Design of Embedded Systems

ESSTA, Energy Saving Smart-home distributed Temperature control Application

Implementation

Falzone Giovanni

jointly M.Sc Embedded Computing Systems

Sant'Anna School of Advanced Studies

University of Pisa

June 22, 2019

Contents

1 Introduction **2**
- 1.1 Central Unit . . . 2
  - 1.1.1 Graphical user interface . . . 2
  - 1.1.2 Communication . . . 3
- 1.2 Room Module . . . 3
  - 1.2.1 Energy Saving mode . . . 3
  - 1.2.2 Valve control . . . 3
  - 1.2.3 Communication . . . 4
  - 1.2.4 Errors . . . 4

2 Implementation, Central Unit module **5**
- 2.1 Wiring . . . 6
- 2.2 Graphic Interface . . . 6

3 Implementation, Room module **8**
- 3.1 Components description . . . 9
  - 3.1.1 ServoMotor . . . 9
  - 3.1.2 Valve position switches . . . 9
  - 3.1.3 Temperature-Humidity Sensor . . . 9
- 3.2 Wiring . . . 10

4 JSON messages implementation **12**
- 4.1 Request Message . . . 12
- 4.2 Status Message . . . 12

5 CUnit tests and simulation of the system **13**
- 5.1 Variables tests . . . 13
- 5.2 JSON Compliance tests . . . 13
- 5.3 Nominal testing and Network noise tests . . . 14

6 Future Work **15**

1 Introduction

The purpose of this project is to realize a smart-home application to control the heating system of a building based on the temperature of each room, in order to minimize the consumption of the building each room apply an energy saving function reducing the desired temperature when it is not needed. The system is composed by two differ modules

- central unit module
- room module

1.1 Central Unit

The *Central Unit* has the role of coordinator that retrieve the status of each room and computes the average values for the building.

1.1.1 Graphical user interface

Using a graphical User Interface the module represents the average values of the building and the values for each room, the graphical User Interface is composed by:

- *Main page* to represent the overview of the building
- *Room page* to represent the status of each room
- *Settings page* to set the desired temperature

Whenever the *Main page* is selected the module shall represent the average values among all the rooms for *Temperature*, *Humidity* and *Usage*.
Whenever the *Main page* is selected the module shall represent the *Energy Saving* if at least one room is set to *Energy Saving mode*.
Whenever the *Main page* is selected the module shall represent the *Warning* if at least one room is set to *crashed*.
Whenever the *Main page* is selected the module shall allow the user to move in the *Settings page*, next and previous *Room page*.

Whenever the *Settings page* is selected the module shall represent the *Desired Temperature* and shall allow the user to increase or deacrease it by a factor of 0.5 C°in the range of 15 C°and 30 C°.
Whenever the *Settings page* is selected the module shall represent the average values among all the rooms for *Humidity* and *Usage*.
Whenever the *Settings page* is selected the module shall represent the *Energy Saving* if at least one room is set to *Energy Saving mode*.
Whenever the *Settings page* is selected the module shall represent the *Warning* if at least one room is set to *crashed*.
Whenever the *Settings page* is selected the module shall allow the user to move in the *Main page*.

Whenever the *Room page* is selected the module shall represent the average values among all the rooms for *Temperature*, *Humidity* and *Usage*.
Whenever the *Room page* is selected the module shall represent the *Energy Saving* if at least one room is set to *Energy Saving mode*.
Whenever the *Room page* is selected the module shall represent the *Warning* if at least one room is set to *crashed*.
Whenever the *Room page* is selected the module shall allow the user to move in the *Main page*, *Settings page*, next and previous *Room page*.

The graphical user interface shall represent the following information as reported in the table 1.

Information	**Format**
Temperature	C°
Humidity	%
Usage	%
Energy Saving	boolean
Warning	boolean

Table 1: Display Information

1.1.2 Communication

Whenever a *Room Request message* is sent and the *Room Status message* is not received within 20s the module shall mark the room as **crashed**. The module shall send the *Room Request message* for each room at least every 30s.

1.2 Room Module

The purpose of this module is to control the temperature of the room acting on a valve in order to reach and maintain the *GoalTemperature*.

1.2.1 Energy Saving mode

In order to minimize the consumption the module keep tracks of the presence of motion inside the room using a motion sensor.

If a motion is detected in the last 30s, the module shall set the *GoalTemperature* to the one set by the user,otherwise the module shall set the *GoalTemperature* to:

$$GoalTemperature = DesiredTemperature - EnergySavingTemperatureOffset \quad (1)$$

Whenever the module is in *Energy Saving mode* it shall notify it through the *Interface*.

1.2.2 Valve control

In order to control the heating of the room the valve is moved to different positions based on the temperature error (*ActualTemperature* - *GoalThemperature*)

as in the following table, whenever one of these rules is valid the module shall move the valve in the correspondent position described in the third column of the table 2 as percentage of maximum flow.

rule	**valve position**	**Flow in %**
$error < -HIGH$	OPEN_POSITION	100
$error \in [-HIGH, -APPROCHING)$	HIGH_POSITION	75
$error \in [-APPROCHING, +APPROCHING]$	MIDDLE_POSITION	50
$error \in (APPROCHING, HIGH]$	LOW_POSITION	25
error $>$ HIGH	CLOSED_POSITION	0

Table 2

Whenever the valve is in *OPEN_POSITION* or *CLOSED_POSITION* the module shall check the position and set **Valve Error** if it is not valid.

In the following table3 are reported the temperature thresholds:

HIGH	2 C°
APPROCHING	1 C°

Table 3

1.2.3 Communication

Whenever a *Room Request message* is not received within 60s the module shall send the *Room Status Message* and set the **Communication error**.

1.2.4 Errors

Whenever one of the errors are set, the module shall notify it through the *Interface*. In the following table 4 are reported the possible errors.

Valve error
Communication error
Sensor error

Table 4

2 Implementation, Central Unit module

Figure 1: Central Unit module

The module is composed by:

- STM32F407VG DISCOVERY powered by ARM Cortex-M4 32-bit
- LCD board Touchscreen 3"5

In the discovery board are running the *Central Unit's* tasks on top of Erika RTOS, the system is composed by three different tasks reported in the table 5. Whenever a *Room Status* message arrives through the *USART* interface, the *Check message* aperiodic task is activated in order to check the message and execute the *Room's* functionalities.
The *Check message* task apply a simple JSON compliance test and check each field name and value of the message, then the message is converted in a *Room structure* and the Room's status and *Building* status are updated.
While the *Check message* task has the role of processing the incoming *Room Status*, the *Polling* task has the role of managing the rooms sending a check message periodically one room after the other, if the response *Room Status* message doesn't arrive within the next task's job it's considered as an error and the same *Room Request* message is sent until a maximum of 3 resend, then the room is marked as crashed.
The *Graphic* task has the role to represent the data of the building and for each room depending on the selected page by the user, in the settings page the user is able to set the desired temperature that will be sent starting from the next job of the *Polling* task.

Name	Frequency	Priority
Check message task	Aperiodic	3
Polling task	0.2Hz	2
Graphic task	10Hz	1
Receiver	USART background	/

Table 5: Tasks running on Central Unit

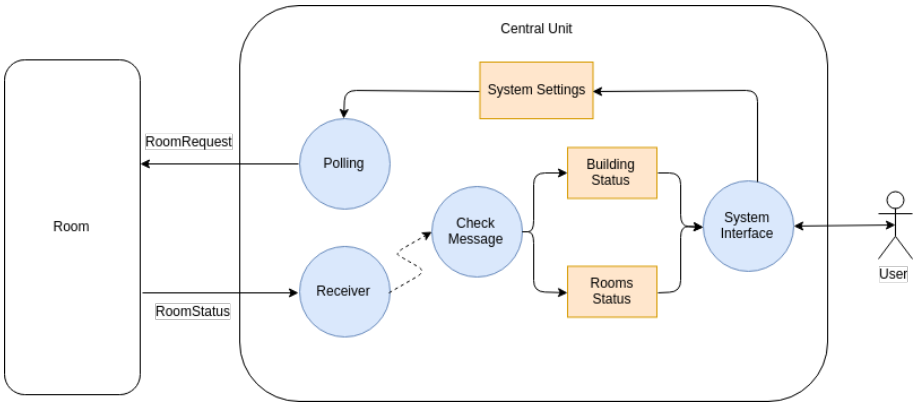


Figure 2: Central Unit blocks

2.1 Wiring

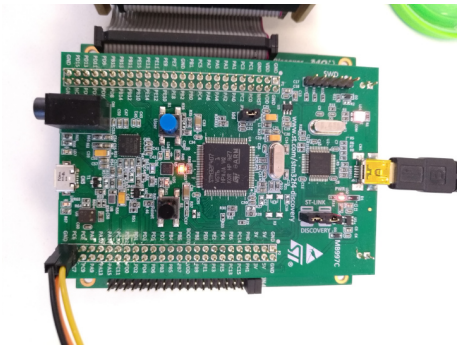


Figure 3: Discovery board wiring

The module is using the *USART6* of the discovery board to communicate with the *room* modules, in the following table is reported **USART**'s configuration.

USART	TX pin	RX pin	baud-rate	Parity Bit	Stop bits	Control Flow
6	PC6	PC7	9600	No	1	No

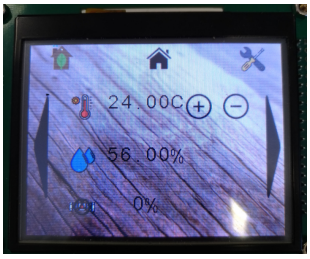
Table 6: USART configuration

2.2 Graphic Interface

The graphic interface is compose by three different screens:

- Home screen
- Settings screen
- Room screen

(a) Home screen



(b) Settings screen

(c) Room screen

The main page is the **home screen** reported in 4a, in this page are reported the average values of the building, If at least one room is marked as *crashed* then a warning appear on the home screen. If at least one room is in *eco mode* then an *eco* icon is displayed on the home screen. If the difference between the desired temperature and the actual average temperature is less then 0.5C°, the shown thermometer icon is hot otherwise cold. In the **settings screen** reported in 4b it is possible to set the desired temperature of the building that is the same for each room. In the **room screen** reported in 4c is displayed the status of the selected room, status composed by:

- Eco mode (the icon on the top-left of the screen)
- Temperature
- Humidity
- Valve position
- Warning (Warning icon between the eco icon and the icon in the top-center)

3 Implementation, Room module

The *Room* functionalities are running on a ATM328P, the tasks of the module are implemented as pseudo-periodical tasks using the *timer0* to check the activation time of each task, the tasks are reported in the following table.

Name	Frequency
Main task	0.5Hz
Control valve task	0.25Hz

Table 7: Tasks running on the Room module

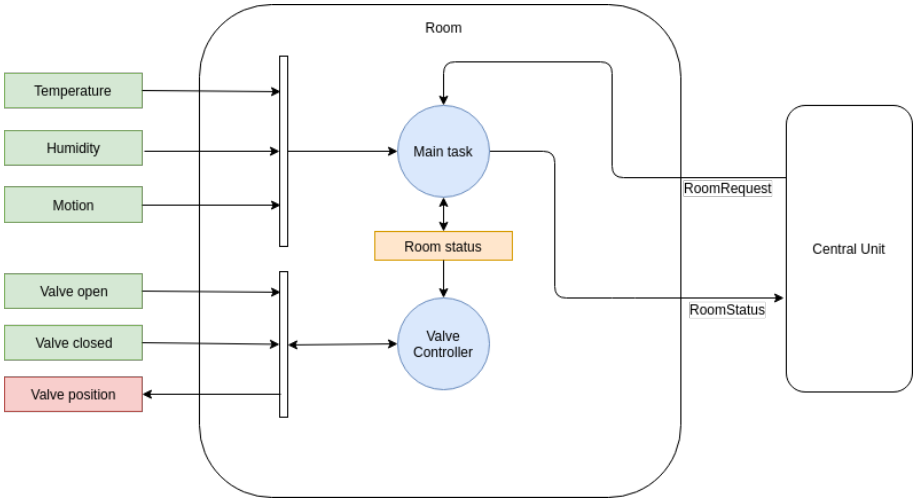
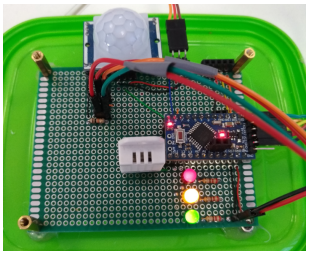
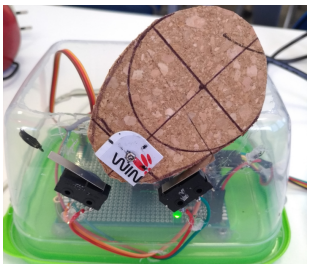


Figure 5: Room blocks



(a) Room module



(b) Valve

The module is composed by:

- Arduino pro mini, Atmega328P (8MHz, 3.3v logic)
- DHT22 Temperature and Humidity sensor
- PIR motion sensor

- Red led
- Green led
- Yellow led
- Servo Motor tower pro SG90
- 2 switch to check the open and closed position of the valve

3.1 Components description

3.1.1 ServoMotor

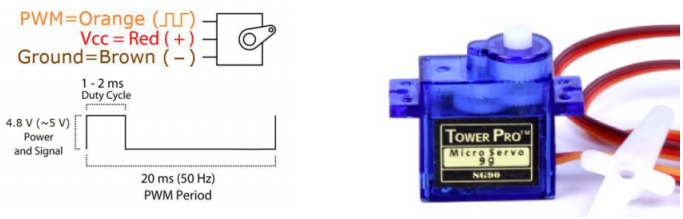


Figure 7: Servo Motor

The chosen Servo motor is a Tower Pro SG90 reported in 7, As reported in the 7, the signal used to move the servomotor in the correct position is a PWM and the duty-cycle describe the position that the motor has to maintain. The signal to control the motor position is managed by the Servo library using the **timer1** a 16-bit timer. The digital circuit inside the servomotor will adjust the position of the rotor using a potentiometer attached to that. In the following table are reported the characteristics of the servomotor:

Voltage(V)	4.8 - 6
Torque(Kg-cm)	2.5
Speed(sec)	0.1
Weight(g)	14.7

Table 8: Servo motor characteristics

3.1.2 Valve position switches

In order to check the position of the valve, two switches are fixed in the open and closed position, during the *valve initialization phase* the possible positions are computed starting from the open and closed position. Whenever the valve is in the open or closed position a check is done using the switches.

3.1.3 Temperature-Humidity Sensor

The sensor DHT22 have been used for the project, in the following tables are reported the features of the sensors. The sensor has a dedicated one-wire communication protocol implemented by the DHT adafruit library, a pull-up resistor is used in order to maintain the line clear when no one is transmitting.

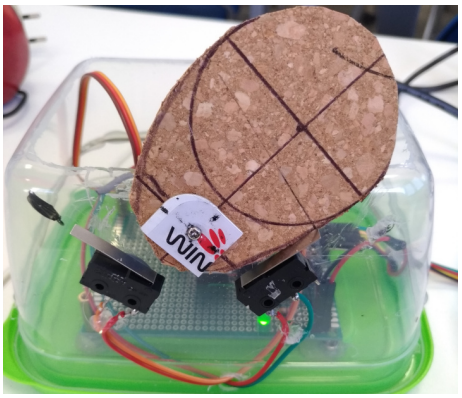


Figure 8: Discovery board wiring

Voltage(V)	3 - 5
Current(mA)	2.5
Humidity(%)	0 - 100
Humidity Accuracy(%)	2 - 5
Temperature(C°)	-40 - 80
Temperature Accuracy(C°)	+/- 0.5
Sampling rate(Hz)	0.5

Table 9: DHT22 characteristics

3.2 Wiring

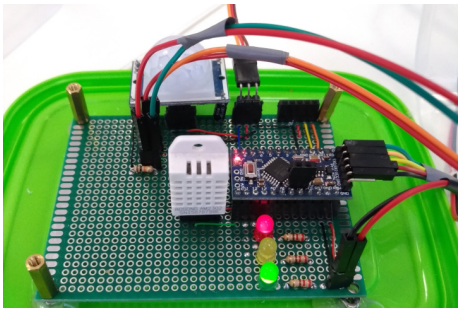


Figure 9: Room board wiring

The board is powered by 5V external power attached to the *Raw pin*, this source is shared with the servomotor and the PIR sensor. In 9 is reported a picture of the implementation, in the following table the pins' configuration.

Red led	13	D OUTPUT
Yellow led	12	D OUTPUT
Green data	11	D OUTPUT
DHT data	10	D INPUT
PIR data	9	D INPUT
ServoMotor PWM	8	D OUTPUT
Open Switch	7	D INPUT
Closed Switch	6	D INPUT
USART TX	0	D OUTPUT
USART RX	1	D INPUT

Table 10: Atmega328P Wiring

4 JSON messages implementation

The modules exchange the informations *Room request* and *Room status* in a *JSON* format. These messages have been implemented using the *JSON* format, in the following are reported two messages exchanged during execution. In the following version of the project the field names and the order of the variables must follow this example otherwise the simple JSON parser will consider the message as corrupted.

4.1 Request Message

```
{
  "Id":"01",
  "DTemp":"30.00"
}
```

4.2 Status Message

```
{
  "Id":"02","Eco":"0",
  "sens":[
    {"Nm":"Tmp","Val":"23.70","Fmt":"C"},
    {"Nm":"Hum","Val":"051.90","Fmt":"\%"}
  ],
  "acts":[
    {"Nm":"Vlv","Val":"100","Fmt":"\%"}
  ]
}
```

5 CUnit tests and simulation of the system

In order to check the behaviour of the system a sequence of tests have been executed on the *ESTA_ JSON_ lib* functions, that is used to convert the incoming JSON string from the module *Room*.

5.1 Variables tests

For each variable present in the JSON message a *Robustness Boundary Values test* is exeuted maintaining the other variables in the nominal value.

Variable	Min	Max	1°	2°	3°	4°	5°	6°	7°
Id	1	2	0	1	2	3	/	/	/
Eco	0	1	-1	0	1	2	/	/	/
Temperature	15.0	30.0	14.99	15.0	15.01	29.99	30.00	30.01	22.5
Humidity	0.0	100.0	-0.01	0.0	0.01	99.99	100.00	100.01	50.00
Valve	0	100	-1	0	1	99	100	101	50

Table 11: Variables tests

5.2 JSON Compliance tests

In order to check the *JSON compliance code checker* the following tests are applied modifying a correct message.

Number	Description
1	Missing odd number of *{*
2	Missing odd number of *"*
3	Missing odd number of *[*
4	Check field name *Id*
5	Check field name *Eco*
6	Check field name *Nm*
7	Check field name *Val*
8	Check field name *Fmt*
9	Check field name *Tmp*
10	Check field name *Hum*
11	Check field name *Vlv*
12	Check Temperature format value *C*
13	Check Humidity format value *%*
14	Check Valve format value *%*

Table 12: JSON compliance tests

In the following picture 10 is reported the result of the tests executed using the framework *CUnit* Version 2.1.3.

```
Suite: Suite_1
  Test: testJSON_check_message ...passed
  Test: testJSON_ID ...passed
  Test: testJSON_eco ...passed
  Test: testJSON_temperature ...passed
  Test: testJSON_humidity ...passed
  Test: testJSON_valve ...passed
Completing suite1


Run Summary:    Type  Total    Ran Passed Failed Inactive
              suites      1      1    n/a      0        0
               tests      6      6      6      0        0
             asserts     43     43     43      0      n/a

Elapsed time =    0.000 seconds
```

Figure 10: CUnit results

5.3 Nominal testing and Network noise tests

In order to check the correctness behaviour of the *Central Unit* module a python script that simulates the rooms has been created and used to test the module simulating two different rooms. The **Room simulator** creates the correct JSON string with random values for the variables and send it; in order to simulate the **noise** of the network this cases have been considered:

Number	Description	Probability
1	Message lost, the response message is discarded	0.1
2	Message corrupted, 3 random characters are replaced with *?*	0.1

Table 13: Network noise tests

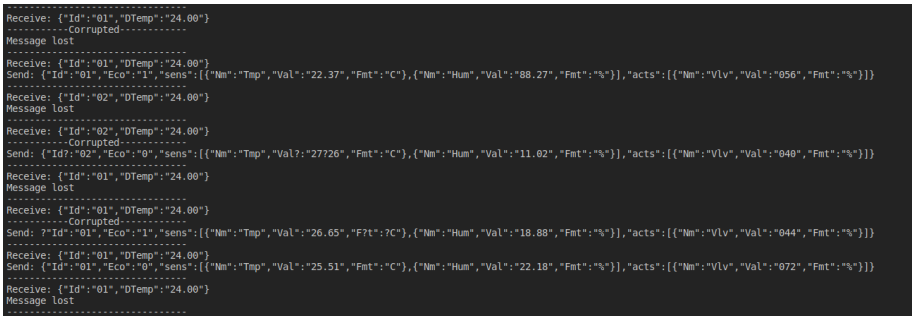


Figure 11: Room simulator

6 Future Work

This kind of **energy saving** application strictly depends on the usage of the building and the building itself.
In order to maximize the effectiveness of the heating system and minimize the consumption, different parameters play an important role, I report some of them in the following list

- room characteristics (dimension, orientation, furniture, etc.)
- usage of the room strictly depends on the users' life style
- external weather condition
- period of the year

These parameters have to be taken in consideration to compute the **energy threshold** of each room in such a way that the single room can reach the desired temperature in the shortest time.

One of the possible future work could be to integrate a *Machine Learning* functionality in order to adjust this *energy threshold* parameter considering all the information related to the construction of the building and the usage coming from the sensors in the rooms.