Instituto Superior Técnico

91215 MEIC-A

91214 MEIC-A

Ambient intelligence: Smart Home Project

18.05.2018

1 Introduction

1.1 Motivation

Saving electrical energy is important for multiple environmental reasons[1], as well for cutting costs. Our main motivation is to create a system that will give an electrical efficient and intuitive way of controlling your house, as well the exploration of combining “smart card”-technology at hardware level with higher level database systems.

1.2 Objectives

The goals of this project are plenty. We wish to obtain a simple and effective communication protocol between a high-level system and a hardware system. This will help us include multiple user interaction features, and more complex modules in the system.
Another goal was to create a simple user experience. This would make a transition for using this system as a part of the daily routine simple and possible.
Our main goal was to create a system which would allow an intelligent behaviour of our system, with a hierarchical control structure and a user based interaction model.

1.3 Related work

We used the domo bus structure for inspiration when building the hardware structure. We utilize the same room and device structure, with device type definitions, and a connection to a specific room. The strengths of this structure is the possibility to define specific functions for similar devices, as well as rooms, and will make the expansion of the system less problematic.

The X10 system is an communication protocol between electrical devices in your home[2]. The X10 system supports multiple different modules using the electrical grid for communication. Our message system is similar to that of X10; using both starting signals and metadata together with data to translate the messages[3]. We have some differences in using a dynamic message length, supplying the message length. This makes message sending more efficient, not needing multiple messages for single commands.

Radio-Frequency IDentification, or shortened RFID, is a technology with huge potential and is already widely used in ID cards, to track merchandise, subway passes and so on. One of the huge advantages of RFID is data stored in the RFID tags have read and write

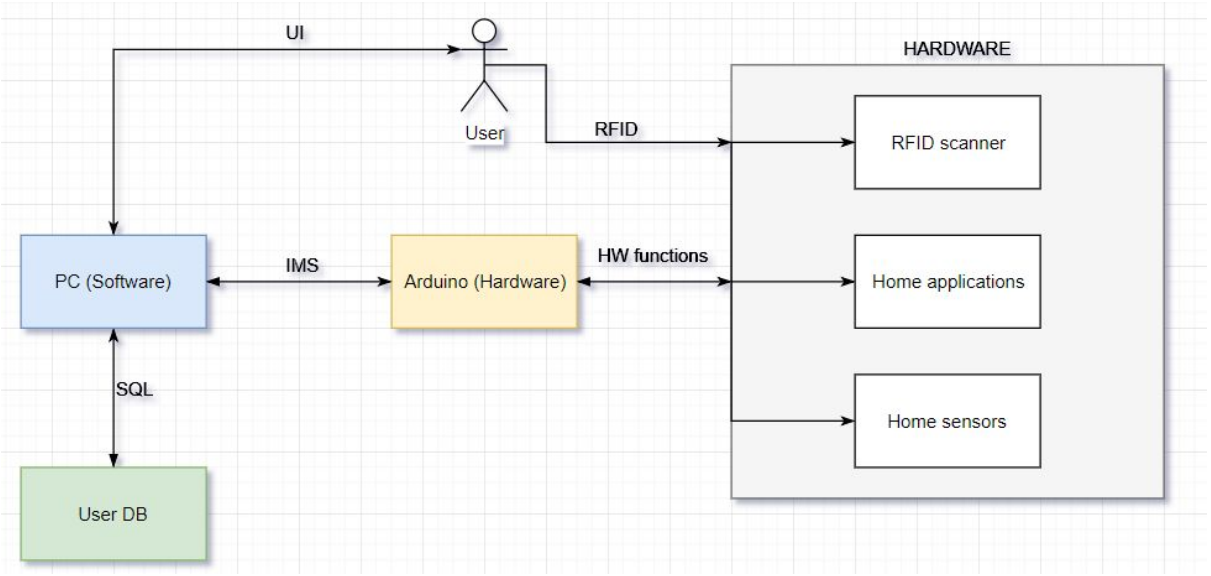
[1]“Why conserve energy”-BY LEGEND- http://legendpower.com/uncategorized/why-conserve-energy/
[2] “X10 industry standard” -BY WIKIPEDIA - https://en.wikipedia.org/wiki/X10_(industry_standard)
[3]“Technology X10 for Home Automation” - BY Renato Nunes
https://fenix.tecnico.ulisboa.pt/downloadFile/1407993358869443/AI_04_X10.pdf, 15.05.2018

capabilities. RFID technology are starting to dominate payment solutions, and it easy to use. [4]RFID chipping of animals are widely used, but how about chipping humans? This way you could have your ID, bank card and reward cards in a chip under your skin. Pickpockets will no longer be able to physically steal your valuables. However, security and protection against unwanted surveillance will be more important than ever.

Perhaps the biggest issue with RFID tags is the security aspects of reading a tag. A RFID tag can not differ from one reader to another, and it can be read within a distance of a few meters. This way the user is vulnerable to eavesdropping and unwanted writing to the user's tag[5]. In our system we have not focused on security, the only information we are extracting from the RFID tag is the user ID. In our project we want to use this technology, and use the unique ID of a RFID tag to recognize a user. When an user enters a room and scan their RFID tag on the reader, the ID of the tag is checked against the database and information is accessed about this user, and eventually set desired preferences in the room

In general we would use a Raspberry pie or any similar technology in general function for the software system. Raspberry is often used in building smart houses, and in other technology fields, like robotics and space science[6]. The Raspberry serves a low power usage and potential for complex systems, which are important for our system.

2 System description



Top level architecture

In this section the system itself will be described. It will describe how the system is built up, the different modules and their functions, and how the system communicates with its different components.

[4] "How RFID works" - BY KEVIN BONSOR & WESLEY FENLON
https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/rfid1.htm
[5] "An introduction to RFID technology" - BY ROY WANT (Published: IEEE Pervasive Computing, 2005)
https://ieeexplore.ieee.org/document/1593568/?part=1
[6] "Rasperry pie, Hardware" - BY WIKIPEDIA https://en.wikipedia.org/wiki/Raspberry_Pi#Hardware

2.1 System architecture

As seen in the illustration above, the system is made by four main modules. These modules are software, hardware controller (Arduino), user database and hardware applications. Each of these modules, and how they interact with each other, will be described in details in this section.

2.1.1 Software module

The role of the software module is to handle more complex actions, such as extended communication with users and data storage.
It will communicate and obtain information from a user using the GUI-submodule through the UI-submodule. To communicate and send commands to the hardware controller module, it uses the Internal Message System (IMS). It will communicate with the database using SQL to extract information and add data obtained from the user or the hardware controller module.
This module is important for the system to be able to act intelligent.

2.1.2 Hardware controller module

The hardware controller is used for more meaningful interaction with the outside world, such as controlling devices and obtaining data from sensors.
It uses generalized functions and methods to control different home applications and devices. It will obtain data from home sensors, such as thermometers. It also operates the RFID component, and reads data from usage of this component. It will send these data to software module using the IMS.
This module is used for actuating on the real world, and to serve a simpler way of communicating with the system for a user.

2.1.3 User database module

The user database is of course used for data storage. This module will only communicate with the software module using the SQL language. It will store important information from the users and the house.
This module will serve as the systems long-term memory. Without this, intelligent behaviour would be non existent.

2.1.4 Hardware application module

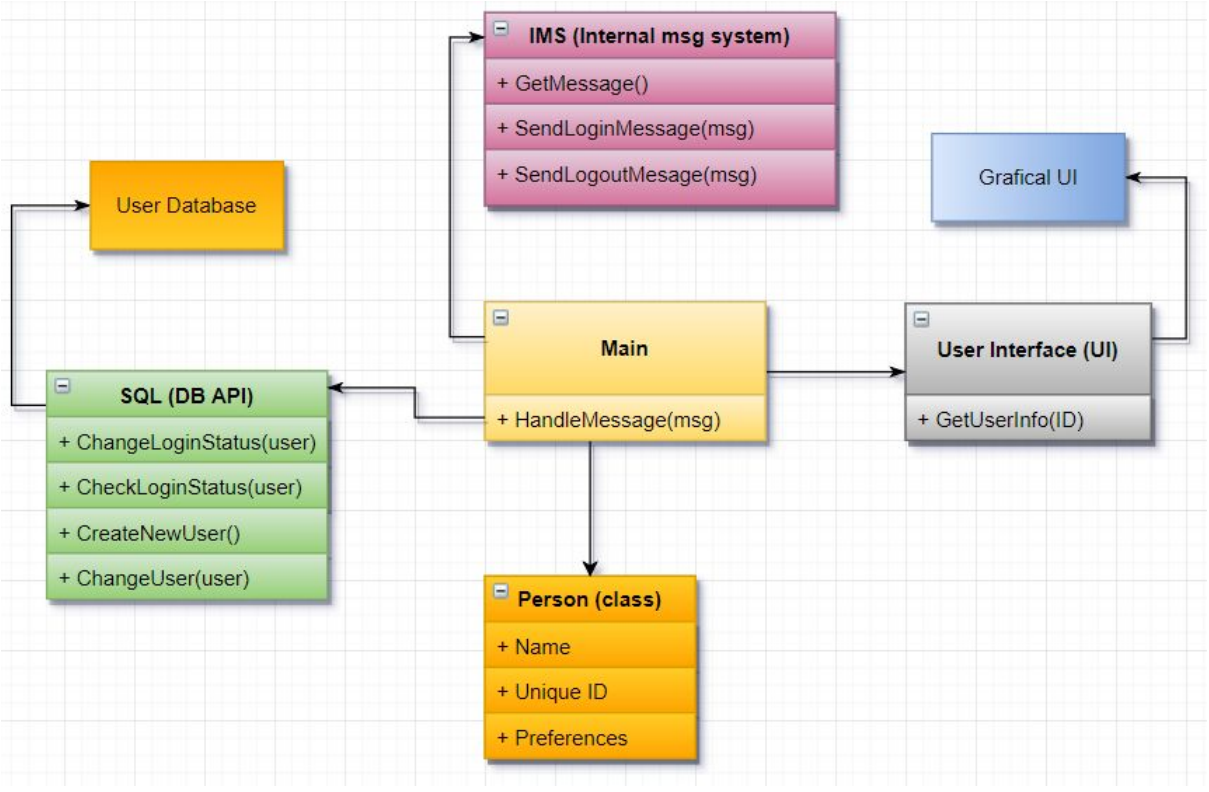
The hardware application module is a more passive module, serving as a collection of applications and devices one would place around a smart-home.
This module will mostly only communicate and be controlled by the hardware controller module. Due to its passive nature, few functions are made for this module. It will also serve as the hardware UI for simple communication with users. It is also used as the sensors for the system, and is easily expandable by introducing additional sensors, which will be explored upon in later chapters.

This module serves as the limbs and sensors of the system, and without it, any meaningful interaction with the real world would be impossible.

2.2 Internal communication

For the systems internal communication, called the Internal Message System or IMS, we designed a character based message system using a serial communication. The details of this system is given in the “System specification” chapter, as this section will only serve as an overview of IMS.
The IMS essentially serves the part of simplifying the communication between the software module and the hardware controller module. This communication is in its core just serial communication, which is difficult to use when building a complex system with multiple commands and data to be sent.
To build this system, we defined some generalized messages which would be used for communication by the main modules. This made the creation of our main system much easier, without needing to add new specific messages during the project, which could make it hard to complete.
As commented in the “Related work“ chapter, our message system is based on X10’s message system, which made it easier to structurize in having a previous system to study and build upon.

3 System Specification



Module level architecture

Implementation details, internal structure, description of main components, their structure and functions; user interface (if there is one); examples of use; etc

3.1 IMS communication protocol

This chapter will serve as the specification of the IMS.

3.1.1 Message types

This section will explain the different message types that are defined for the IMS. The specific values in each message is explained in the next section.

Login

The login message is a command used by the software module. This commands give the hardware controller module a room, and optionally some preferences. These preferences refers to how the room should be adjusted, such as lighting and heating. If a preference type is not present in the room, the value will be ignored. If a preference is not given, it will be adjusted to a default value.

Logout

This is equivalent to the login message, except from a visual confirmation of the logout is given. The preference sent is the users “logout preference”.

Mode change

This message type is used for being able to change modes at the arduino, e.g. turning of the RFID. Currently, no modes has been introduced.

ID reading

This message is given from the hardware module to the software. This message will most typically lead to a login or logout of a user from a room, or the creation of a new user

Acknowledge message

When sending messages using the IMS, we have the option of requesting an acknowledge message in return after the message has been successfully read and translated. This message may also inform that the previous message was read successfully, but not managed to be executed.

Login	_	len	len	Msgtype	Room	Type pref	Value pref	...	ack
Logout	_	len	len	Msgtype	Room	Type pref	Value pref	...	ack
Mode change	_	len	len	Msgtype	New Mode	ack			
ID-reading	_	len	len	Msgtype	Room	ID value (MSD)	...	ID value (LSD)	ack
Ack-msg	_	len	len	Msgtype	Success /failure				

IMS structure

3.1.2 Message structure

As shown in the illustration, the different messages has both some common and uncommon fields. These fields are the data and metadata of the message. All messages will always have a starting character field, length fields and type enum field.

Starting character

This field is a reserved character which is used for the message module to recognize the beginning of a message. This is because of the difficulties using serial communication, with potential noise and partly losses of messages.
In the current version of the system the starting char is the underscore, '_'.

Length

The length fields is used for the message module to know the number of characters to read from a message. This will fix problems in reading too long messages, and knowing if a message did not properly send.
The length field are always of equal length, and will use the decimal system for the digits.

Message type enum

This field is used for the message module to know how to translate the message. The table for the different types are given below. A detailed description for each type is given later in this chapter.
Message Types:
login = 0
logout = 1
mode_change = 2
id_reading = 3
ack_message = 4

Room values

This value will always have the same position in the message, as shown in the illustration above. Some messages does not contain this value, and this is translated by the message type value.
The different enums for the rooms are given in the table below.
Room:
Kitchen = 0
Livingroom = 1

Preference value type

This is metadata that dictates which preference the preference value refers to. These values will always arrive in pairs. Below is the table for the preference value types
Preference:
Lighting = 0
Volume = 1
Heating = 2

Preference value

The preference types will always be sent in pairs with a preference value. The preference value is set by the user upon user creation or edited later.
These values are sent as HEX, and will be interpreted with the power of 10.
Value range of the preferences:
Lighting [0-100%]
Volume [0-100%]
Heating [0-45 C] (Still sent between 0-100%, where 100% = 45 C)
e.g. [..,1, 5,..] would be translated as a music volume preference of 50%.

ID field

The ID field is the received ID by the RFID scanner. The ID field is not given a specific length, and will be calculated by the message volume using the total length and the constant length of the rest of the message.

Acknowledge field

This field is set to 1 if the message or command send needs a confirmation of receiving or execution of command.

Success/failure

This field is only used in the acknowledge message, and will be 1 if the previous command was successful and 0 if otherwise. Placement shown in the illustration above.

3.2 Software Module

This section will specify the different sub modules in the software module. This section will only list the most relevant methods of each sub module. A more functional explanation of these modules are found in the “System Description” chapter.

3.2.1 Main

This contains the main cycle of the software system, for the IMS and UI. This cycle will repeatedly check after new messages in the IMS using the SerialConnector. If the message is legit, a specific message object will be return to the main. These message objects will be described in the IMS module. Based on the current mode, Normal or Create user, the system will use the function handle_message(msg) to act according to the message.
One may also use the GUI to change these modes.

Normal mode will check if the message contains an ID. If it contains an ID, the system will check if this ID is logged in in one of the rooms using the DBconnection. If it is logged into the same room as the message was sent from, the main will send a logout message to the arduino, using the class method send_login_message(msg_str) from the SerialConnector class, with information of the room and logout preferences. If the user is not logged into any rooms, the system will send a login message with room information and login preferences in the same manner as logout. If the user is logged in another room, the user will first be logged out of this room, and logged in to the new room.
When this is done, the DB will be updated accordingly

Create User mode will take the id from the message, and start creating a new user using the UI module method get_user_info(msg).

3.2.2 DB connection module

This sub module uses class based methods to communicate with the user database using SQL. The database contains 3 tables:

User table:
This table contains the basic information of the user, and an unique ID which is the same as the users RFID tag.

Preferences:
This table contains the preferences of an user. The table use the users unique id as a foreign key to refer to a specific user. The number of preferences are scalable

Rooms:
This table contains login information of each room in the house, such as who is the “owner” of the room, that is who logged in previously.

3.2.3 UI

This sub module is the user interface of the software system. It contains methods that will directly communicate with the user to obtain information.
It also contains its own graphical module, GUI.

3.2.4 Internal message Module

This sub module is the communication module for sending and receiving messages from the hardware controller module. It uses the IMS communication protocol, as described in the “IMS communication protocol” chapter. The module has some predefined message classes to have a more systematic and easier handling of these messages. The module translates the message, and parses it to the correct message object.
An example of a message class and its properties is given below.

class IDmessage:
#properties

- name
- length
- type
- user_id
- current_room
- needs_ack (Is True if this message requires an acknowledge message)

3.3 Hardware:

This chapter will describe the hardware module.

3.3.1 Functionalities on Breadboard:

Our Arduino are connected to a breadboard. The breadboard contains a RFID reader, a piezo, 3 LEDs and a temperature sensor. All these are connected with the computer through the message system. With these parts we are able to feed the software with input from the RFID reader and the temperature sensor, and then receive commands such that the LEDs are set with the correct outputs. We have also connected a piezo to the breadboard to play a sound when the user is logged in and out. This will lead to a better interface for the user.

Passive RFID reader

The RFID reader used in this project is a passive RFID reader (RC522). A passive RFID reader works the following way:
The RFID read-write device emit electromagnetic waves, and this forms an electromagnetic field. When the passive RFID tag enters this field RFID tag gets energy from the field and the microchip circuit is activated. The biggest drawback of this reader is that the induction

range is very close[7]. However, this can also be an advantage for security. Eavesdropping will be considerable more difficult to achieve when the range is close.

Music

We wanted to simulate the possibility to play music in our smart house. Since we concluded that the piezo would make a rather annoying music simulator, specially while testing, we chose to just send a message back to the software module, informing what kind of preferences would be set if the correct hardware was in place, this information will be printed on screen.

In current edition the only outputs are “music playing” and the wanted volume. Added functionality could be to have different playlists for different rooms and the possibility to change songs and volume dynamic with a remote control.

Heating

Every house should have a heating system. Heating consumes a lot of power, and an effective regulation would be an advantage for the environment and personal economy. Our heating system is controlled by a temperature sensor, if the sensed temperature is warmer than the preferred temperature, the software module will send a message to the air conditioning to increase the fan speed and set the preferred temperature

An upgrade could be a sensor in the window detecting if the window are open. If so, the air conditioning be turned off until the window is closed. This way you avoid unnecessary power consumption.

Lightning

Perhaps the most important part in a house is lighting. In our small smart home we only have 3 lamps. Two of these lamps are located in the living room, and the last one is in the kitchen. We wanted to have an illusion of light intensity in our system, we solved this by define the lamps in the living room the following way; one light will be lit if intensity is 50% or less. Higher intensity will lit both lights. This is a very easy solution, but we concluded that with our basic tools, this describes the wanted functionality well.

Photoresistor

In the kitchen we add a photoresistor to our system. This way we could manage to have a dynamic control of the intensity of the LEDs by how much light there is in the room, and this way save power.

The idea is to have a way to lower the power consumption, but also guarantee for the wanted light intensity in the room. The way to obtain this is to have a sensor that

[7]Product Description of RC522 module
https://www.aliexpress.com/item/RFID-13-56MHZ-passive-RFID-reader-RC522-module-STC89C52/32813926337.html
15.05.2018

continuously register the light in the room and sends this to the arduino. The arduino has gotten a preferred light intensity from the software module. The photoresistor will change intensity of the kitchen lamp when the light in the room changes. When there is daylight, the lamp will have lower intensity. The maximum intensity of the lamp is set by the user's preference, and will be scaled accordingly.

3.3.2 Arduino setup:

RFID reader connection

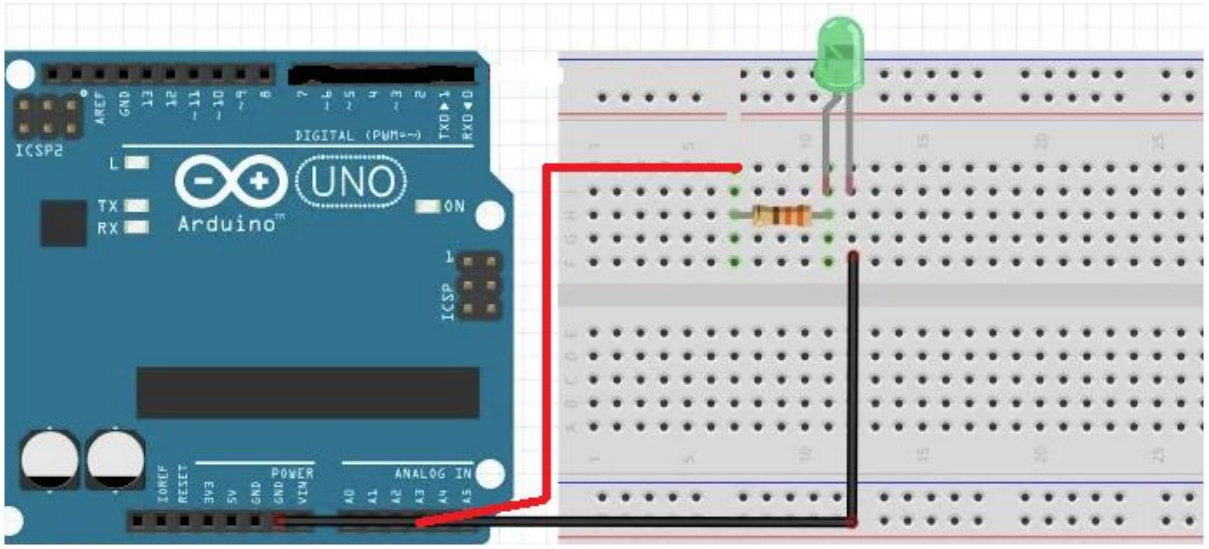
The pin configuration for the **RFID-RC522** reader

RFID-RC522 PIN	**ARDUINO UNO**
SDA	10
SCK	13
MOSI	11
MISO	12
IRQ	UNUSED
GND	GND
RST	9
3.3V	3.3V

LED connections

Lamps in rooms and their connection to the arduino board.

Room/PIN arduino	PIN 5~	PIN A3	PIN A5
Kitchen	Red led	-	-
Living room	-	Green led	Yellow led

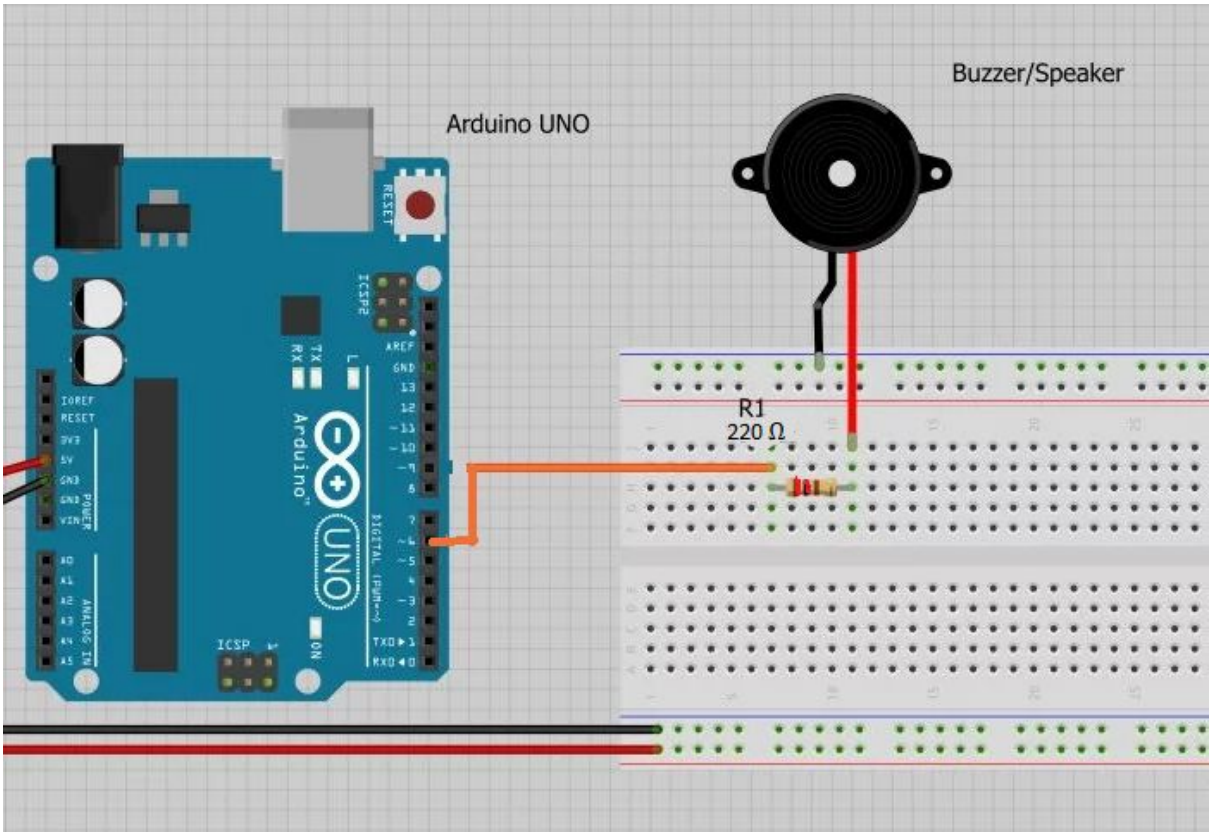


[Source:http://www.instructables.com/id/Arduino-Blinking-LED/ with minor fixes 14.05.2018]

Each led is connected with a 220 Ω resistor as shown in the figure. The pin configuration can be read in the table over.

Piezo connections

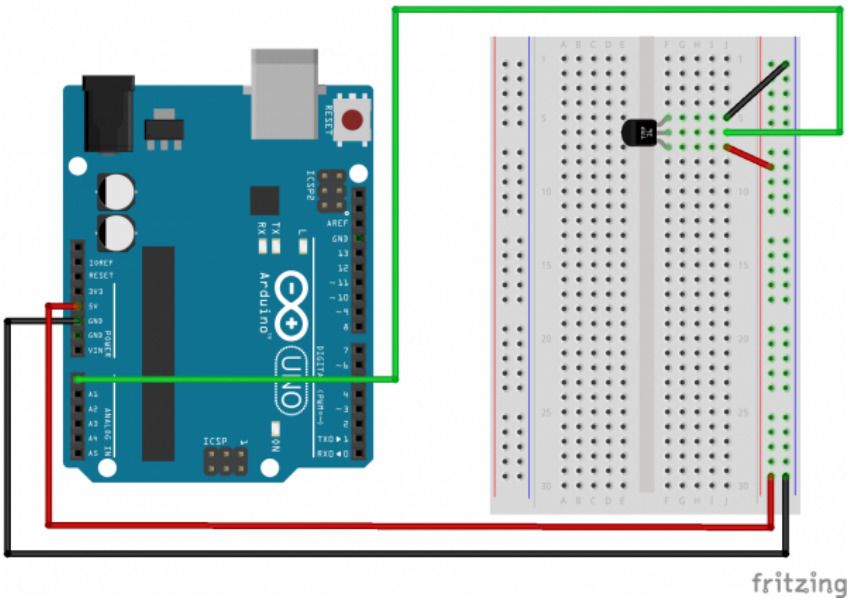
The only purpose for the speaker is to make a more intriguing interface for the user. The following connections were made:



[Source:http://www.instructables.com/id/How-to-use-a-Buzzer-Arduino-Tutorial/ with minor fixes 14.05.2018]

Temperature sensor connection:

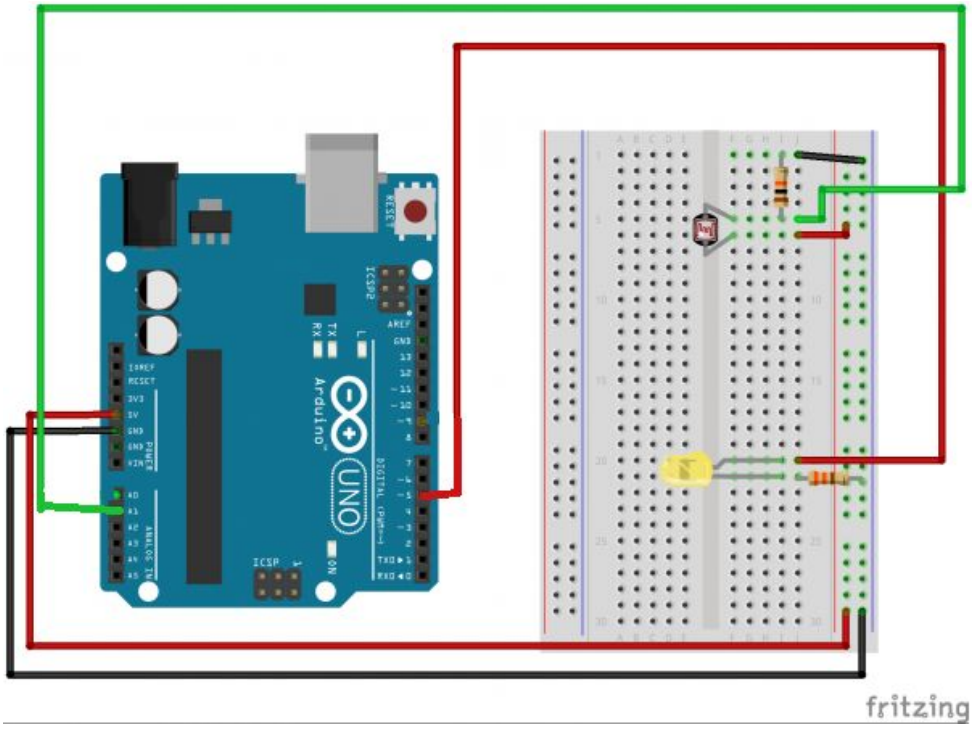
The circuit for reading the temperature sensor



[Source:https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32/experiment-7-reading-a-temperature-sensor 15.05.2018]

Photoresistor connection with LED

The circuit used to obtain dynamic adjustment of light



[Source: https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32/experiment-6-reading-a-photoresistor with minor fixes15.05.2018]

3.3.3 Commands:

`set_lamps(int room,int intensity), set_music(int room, int pref_value)` and `set_radiator(int room, int pref_value)` are possible to scale up. This is because of the function's general form. Under this paragraph, there is a part of the function that handles lamps. This can be scaled up by adding a parameter `int floor` and manually set new lamps with wanted functionality. In the part *Produced Input and Output (under section: Lightning)*, we discuss our choice of intensity and lamps.

```
void set_lamps(int room,int intensity){
  if(room == living_room){
    if (0 < intensity &&  intensity <= 50){
      digitalWrite(LAMP2,1);
     digitalWrite(LAMP3,0);
      }
    else if (100 >= intensity && intensity > 50){
     digitalWrite(LAMP2,1);
     digitalWrite(LAMP3,1);
      }
    else if (intensity == 0){
     digitalWrite(LAMP2,0);
     digitalWrite(LAMP3,0);
      }
    else{
     Serial.println("Intensity out of bound");
      }
  }
}
....
};
```

3.3.4 Message handling:

Reading a message:

The function `listen_msg()` listen on serial com, if it detects a message starting with underscore(_) the message is read. `listen_msg()` returns the message as a string. The message is then sent to `execute_msg(msg)` as an argument. This function decodes the message and checks the `msg_type`, `room`, `pref_type` and `pref_value`. If `msg_type` is identified as login or logout, the function `adjust_room(int room, int pref_type, int pref_value)` are called. This function sets the desired preferences from the message by using the set-functions.

Sending a message:

The main loop are always trying to detect a RFID tag. When a RFID tag are detected, its ID is sent to software by using the function `send_msg_id(String RFID_id, int room)`. Since we only have one RFID reader we have to change the room manually. To simulate music and heating we are also sending messages to software. By using the function `send_msg(int room, String input, int msg_type)`, you will be able to send a message to software informing it that it should print music/heating preferences on screen.

3.5 Example of use

Below is given an example use case:
“Sigurd wakes up 7 am. He starts by logging into the bathroom, using his unique smart card. The lighting slowly rises to his exact preference, with some soothing music from the radio. The heating in the floor is burning warm, just as he likes it.
The he goes to the kitchen, and logs in. He doesn't need to log out of the bathroom, the system automatically fixes this when he logs into the kitchen.
The sun rises in the window during his breakfast, and the system slightly dims the light to account for the natural light.
He logs into the system, and changes some of his preferences. He thinks the music is a bit to low, so he increases the volume by 10%. The change is saved, and the change is immediate in the house.
He walks out the main door, and logs out. His logout preferences is having dimmed lights in the hallway, because he is afraid of burglary. The rest of the house is dark, and the heating is minimal so he won’t freeze when he returns..”

4 Evaluation

4.1 Hardware

It was hard to make modules in arduino. All functions had to be made in main script. This was an okay solution and worked well for the purpose of our project. However, in a larger system this would be a serious drawback. Working with Arduino was overall a good experience. A lot of code examples are available including circuit sketches and data sheets.

Serial communication would be a challenge in a larger system, however, if you have a great message handler this can still be a plausible solution. The message handler in the hardware has to be upgraded, and a module structure would probably be the best solution.

4.2 Software

Because of a long process of preparation and software designing, the implementation of the system went smoothly. The software has been equipped with failsafes, so no frequent crashes apperd. The current IMS is currently not easy to expand rapidly upon, because of the limitations of the message handler in the main module.
The DB is fine and scalable, but a more generalized class for these functions might be easier if the system where to be expanded.

To evaluate the intelligence of the system, let's look at the goals. We wanted the system to make the user save both electricity and money, and be intuitive to use. The system will most definitely help he user to save electricity, without any denial of usage, which will save the user money.The simplicity on the other hand is not quite there. Having to carry around your special card in the house might be cumbersome.
All in all, we feel that this system fulfill most of the established goals.

5 Further expansion

The system can be expanded by adding sensors, components and devices. The message system and the database are easily scalable making the expansion easier. Instead of a PC a Raspberry pie should control the software module and be the computer in the system. If the arduino is not capable of directly controlling a device a control system for this device has to be added.

To fix the issues in expanding of the message handler, which was discussed in the "Evaluation" chapter, a rework of the method will be required. A good solution is to make a specified class to handle messages, with a concrete protocol.
The systems intelligent behaviour could also be improved, by introducing features such as scheduling and room specific preferences. This would not be hard to expand upon in the current system.

6 Conclusion

When testing the system we were especially satisfied with the message system. We also managed to implement several energy efficient solutions in our project, such as the photoresistor. The structure of the database module is also a construction we are satisfied with and worked well in our system.
We found the system to be both intuitive and intelligent in a satisfying way. We also recognize that a few expansions in the system would help the intelligent functionality to increase.
If we managed to implement this system in an actual home, with control of both applications and sensors, we would have a proper smart home.