



Design of Embedded Systems

ESSTA, Energy Saving Smart-home distributed
Temperature control Application

Falzone Giovanni

jointly M.Sc Embedded Computing Systems

Sant'Anna School of Advanced Studies

University of Pisa

June 5, 2019

Contents

1	Introduction	2
2	Room Module	3
2.1	Temperature control	3
2.2	Energy Saving mode	3
2.3	User Requirements	4
2.4	Functional requirements	4
3	Central Unit	7
3.1	User Requirements	7
3.2	Functional requirements	7
4	Central Unit	8
5	ESSTA SySML description	8
6	Implementationm, Central Unit module	9
6.1	Wiring	9
6.2	Graphic Interface	10
7	Implementation, Room module	11
7.1	Components description	11
7.1.1	ServoMotor	11
7.1.2	Valve position switches	12
7.1.3	Temperature-Humidity Sensor	12
7.2	Wiring	13
8	Implementation values	14
8.1	Parameters	14
8.2	JSON messages	14
8.2.1	Request Message	14
8.2.2	Status Message	14
9	CUnit tests of the system	14
10	Hardware implementation	14
10.1	Room module	14
10.2	Central Unit moduke	15
10.3	Network configuration	15

1 Introduction

The goal of this project is to realize a smart-home application to control the temperature of different room in order to minimize the consumption of the entire building.

The system is composed by two differ modules

- A central unit
- A room module

The *Central Unit* module has the role of a coordinator that check the status of each room and represents the status of each room in the LCD touchscreen panel, the module allow to set the desired temperature for the builng.

The *Central Unit* has also the role to compute the average values of the entire building in order to represent the status of the building.

The *Room* module has the aim of manage the room sensing the temperature, humididty, presence of motion and acting on a valve in order to reach and mantain the desired temperature.

2 Room Module

The aim of this module is to control the temperature of the room and act on the valve in order to achieve the desired temperature set by the user through the *Central Unit* but minimizing the consumption reducing the desired temperature when the room is not used.

The module is composed by:

- Temperature sensor
- Humidity sensor
- Motion sensor
- Valve actuator
- Error led
- Motion detection led
- Eco mode led

2.1 Temperature control

The module adjust the temperature of the room acting on a valve. The valve is set to different positions based on the temperature error (difference between the actual temperature and the desired themperature):

- if the temperature error is below then -COLD_THRESHOLD the valve is moved in OPEN_POSITION
- if the temperature error is below then -WARM_THRESHOLD the valve is moved in HIGH_POSITION
- if the temperature error is between a [-APPROCHING_THRESHOLD, APPROCHING_THRESHOLD] the valve is moved in HALF_POSITION
- if the temperature error is greater then WARM_THRESHOLD the valve is moved in LOW_POSITION
- if the temperature error is greater then HOT_THRESHOLD the valve is moved in CLOSED_POSITION

2.2 Energy Saving mode

In order to minimize the consumpion the module keep tracks of the presence of motion inside the room using a motion sensor. If a predefined number of motion is detected in a time slot, the module set the desired temperature to the one set by the user. If there is the number of motions counted is greater then a predefined threshold the module set the desired temperature to a value equal to the user desired temperature minus a default value ENERGY_SAVING_TEMPERATURE_DIFFERENCE.

2.3 User Requirements

- 2.3.1. When a motion is not detected for a predefined period the module shall move in Eco mode
- 2.3.2. When the difference between the actual temperature and the desired temperature is included in the range [-APPROACHING_THRESHOLD, APPROACHING_THRESHOLD] the valve shall be in HALF_POSITION

2.4 Functional requirements

2.4.1. Initialization

- 2.4.1.1. During the initialization phase the module shall check the valve moving it from the DAFULT_CLOSED_POSITION to the DA-FULT_OPEN_POSITION
- 2.4.1.2. During the initialization phase the module shall set the CLOSED_POSITION, LOW_POSITION, MIDDLE_POSITION, HIGH_POSITION, OPEN_POSITION
- 2.4.1.3. During the initialization phase the module shall check the temperature sensor until a correct value is received
- 2.4.1.4. During the initialization phase the module shall check the humidity sensor until a correct value is received

2.4.2. Communication

- 2.4.2.1. Whenever the module receive a correct request message from *Central Unit* it shall send its status in conformance with JSON format
 - 2.4.2.1.1. The status message shall include its ID in the status message
 - 2.4.2.1.2. The status message shall include the Eco mode status
 - 2.4.2.1.3. The status message shall include its sensors list
 - 2.4.2.1.4. The status message shall include its actuators list
 - 2.4.2.1.5. The status message shall include the name of each sensor and actuator
 - 2.4.2.1.6. The status message shall include the format for each numerical value
- 2.4.2.2. Whenever a request message from the *Central Unit* is corrupted the module shall go in COMMUNICATION_ERROR

2.4.3. Valve management

- 2.4.3.1. The module shall change the position of the valve every VALVE_PERIOD seconds
 - 2.4.3.1.1. The valve shall be in OPEN_POSITION whenever the difference between the actual temperature and the desired temperature is below -COLD_THRESHOLD C°
 - 2.4.3.1.2. The valve shall be in HIGH_POSITION whenever the difference between the actual temperature and the desired temperature is greater then -COLD_THRESHOLD C°and below -APPROACHING_THRESHOLD C°

- 2.4.3.1.3. The valve shall be in MIDDLE_POSITION whenever the difference between the actual temperature and the desired temperature is greater or equal than -APPROACHING_THRESHOLD C° and below or equal than APPROACHING_THRESHOLD C°
- 2.4.3.1.4. The valve shall be in LOW_POSITION whenever the difference between the actual temperature and the desired temperature is greater than APPROACHING_THRESHOLD C° and below or equal than HOT_THRESHOLD C°
- 2.4.3.1.5. The valve shall be in CLOSED_POSITION whenever the difference between the actual temperature and the desired temperature is greater than HOT_TEMP C°

2.4.4. Sensors management

- 2.4.4.1. The module shall update the actual temperature and humidity every TEMPERATURE_HUMIDITY_PERIOD seconds
- 2.4.4.2. Whenever the actual temperature is below MIN_TEMPERATURE C° or greater than MAX_TEMPERATURE C° the module shall go in temperature error state
- 2.4.4.3. The module shall update the presence of motion every MOTION_PERIOD seconds
- 2.4.4.4. Whenever a motion is detected the module shall increase the MOTION_COUNTER otherwise decrease until zero
- 2.4.4.5. Whenever the MOTION_COUNTER reaches the MOTION_THRESHOLD value the module shall notify it turning on the MOTION_LED and updating the motion status
- 2.4.4.6. Whenever the MOTION_COUNTER is below the MOTION_THRESHOLD value the module shall notify it turning off the MOTION_LED and updating the motion status
- 2.4.4.7. Whenever the valve is in OPEN_POSITION the module shall check the OPEN_POSITION_SENSOR and raise a VALVE_ERROR if it is not consistent
- 2.4.4.8. Whenever the valve is in CLOSED_POSITION the module shall check the CLOSED_POSITION_SENSOR and raise a VALVE_ERROR if it is not consistent

2.4.5. Energy Saving management

- 2.4.5.1. Whenever the motion status is set the module shall move in normal mode otherwise in Eco mode
- 2.4.5.2. Whenever the module is in Eco mode the module shall notify it turning on the ECO_MODE_LED
- 2.4.5.3. Whenever the module is in Eco mode the module shall set the desired temperature with the difference between the desired temperature and the ENERGY_SAVING_TEMPERATURE_DIFFERENCE

2.4.6. Error handling

- 2.4.6.1. Whenever an error case is achieved the module shall notify the presence of errors turning on the ERROR_LED
- 2.4.6.2. Whenever a valve error case is reached the module shall recompute the positions values

3 Central Unit

3.1 User Requirements

- 3.1.1. The module shall represent the average temperature of the building in *Celsius* format
- 3.1.2. The module shall represent the average humidity of the building in *Percentage*
- 3.1.3. The module shall represent the usage of the heating system in *Percentage*
- 3.1.4. The module shall represent the average temperature of each room in *Celsius* format
- 3.1.5. The module shall represent the average humidity of each room in *Percentage*
- 3.1.6. The module shall represent the usage of each room in *Percentage*
- 3.1.7. The module shall allow to set a desired temperature in *Celsius* for the building
- 3.1.8. The module shall represent the reachability of each room

3.2 Functional requirements

3.2.1. Running

- 3.2.1.1. The module shall send a *request message* every POLLING_PERIOD
- 3.2.1.2. The module shall update the graphic at least every GRAPHIC_PERIOD
- 3.2.1.3. The module shall check the JSON compliance of the received messages and discard the corrupted ones

3.2.2. Communication

- 3.2.2.1. The module shall send the *request message* in JSON compliant format
 - 3.2.2.1.1. The request message shall include the *ID* of the target room
 - 3.2.2.1.2. The request message shall include the *Desired Temperature*
- 3.2.2.2. The module shall check the compliance of the received messages
- 3.2.2.3. The module shall check the correctness of the data in the JSON messages
- 3.2.2.4. The module shall discard every corrupted received message and resend the request
- 3.2.2.5. The module shall resend the same request at least 3 times before marking the target room as crashed

3.2.3. Values

- 3.2.3.1. The temperature received shall be in the range [MIN_TEMPERATURE, MAX_TEMPERATURE]
- 3.2.3.2. The humidity received shall be in the range [MAX_HUMIDITY, MIN_HUMIDITY]
- 3.2.3.3. The valve status received shall be in the range [MIN_VALVE, MAX_VALVE]

4 Central Unit

5 ESSTA SySML description

6 Implementationm, Central Unit module

In this chapter it is shown the Implementation of each module, the hardware is chosen to be compliant with the requirements of the project.



Figure 1: Central Unit module

The module is composed by:

- STM32F407VG DISCOVERY
- LCD board Touchscreen 3"5

On the discovery board is running the code on top of Erika RTOS, the system is composed by three different task:

- **Graphic task** running at 10Hz (every 100ms)
- **Polling task** running at 0.2Hz (every 5s)
- **Check Message**, aperiodic task

6.1 Wiring



Figure 2: Discovery board wiring

The module is using the **USART6** of the discovery board to communicate with the *room* module, in the following table are reported the features of the **USART6**

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

6.2 Graphic Interface



(a) Home screen

(b) Settings screen



(c) Room screen

The graphic interface is composed by three different screens:

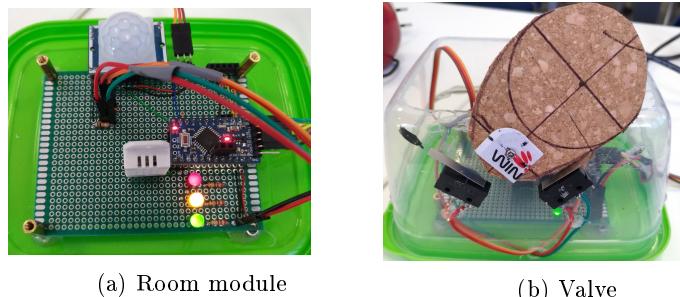
- Home screen
- Settings screen
- Room screen

The main page is the **home screen** reported in 3a, in this page the user can see the average values of the building, if no room is attached it will display zero or the last computed value. If at least one room is in *room crashed mode* 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. Depending on the difference between the desired temperature and the actual average temperature the thermometer icon will change in cold or hot. In the **settings screen** reported in 3b it is possible to set the desired temperature of the building. In the **room screen** reported in 3c is displayed the status of the selected room, status composed by:

- Eco mode
- Temperature
- Humidity
- Valve position

7 Implementation, Room module

The room module is implemented on a ATM328P, the code is implemented using the Arduino toolchain. The tasks of the module are implemented as pseudo-periodical tasks using the *timer0* to check the activation time of each task branch.



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

7.1 Components description

7.1.1 ServoMotor

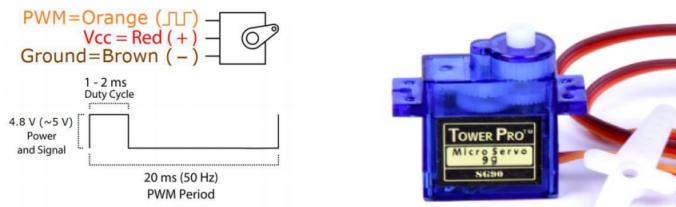


Figure 5: Servo Motor

The chosen ServoMotor is a Tower Pro SG90 reported in 5, As reported in the 5, 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

7.1.2 Valve position switches

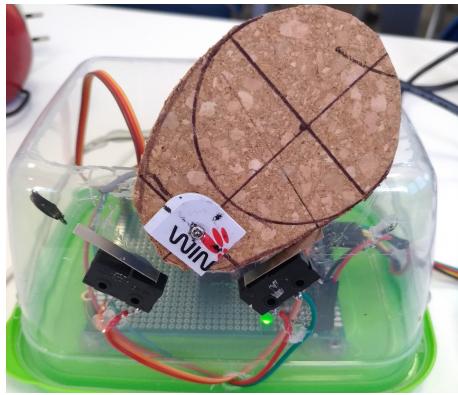


Figure 6: Discovery board wiring

In order to check the position of the valve, two switches are fixed in the open and closed position. During the *init_valve* phase the position in degree is saved in order to adjust the possible positions of the valve. Whenever the valve goes in one of these positions a check is done using the correct switch.

7.1.3 Temperature-Humidity Sensor

The chosen sensor is a DHT22 that is perfect to work in closed environment, a pull-up resistor, connecting the data pin to VCC, is used in order to mantain the line clear when no one is transmitting. The sensor has a dedicated one-wire communication protocol implemented by the DHT adafruit library.

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

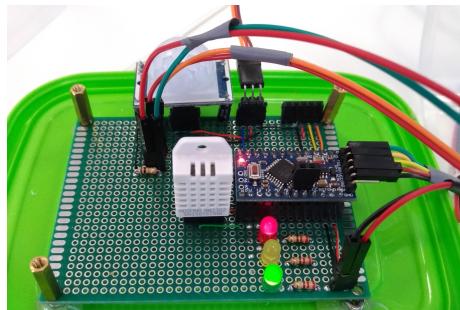


Figure 7: Room board wiring

7.2 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 7 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

8 Implementation values

8.1 Parameters

DEFAULT_OPEN_POSITION	160
DEFAULT_HIGH_POSITION	115
DEFAULT_MIDDLE_POSITION	70
DEFAULT_LOW_POSITION	45
DEFAULT_CLOSED_POSITION	20
MIN_TEMPERATURE	15
MAX_TEMPERATURE	30
MIN_HUMIDITY	0
MAX_HUMIDITY	100
MIN_VALVE	0
MAX_VALVE	100

8.2 JSON messages

8.2.1 Request Message

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

8.2.2 Status Message

```
|| {  
  "Id": "02", "Eco": "0",  
  "sens": [  
    {"Nm": "Tmp", "Val": "23.70", "Fmt": "C"},  
    {"Nm": "Hum", "Val": "51.90", "Fmt": "\%"}  
  ],  
  "acts": [  
    {"Nm": "Vlv", "Val": "100", "Fmt": "\%"}  
  ]  
}
```

9 CUnit tests of the system

10 Hardware implementation

In this chapter it is shown the hardware implementation of each module, the hardware is chosen to be compliant with the requirements of the project.

10.1 Room module

The room module is implemented on a ATM328P

- Atmega328P
- Xbee Pro S1

- DHT22 Temperature and Humidity sensor
- PIR motion sensor
- Red led
- Green led
- Yellow led

10.2 Central Unit module

- STM32F407VG DISCOVERY
- 3"5 LCD board Touchscreen
- Relè switch
- Fan

10.3 Network configuration

PAN ID:

Device	Role	ATID	ATDL	ATMY
<i>Central Unit</i>	Coordinator	1	87837	787
<i>Room 1</i>	End device	2	78	5415
<i>Room 2</i>	End device	3	78	5415