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 3, 2019

Contents

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
- Wireless communication module
- Error led
- Eco mode led
- Motion detection 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 thresholf 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. Whenever the module is turn on it shall send an initialization message to the *Central Unit* and wait for the response

2.4.1.2. During the initialization phase the module shall continue blinking the ERROR_LED

2.4.1.3. During the initialization phase the module shall check the valve moving it from the CLOSED_POSITION to the OPEN_POSITION

2.4.1.4. During the initialization phase the module shall check the temperature sensor until a correct value is received

2.4.1.5. 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. The module shall move in COMMUNICATION_ERROR status if does not receive the check message from the *Central Unit* within 1 minute

2.4.2.2. The module shall send its status to the *Central Unit* every 10 seconds

2.4.2.3. The module shall send its status in conformance with JSON format

2.4.2.3.1. The status message shall include its ID in the status message

2.4.2.3.2. The status message shall include the Eco mode status

2.4.2.3.3. The status message shall include its sensors list

2.4.2.3.4. The status message shall include its actuators list

2.4.2.3.5. The status message shall include the name of every sensor and actuator

2.4.2.3.6. The status message shall include the format for every numerical value

2.4.2.4. Whenever a check 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 30 seconds

2.4.3.2. The valve shall be in one of the allowed positions

2.4.3.2.1. The valve shall be in OPEN_POSITION whenever the difference between the actual temperature and the desired temperature is below -COLD_THRESHOLD C°

2.4.3.2.2. 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°

2.4.3.2.3. The valve shall be in HALF_POSITION whenever the difference between the actual temperature and the desired temperature is greater or equal then -APPROACHING_THRESHOLD C°and below or equal then APPROACHING_THRESHOLD C°

2.4.3.2.4. The valve shall be in LOW_POSITION whenever the difference between the actual temperature and the desired temperature is greater then APPROACHING_THRESHOLD C°and below or equal then HOT_THRESHOLD C°

2.4.3.2.5. The valve shall be in CLOSED_POSITION whenever the difference between the actual temperature and the desired temperature is greater then HOT_TEMP C°

2.4.4. **Sensors management**

2.4.4.1. The module shall update the actual temperature every 10 seconds

2.4.4.2. Whenever the actual temperature is below 15 C°or greater then 40 C°the module shall go in temperature error state

2.4.4.3. The module shall update the actual humidity every 10 seconds

2.4.4.4. The module shall update the presence of motion every 5 seconds

2.4.4.5. Whenever a motion is detected the module shall notify it turning on the MOTION_LED

2.4.4.6. Whenever a motion is detected the module shall increase a MOTION_COUNTER value

2.4.4.7. Whenever a motion is not detected and the MOTION_COUNTER is greater then zero the module shall decrease the MOTION_COUNTER

2.4.5. **Energy Saving management**

2.4.5.1. Whenever the MOTION_COUNTER reaches the MOTION_THRESOLD value the module shall move in normal mode

2.4.5.2. Whenever the MOTION_COUNTER is below the MOTION_THRESOLD value the module shall move in Eco mode

2.4.5.3. Whenever the module is in Eco mode the module shall notify it turning on the ECO_MODE_LED

2.4.5.4. 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 precense of errors turning on the ERROR_LED

2.4.6.2. Whenever an error case is achieved the module shall continue sending a status message with the error field set to 1 to the *Central Unit*

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 averahe humidity of the builg 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 averahe humidity of each *Percentage*

3.1.6. The module shall represent the usage of each room *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 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 systen 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

4.1 Wiring

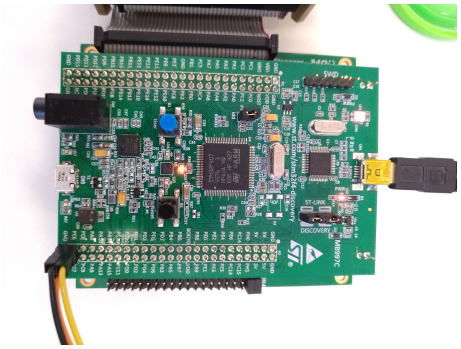
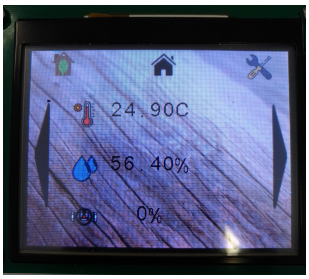


Figure 2: Discovery board wiring

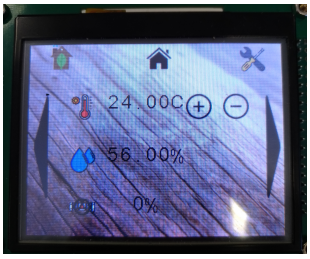
The module is using the *USART6* of the discovery board to comunicate with the *room* module, in the following table are reported the features of the **USART6**

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

4.2 Graphic Interface



(a) Home screen



(b) Settings screen

(c) Room screen

The graphic interface is compose by three different screens:

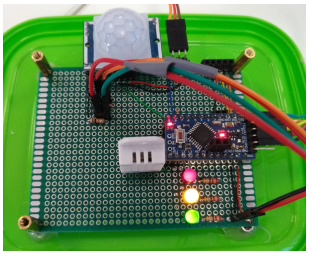
- Home screen
- Settings screen
- Room screen

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 termometer 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:

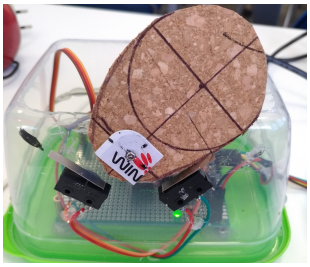
- Eco mode
- Temperature
- Humidity
- Valve position

5 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.



(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

5.1 Components description

5.1.1 ServoMotor

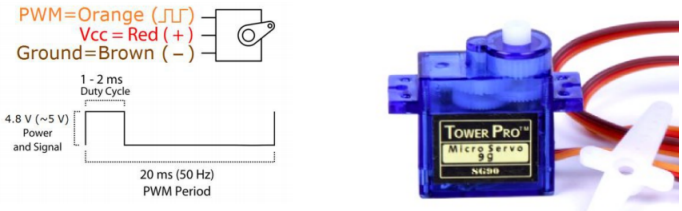


Figure 5: Servo Motor

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

5.1.2 Valve position switches

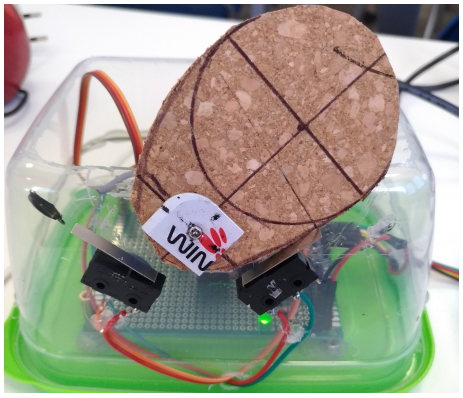


Figure 6: Discovery board wiring

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.

5.1.3 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

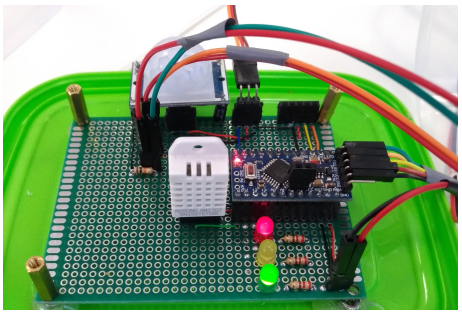


Figure 7: Room board wiring

5.2 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

References

[1]