	10

7 Documentation to be submitted

To submit the project, the submission web page for the course must be used. It is located on "Aula global" on the project paragraph.

The files to be delivered are the following:

memory.txt | memory.pdf | memory.doc

Project memory. Including the design and validation of the cycling scheduler

autores.txt

Names and NIAs of all the students involved in the project.

arduino_code.ino

Source file of the tasks for the Arduino subsystem.

<video_file>

A video recorded of a real Arduino board executing commands received from a i386 PC running the rtems_msg_router.c code file. The video should also show the breadboard with the LED and the Light Sensor.

Alternative a similar version of the video using the tinkercad and the arduino_msg_router.ino code file could be accepted, but with a lesser grade.

Project Deadline

Friday October 16, 2023.