Brunel University

School of Engineering and Design

Final year project

# Smart system for control and customisation of office buildings energy consumption

*Author:*
Dionysios Kalantzis

*Supervisor:*
Dr. Maysam Abbot

April 10, 2012

## Abstract

*"Apollo the god who sees and foresees everything."*

This project describes *Apollo*: a smart system that provides sensing features such as temperature, light, gas detection. *Apollo*'s novel approach is based on the use of heterogeneous wireless communication (802.15.4) technology, which allows to communicate with various hardware platforms. Furthermore, it can operate the lighting and/or air-conditioning settings according to the room occupant's preferences to achieve optimum working environment conditions. By using the building's energy resources on demand, *Apollo* reduces wastes ultimately minimising energy consumption. Additionally, the computation for all the above was moved from a dedicated machine to a central wireless micro-controller, reducing the cost and the installation complexity. This report describes the design choices and the implementation steps of the system. Actual sensor recordings and measurements are presented to evaluate the overall performance of *Apollo*. In conclusion, possible improvements of the system are suggested.

**Acknowledgements**

I would like to thank Dr. Maysam Abbot for his supervision and advice on how to approach and successfully complete this project.

Many thanks to Dr. Ioannis Chatzigiannakis, Orestis Akribopoulos, Vasileios Georgitzikis and Dimitrios Amaxilatis for their guidance and technical support.

Special thanks go to Miss Maria Kanellopoulou and Mr Francesco Giovannini for their encouragement and moral support. Additionally I would like to acknowledge all my friends for their support.

Finally I would like to extend my heartfelt gratitude to Konstantina Karapapa and Andreas Kalantzis, to whom I dedicate this work.

# Contents

# List of Figures

# Chapter 1

# Introduction

Air is one of life's essentials for all creatures on Earth. Humans live, work and prosper in air. The quality of air affects humans performance. Throughout the years various methods have been devised to assure the good air quality. For example miners were using canary birds as warning systems until 1987. The presence of toxic gases such as carbon monoxide, methane or carbon dioxide could have had deadly consequences for them. Therefore the notice of any distress from the canary birds would act as a warning. Studies have shown that low air quality standards affects the performance of the people in a working environment [**?**]. Therefore it necessary to keep high air quality standards, which will improve people's working performance as well as their general well-being.

Another great issue in today's fast growing world is power consumption and how to minimise it. More and more devices are being developed and are becoming available to consumers. Regardless if these devices are mechanical or electrical, they need energy to operate. Meaningless use of energy increases unnecessary costs and affects the global pollution.

Innovation in technology nowadays has made us take certain services for granted. Such services include the Internet, which can be used as a powerful tool to monitor and act on life quality standards. In addition people are able to operate their home and office appliances with the simple pressing of a few buttons. Wireless Sensor Network is a distributed network of sensors that accommodate sensing features of environmental changes which can be reported all over the world via the Internet. The recent years have witnessed an explosion in the number of system applied relying upon WSN.

## 1.1 Project Aims and Objectives

This project presents a wireless system devised to improve the air quality in office buildings and other closed working environments. The novel approach of this system relies in the use of a low-power wireless standard protocol – *IEEE 802.15.4* – capable of communicating with heterogeneous systems. *IEEE 802.15.4* is the default standard for low-rate, low-power wireless communication. In addition, it is used by various wireless network embedded platforms such as SunSPOT, iSense, XBee, TelosB, ZigBee and more. This system also introduces the ability of the afore mentioned platforms to communicate between each otherl.

Additionally the devised solution can also control and minimise energy consumption by using the building's resources only whenever they are needed. This is achieved by constantly monitoring the environmental conditions in a room and operating the lighting and air-conditioning according to the presence and the preferences of the occupants. The energy is therefore used only upon demand and energy wastes are minimised – lights switched on and forgotten about, constantly running air conditioning, etc.

A smart system, such as the one presented in this report, can prove to be an effective solution even to complicated building constructions as it relies on wireless connectivity for communication. There is no need for structural building modifications in order to install and run the system. Moreover the system's network can easily be expanded because of its inherent heterogeneous ability to communicate with various platforms. Finally, its simple installation procedure and ease of use make it an outstanding solution.

## 1.2 Implementation Steps

For the project implementation the following steps were completed:

- A detailed project plan was presented. Amendments were carried out to accommodate for additions and modifications.
- A thorough research on the components was carried out. Additionally, an in depth understanding of the software tools, which would be used, was acquired.
- An algorithm was devised for the overall system operation
- The circuit boards were designed and built

- The Arduino control board was programmed to achieve the desired functions
- The system was tested to find and detect errors
- Sensors measurements were collected to evaluate the overall system performance

# Chapter 2

# Literature review

## 2.1 Energy efficiency

A lot of attention has been focused on the energy use of new buildings through regulations and various standards. These just address only part of the problem as almost 70% of the buildings which will be needed by 2050 have already been built. This becomes even more dramatic as such buildings have been constructed before 1985, the year during which the energy efficiency requirements in the Building Regulations were first introduced [?, ?].

Modifying an existing building can be very expensive and extremely inconvenient, in particular if it is currently being used. Therefore it seems only reasonable to invest in new technologies that could reduce the energy consumption without altering existing constructions. *Apollo* is a plug and play 3 piece system which can control the energy use of the room it is installed in.

## 2.2 Ventilation and air quality standards

Clean air is very important for a working environment. Productivity may be reduced by up to 20%, as studies has shown. The pollution, which both humans and machines produce can make the air feel heavy and humid. This is due to the fact that both continuously produce vapours and heat. The result of operating in this environment could be tiredness and discomfort from the increased $CO2$ and temperature levels. Studies has shown that in order to maintain a good indoor climate in an office, the air should be replaced at least three times per hour. In many rooms, such as laboratories, cooling is also essential if just the ventilation is not enough [?].

## 2.3 Building Management System

Building Management System (*BMS*) is a computer base control system for monitoring and operating mechanical and electrical equipment. BMS made its appearance in the early 1970's with the introduction of complex electronic devices that could collect and store data for managing services such as lighting and heating. BMS systems have seen great development since their appearance. They used to involve big electromechanical systems controlling the buildings. Nowadays BMS is a software application that collects data from all over the building and such data is then presented on a computer screen. BMS systems functionality varies, depending on the demands and the building owner's investment. For example, a BMS system may control the lighting, heating, ventilation and air conditioning for the whole building or for each floor separately, or even by splitting the floor into sections and so on. The downside of BMS is the high cost and the mandatory prerequisite of having to include its installation at the time the building is constructed. The cost also increases with the depth of systems specialisation regarding the level of section control [?].

## 2.4 Internet of Things

The *Internet of Things* (IoT) is the concept of globally interconnected devices – *things* using RFID *(Radio-Frequency IDentification)* technology with virtual representation in an Internet-like structure. Today's rapid technology developments have rendered possible the interconnection of heterogeneous devices as *things*. Once connected to the Internet these devices can report in real time from anywhere on the globe. Moreover due to their uniqueness, it is easy to map where the data are coming from and most importantly what they are reporting [?].

IoT vision is not limited to just reporting. More and more processes and services are slowly making use of the global infrastructure provided by IoT. Such processes and services could have environmental and, ultimately, an economic impact. In particular, *Apollo* could clearly be included into IoT, as it monitors environmental changes and by acting accordingly energy is saved, effectively contributing to cost reduction in terms of annual consumption [?].

An example of such IoT project is Pachube:

> Pachube ("patch-bay") connects people to devices, applications, and the Internet of Things. As a web-based service built to manage the world's real-time data, Pachube gives people the power to share, collaborate, and make use of information generated from the world around them. [?]

Moreover *Pachube* is a platform where sensors report in real time and their data are accessible anywhere on the world. This is again due to the fact that *Pachube* traffics its data over the Internet.

## 2.5 Wireless Sensor Network

Wireless sensor network (*WSN*) is a concept that makes IoT vision more and more real. The first appearance of the term dates back to the 1990s.

> Researcher named Kris Pister dreamed up a wild future in which people would sprinkle the Earth with countless tiny sensors, no larger than grains of rice. [?]

These particles were called *"smart-dust"* [?]. WSN is a network consisting of distributed devices that provide sensing features. Lately, an increasing number of systems relying on WSN are being developed. Arduino, which is one of the most common hardware platforms, would make the perfect wireless sensor network node, due to its small size (Arduino Pro Mini), low cost and modularity.

A unique experiment is currently being proposed by *Smart Santander*. A city-scale experiment for applications and services of a smart city. *Apollo* could easily be adapted into this experiment, providing information on the energy consumption of the office buildings and not only. This way, the city's resources use would be optimised, achieving greater energy efficiency.

This project was conducted in collaboration with the Computer Technology Institute and Press "Diophantus". The Institution is based in Patra, Greece.

> The Computer Technology Institute and Press "Diophantus" is a research and technology organization focusing on research and development in Information and Communication Technologies (ICT) [?].

The main reason for this collaboration is the common interest on the subject as well as the technical expertise provided by the Institution regarding WSN and the wireless communication protocol used.

# Chapter 3

# System Architecture

*Apollo* is an automatic system which detects the environmental conditions in a room or office. As a smart system it reacts, according to the preferences of the users in the office, to achieve the optimum environment to maximise the work performance.

In Figure 3.1 below there is a visual representation of *Apollo*. The communication between the board is shown with the use of the arrows.

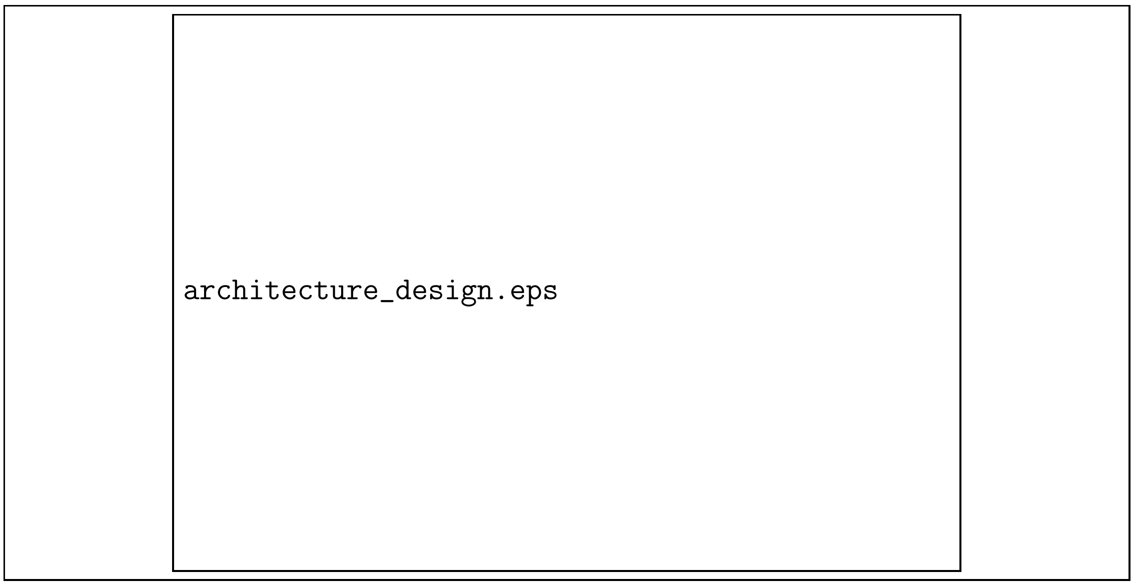

Figure 3.1: Board communication - Block diagram

The *Controller* is a SunSPOT (Sun Small Programmable Object Technology) – a wireless sensor network mote developed by Sun Microsystems [?]. Due to is computational power it is preferred over the Arduino to make the decisions regarding the operation of the lights and/or the air-conditioning.

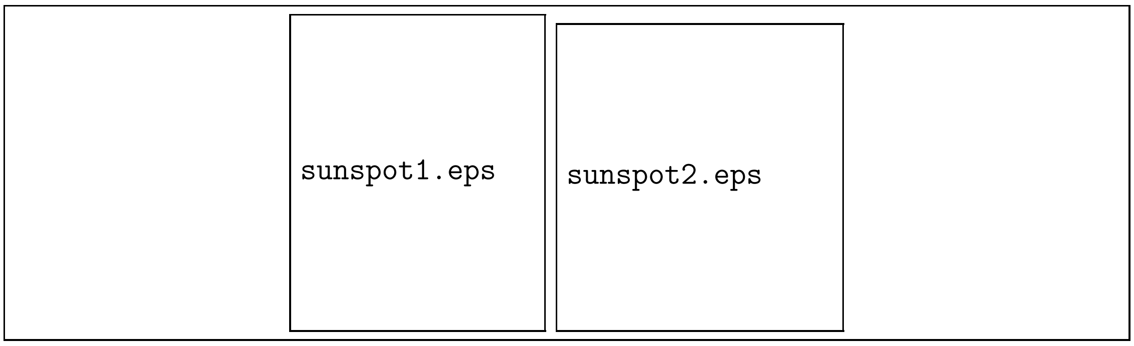

Figure 3.2: Sun SPOT

The system works in two stages – the *passive* process and the *active* process. In order for the system to work on the later stage it is necessary to go through the *passive*. The difference between these two stages is that during the former (*passive*) stage the system collects the user(s) preferences regarding the environmental changes. It also observes the environmental values when the user(s) choose to switch on the lights and/or the air conditioning. Those markers will set the threshold of when the system will operate on the *active* stage. During this stage (*active*) the system operates the lights and the air conditioning.

As the weather conditions change over the year it is advised to use the system on the *passive* stage approximately for one to two weeks every four months. This is to calibrate the system with reference to the four different seasons (*spring, summer, autumn, winter*) over the year. In general this is needed to be done only once after the installation of the system. From then onwards the *passive* stage is needed to be run only when the users feel the need to make any core modifications of the thresholds.

Regarding the modifications, user(s) are also able to use the web interface of the system to manually control the lighting and/or air conditioning. The web interface is a medium where user(s) can remotely access and interact with the system. This could mean operating manually the lighting and/or air conditioning remotely. The user(s) are also capable to check the status of the system via the web interface.

The status of the system is generally used for security reasons as *Apollo* can inform the owner of the office of any unusual changes in environmental conditions. This is useful for example to detect fires. Moreover every time the system recognises an unauthorised person using the office can remotely inform its owner through the internet with an email.

Figure 3.3 gives an overall description of *Apollo*. The different wireless sensor system nodes can communicate between each other in a heterogeneous network. The nodes feed their data into the database via a Controller (SunSPOT). *Apollo* can also receive instructions from the Controller to operate the lighting and/or air conditioning of the room it is installed in.

`syst-arch.eps`

Figure 3.3: System Architecture

# Chapter 4

# System Specifications

*Apollo* is a three (3) piece system:

- ► The sensor board
- ► The control board
- ► The action board

The main reason for the existence of the 3 different pieces is to minimise the mean time of repair. Each board can easily removed and individually checked for defect(s).

## 4.1 The sensor board

The sensor board is a PCB(*Printed Circuit Board*) responsible for collecting the environmental changes of the space it is installed in. To achieve the above, the board is equipped with the following:

- ► Gas sensors:
  - ▷ MQ - 4 Methane - *Compressed Natural Gas* sensor
  - ▷ MQ - 7 Carbon Monoxide sensor
- ► Light sensor:
  - ▷ LDR - Light Dependent Resistor
- ► Motion Sensor:
  - ▷ PIR - Passive InfraRed motion sensor
- ► Temperature Sensor:
  - ▷ LM335 - Precision Temperature Sensor

### 4.1.1 MQ - 4 Methane (Compressed Natural Gas) sensor

This is an easy to use *compressed natural gas* sensor capable of sensing natural gas, which is mainly composed of methane ($CH_4$). The detection range for this sensor is between 200 to 10000ppm of methane - compressed natural gas [?].

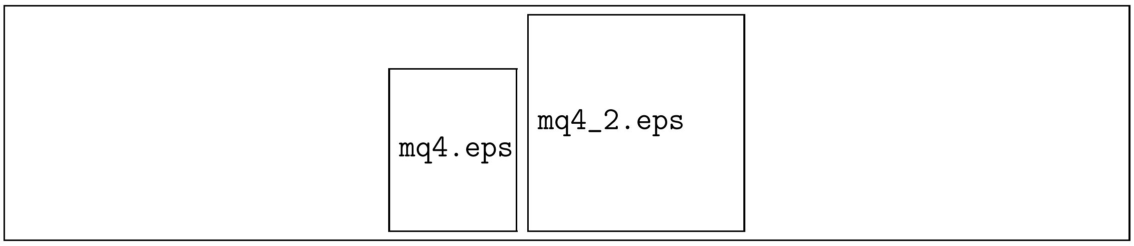

Figure 4.1: MQ - 4 Methane - Compressed Natural Gas sensor

From the following diagram it is easy to see the simplicity of sensor's drive circuit.

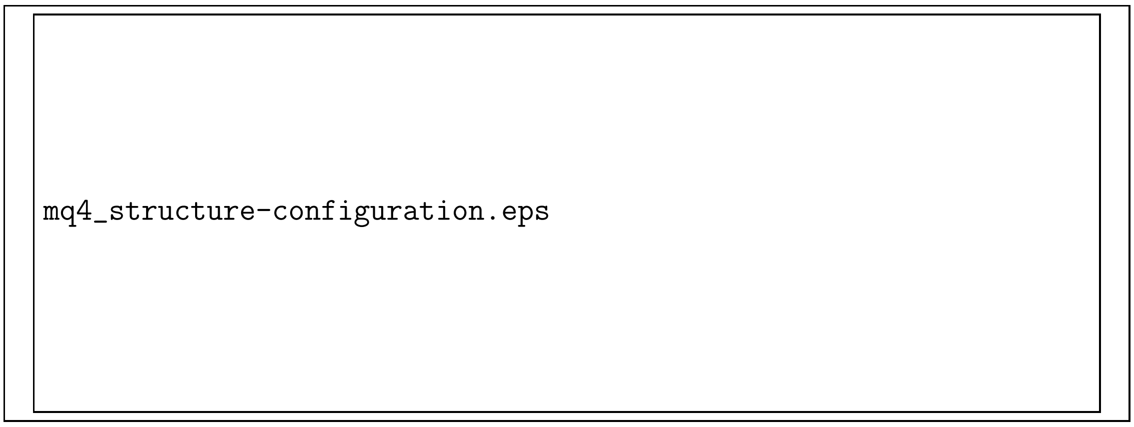

Figure 4.2: MQ - 4 sensor's drive circuit

The sensor requires an input of 5 Volt and its output is an analogue resistance. The output is inversely proportional to the gas concentration. This means the greater the gas concentration is the lower the output value will be.

### 4.1.2 MQ - 7 Carbon Monoxide sensor

The *Carbon Monoxide* sensor is as well an easy to use sensor. The detection range is between 20 to 2000ppm of $CO$ [?].

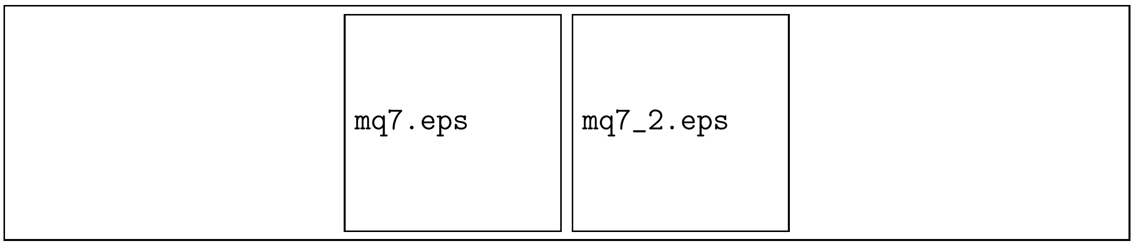

Figure 4.4: MQ - 7 Dimensions

The MQ - 7 has a slightly more complicated operation technique comparing to the MQ - 4.It detects the concentration of carbon monoxide by cycles of 5 Volts and 1.5 Volts. Specifically when at sensor is applied 5 Volts, the internal resistance heats up and cleans any other gases absorbed under the cycle of 1.5 Volts, when the internal resistance is at low temperature. The conductivity of the sensor increases as along with the gas concentration [**?**].

### 4.1.3 LDR - Light Dependent Resistor

This is the simplest light sensor. It is a variable resistance of a range between 10 M(Ω) down to 100 Ω.

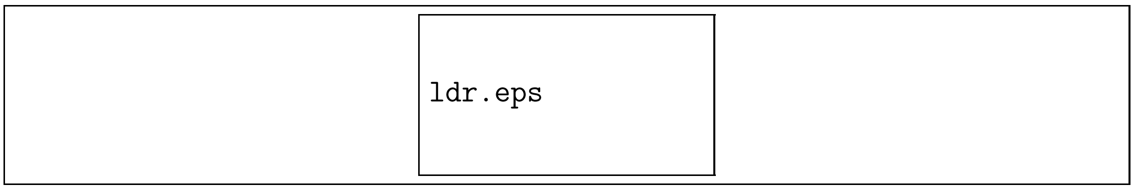

Figure 4.5: LDR - Light Dependent Resistor

The LDR, in dark condition, has a very high resistance allowing to a small amount of current to pass. As the light (*lux*) increases, the resistance decreases proportionally.

### 4.1.4 PIR - Passive InfraRed motion sensor

PIR *Passive InfraRed* motion sensor is a ready to use integrated circuit. It can detect movement up to approximately 6 meter.

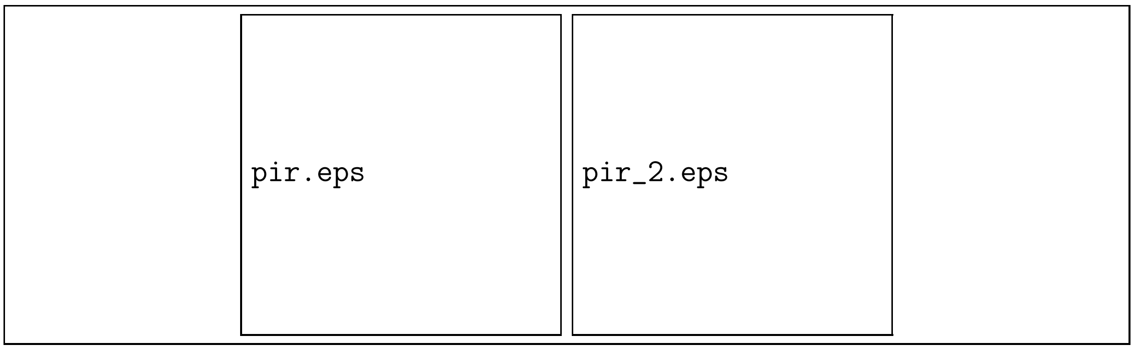

Figure 4.6: PIR - Passive InfraRed motion sensor

The sensor has a 3-pin connection interface. The red wire is the power (5 Volts), the brown wire is the ground ($GND$) and finally the black wire is the output. The alarm pin (output) is an open collector and therefore the introduction of a pull up resistor is essential. In a no movement state the sensor outputs a high signal (5 Volts). When on movement detection state the alarm pin goes low (0 Volt). The open drain setup allows multiple motion sensors to be connected on a single line. When one of the sensors detects movement the alarm pin will be pulled low.

### 4.1.5 LM335 - Precision Temperature Sensor

The LM335A is an analog temperature sensor. The LM335A works like a Zener diode with a breakdown voltage proportional to absolute temperature at 10mV/°K. When a resistor is connected between the 5V and the GND the LM335A will output an analog voltage of 2.98V (298 Kelvin is 25°$C$). The output of the sensor is linear and when calibrated at 25°$C$ the LM335A has typically less than 1°$C$ error over a 100C temperature range. The sensor can operate continuously from 40°$C$ to 100°$C$.

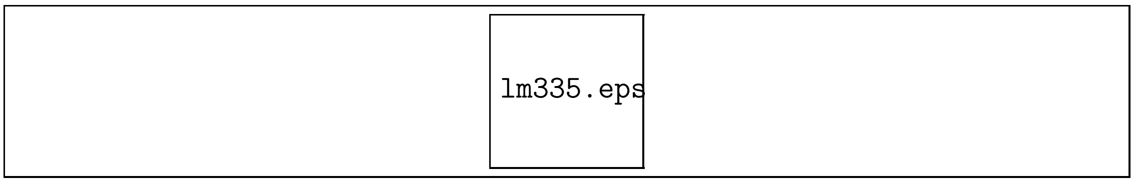

Figure 4.7: LM335 - Precision Temperature Sensor

## 4.2 The control board

The control board is the core of the *Apollo*. It is a PCB board which is equipped with:

▷ Arduino Pro Mini

▷ XBee RF Module

The board is connected with the sensor board via a pin cable. The sensor collects the data which are then prepared for transmission and sent. This is done with the use of the XBee RF Module.

### 4.2.1 Arduino Pro Mini

Arduino is an open-source platform based on a micro-controller and a simple software development environment. It can be used to build interactive applications, receiving inputs from several types of sensors and interacting with the environment controlling various actuators.

Arduino Pro Mini board is based on the ATMEL AVR 8-bit micro-controller, specifically on the ATmega168. Despite its size, it is equipped with both analogue as well digital input/outputs. More specifically it has 14 digital input/output pins, of which 6 can be used as PWM (*Pulse-width modulation*) outputs. In addition it has 6 analogue inputs, as well as a reset button. Each pin is a hole allowing pin headers to be mounted (soldered).

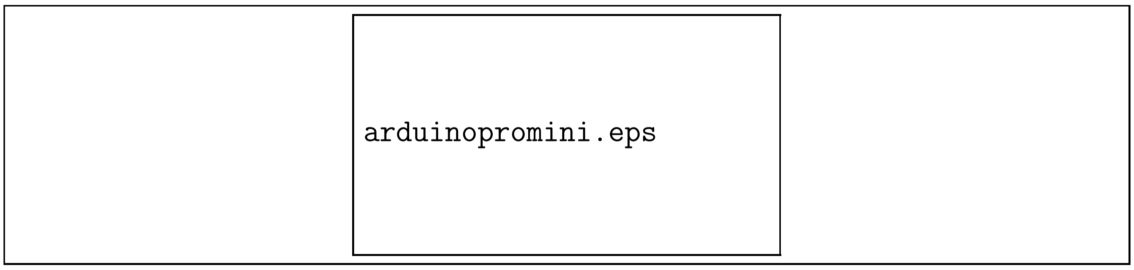

Figure 4.8: Arduino Pro Mini

There are two versions of the Arduino Pro Mini. One, which runs at 3.3 Volts and 8 MHz and another which runs at 5 Volts and 16 MHz. The former one is used for *Apollo*. Arduino Pro Mini has three different memories embedded on it; one

**Flash** of 16 MB, one **SRAM** of 1 KB and one **EEPROM** 512 bytes.

It can be powered up by either USB connection or a external power supply. The recommended input voltage is between 5 to 12 V.
Below are the Arduino Pro Mini specifications:

| | |
|---|---|
| Micro-controller | ATmega168 |
| Operating Voltage | 5 Volts |
| Input Voltage | 5-12 Volts |
| Digital I/O Pins | 14 (6 provide PWM output) |
| Analogue Pins | 6 |
| DC Current per I/O Pin | $40mA$ |
| Flash Memory | 16 KB (2 KB dedicated to the bootloader) |
| SRAM | 1 KB |
| EEPROM | 512 bytes |
| Clock Speed | 16 MHz |

Table 4.1: Arduino Pro Mini Specifications

### 4.2.2 XBee RF Module

Arduino does not provide built-in wireless connectivity. Therefore the wireless communication is achieved by connecting an IEEE 802.15.4 module, the XBee RF.

XBee RF is an embedded module providing wireless connectivity to other devices. The module uses the IEEE 802.15.4 standards for a low-rate wireless PAN (*Personal Area Network*) [?]. This makes it ideal for a low-cost, low-power wireless sensor network, such as *Apollo*.

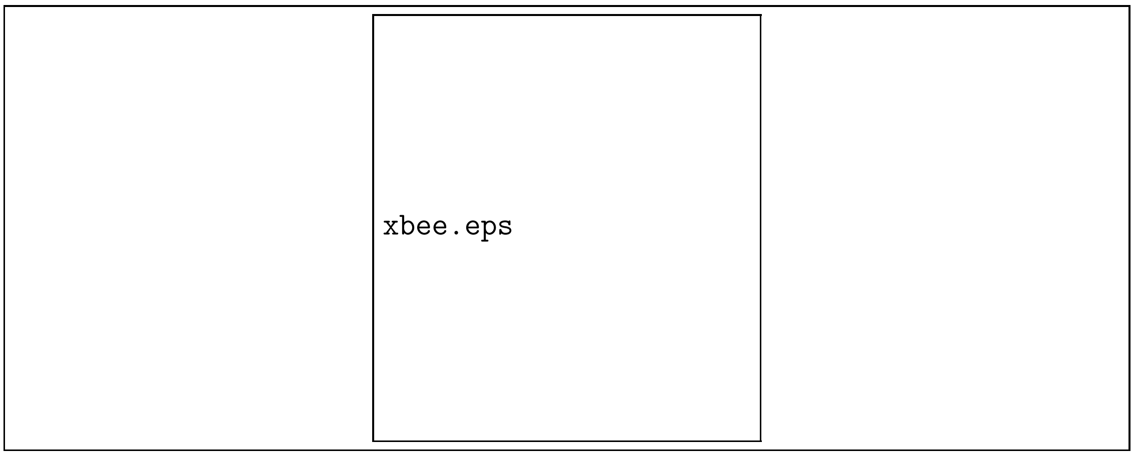

Figure 4.9: XBee RF module

XBee is connected to the Arduino via a serial port and adds wireless capabilities by forwarding any data received from the Arduino to the wireless network and forwarding received wireless to the Arduino respectively.

## 4.3 The action board

The action board is a PCB board equipped with holes to fit up to 4 relays. The relays used are fully sealed SPDT (*Single Pole Double Throw*).

### 4.3.1 Relay SPDT Sealed

This is a high quality SPDT sealed relay. Suitable for high voltage and high current devices. [**?**]

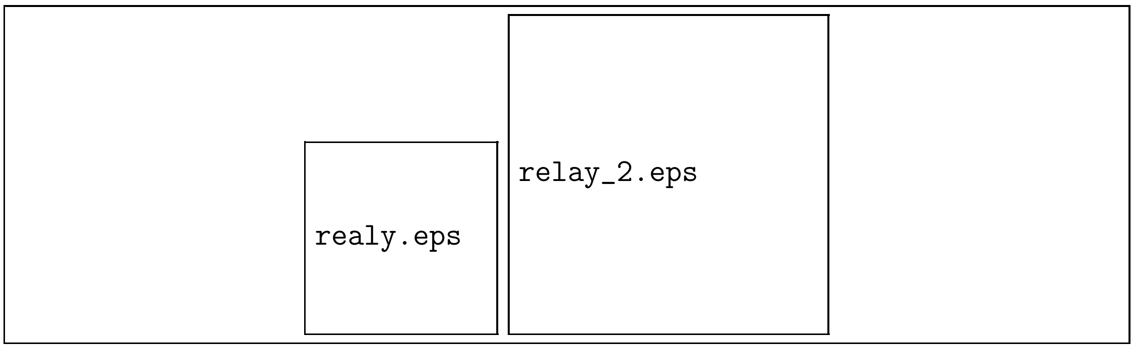

Figure 4.10: Relay SPDT Sealed

Below are the SPDT specifications:

| | |
|---|---|
| Max. Operating Current | 5 A |
| Max. Operating Voltage | 220 V AC / 30 V DC |
| Max. Operating Power | 1250 V AC / 150 W |
| Contact resistance | 100m$\Omega$ |
| Ambient Temperature | -40 $\sim$ +85°C |
| Mechanical life | 10000000 Ops |
| Electrical life | 100000 Ops |

Table 4.2: SPDT Specifications

### 4.3.2 LEDs

The action board is also equipped with **LEDs** (*light emitting diodes*) which are connected to the relays. This provides a visual and fast inspection for defect(s) on the board.

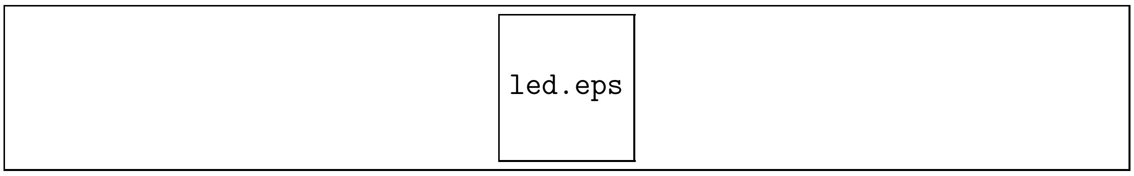

Figure 4.11: LEDs

## 4.4 Software

### 4.4.1 Arduino IDE

**Arduino IDE** is a software development environment for Arduino platform program applications. It comprises a toolbar, which includes the most common functions, a text editor for writing the code and a message area. Arduino IDE is also used to upload programs and communicate with the Arduino boards [?]. Figure 4.12 below shows a screenshot of the IDE:

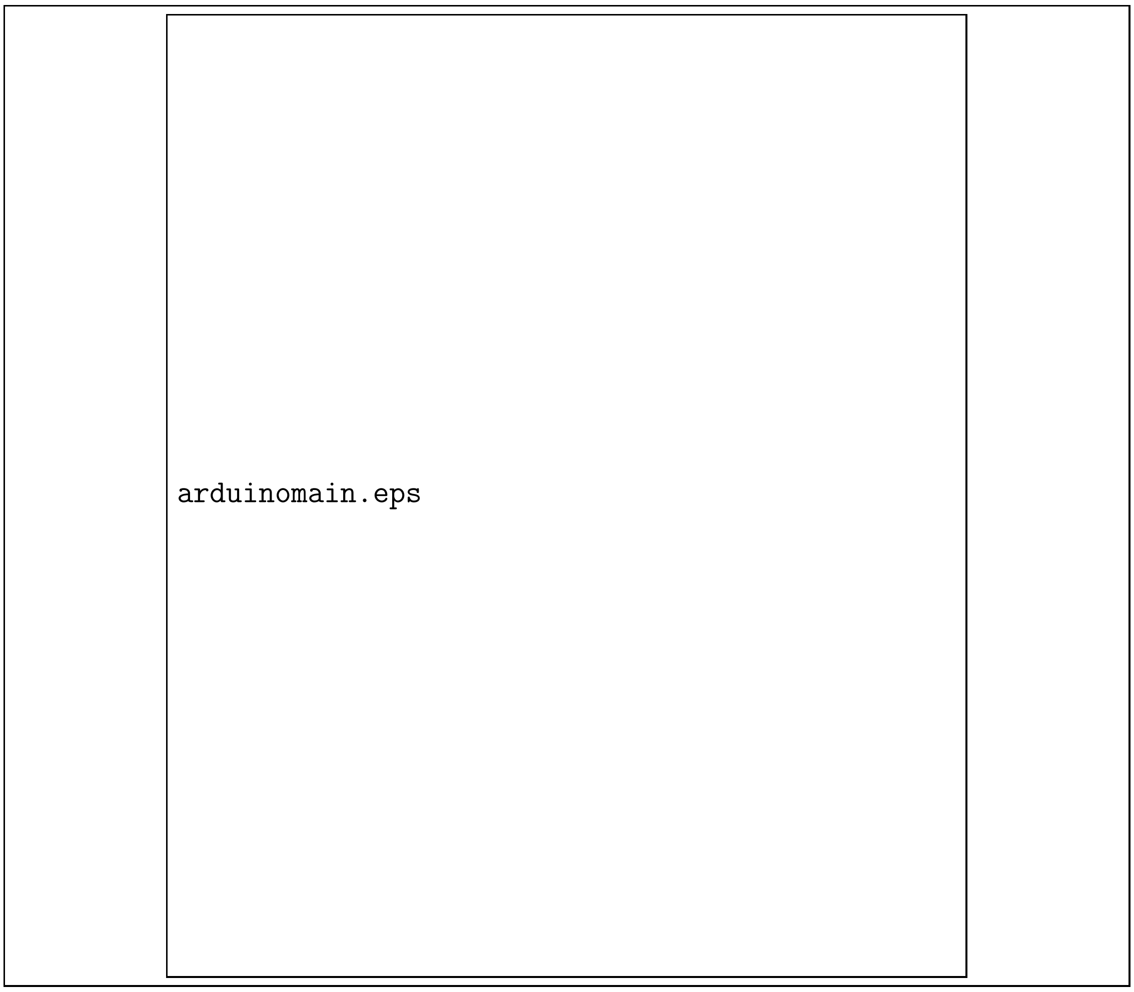

Figure 4.12: Arduino IDE software development environment

The toolbar buttons allow the user to verify programs, create new, open existing, save, upload to the Arduino board and the serial monitor. The programs written using Arduino Software are called **Sketches**. The serial monitor displays serial data being sent from the Arduino board it also can send data to the board. The message area is used to display errors and to provide feedback at the process of saving and uploading the programs on the board. The console displays text output by the Arduino environment including full error messages [**?**].

Sketches are written in C/C++ and require two methods to be defined:

- ▷ **setup():** runs only once at the beginning of the program and initialises the variables
- ▷ **loop():** runs repeatedly inside the program until the power of the board is cut

Beside these two basic methods Arduino IDE includes others which can be split in **structure**, **values** and **functions** [?]. These are defined below:

- ▷ **Structure:** includes all the control flow structures of numerical and logical operations, such as: **if, if...else, and more**,
- ▷ **Values:** include all the data structure, such as: **boolean, char, int, byte, long, and more**,
- ▷ **Functions:** include all the management methods for the pins and the general internal function of the micro-controller, such as: **pinMode(), digitalWrite(), digitalRead(), analogRead(), tone(), interrupts(), and more**.

### 4.4.2 CoolTerm

**CoolTerm** is a software for communication with hardware connected to serial ports. It is a serial port terminal application for the exchange of data with hardware such as micro-controllers. It relies the same concept as the serial monitor of the Arduino IDE. The main differences that made it more suitable for this project are the capability of multiple concurrent connections if multiple serial port connections are presented. It can also display the received data in plain text as well as hexadecimal format thus rendering it more human-readable. In addition, it can capture the received data into an output text file [?].

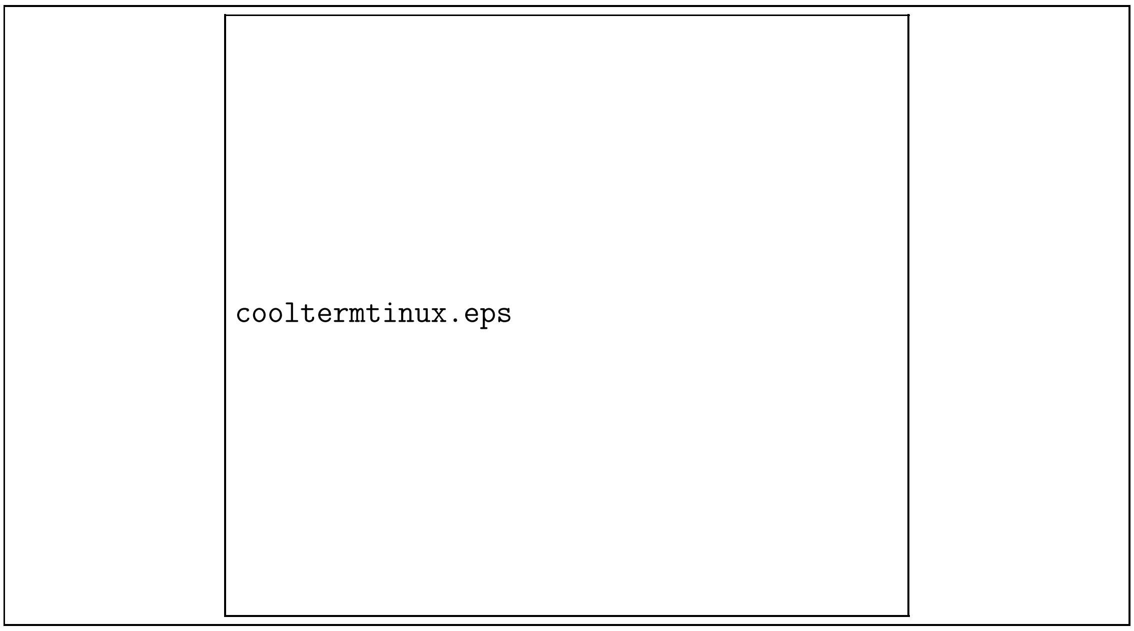

Figure 4.13: CoolTerm - serial port communication

### 4.4.3 Eagle PCB Software

Eagle PCB Software is a powerful and flexible PCB design program. Eagle is an acronym for Easily Applicable Graphical Layout Editor. It was used to design and print the PCB designs for this project (*Apollo*). To do this the schematic design of the three different boards first had to be drawn. The board design was then produced by arranging the components appropriately [?].

Figure 4.15: Eagle - Board design

## 4.5 Database

The database is called *Überdust*. It stores data using the WiseDB component of the WiseML library based on Hibernate. Hibernate is a well established relational persistence framework for Java. Hibernate facilitates the storage and retrieval of Java domain objects via Object/Relational Mapping (ORM). Although *Überdust* was originally developed for storing sensor data from wireless sensor networks, any source can provide data as long as it is in a simple timestamp-based format and a small set of metadata is already provided in the persistence store. *Überdust* was provided by the Computer Technology Institute and Press "Diophantus". [?]

`Uberdust.eps`

Figure 4.17: Uberdust

# Chapter 5

# System Design

This section presents the overall design of the system as well as of the individual *Apollo* components. In addition, the design choices are explained and justified in detail and their advantages are analysed in depth.

The main guides for designing *Apollo* were the air quality standards for office and laboratory buildings. In addition the installation of the system should have the minimum impact on the building's structure. The maintenance time and cost of all three boards was another major factor for their design. Moreover the system's capacity for heterogeneous connection and interaction with other systems was one of the crucial characteristics for its design.

*Apollo* passed various stages before taking the shape and form that it has to date. The original approach included only two boards: ***control*** and ***sensor*** board. After a series of prototype testing it seemed cogent to break the *control* board in two discrete ones. The outcome of this was the three boards: ***sensor, control, action***.

The significant difference between the two approaches is the separation of the Arduino platform from the control board with the relays. The reason for this design choice was the minimisation of the repair ($MTTR$). This on the other hand increased the overall cost of the system. The cost increase is of approximately 1, according to the School technician. It is not possible to calculate the exact cost increase, as for the development of the PCBs the facilities of the School were used.

For the design of the boards, CadSoft Eagle CAD software was used. Eagle is equipped with an Electrical Rule Check *ERC* as well as with a Design Rule Check *DRC* mechanism. These are integrated functions that check the schematics for electrical and design errors respectively. If any error(s) are detected, a dialogue window pops up with details on the errors.

## 5.1 Sensor board

### 5.1.1 Schematic implementation

The design process of the sensor board (using Eagle CAD software) is presented below.
The development begins with the creation of the schematic in order to place all the necessary components as shown in Figure 5.1.

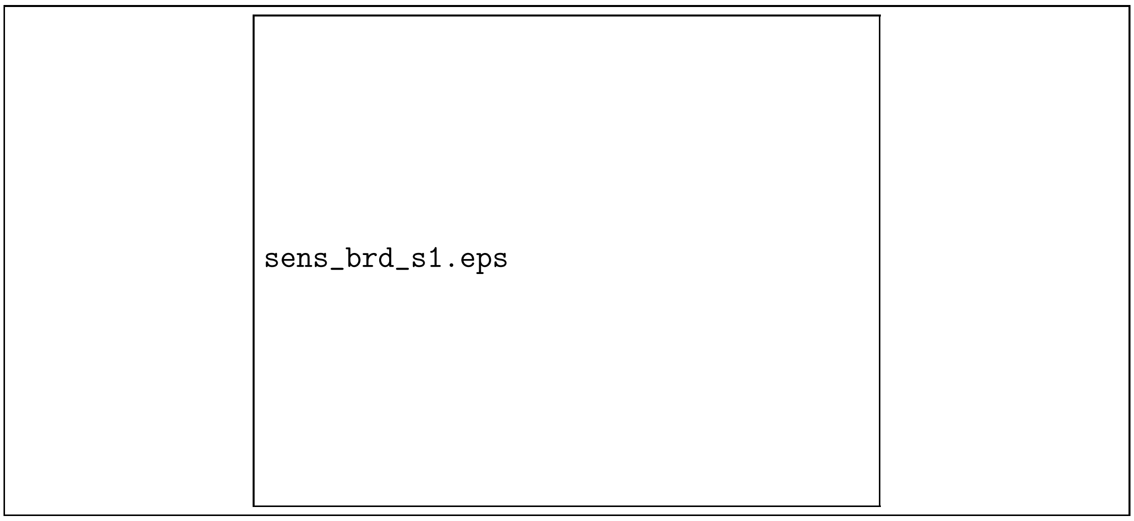

Figure 5.1: Sensor board - Adding all the parts (Stage 1)

Each part is sensibly named to allow easy identification of the separate components. This is shown in Figure 5.2.

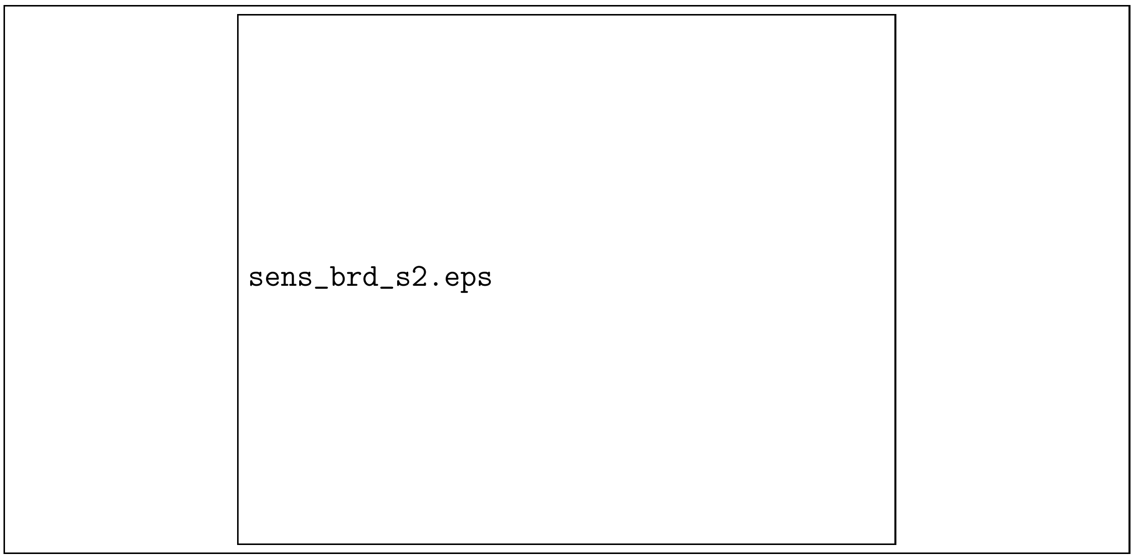

Figure 5.2: Sensor board - (Stage 2)

At this stage the parts are being spread out on the board. The reason for this is for the following step to be easier and more efficient.

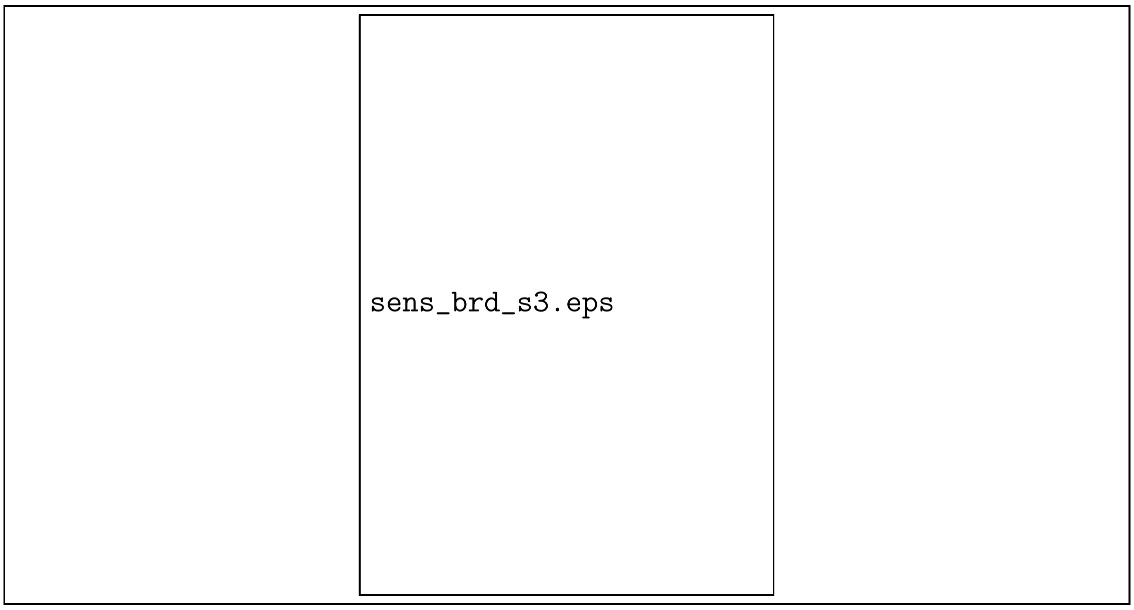

Figure 5.3: Sensor board - (Stage 3)

In the following Figure 5.4 the wiring of the parts of the sensor board is being traced.

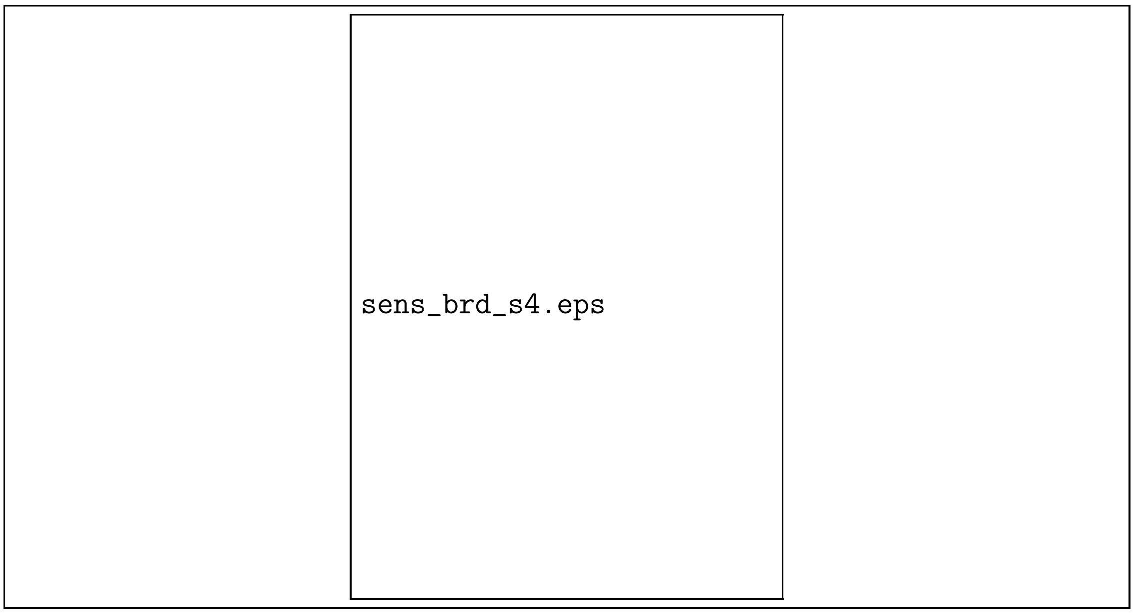

Figure 5.4: Sensor board - (Stage 4)

To check if the wiring is properly done, and there are neither short-circuits nor unconnected parts an **ERC** was performed. The result is shown in Figure 5.5. No errors were reported back. The warnings are of somewhat minimum importance as the parts used are universal. On the other hand it is crucial for all pins to be connected either to other parts and/or to the power supply and/or to the ground.

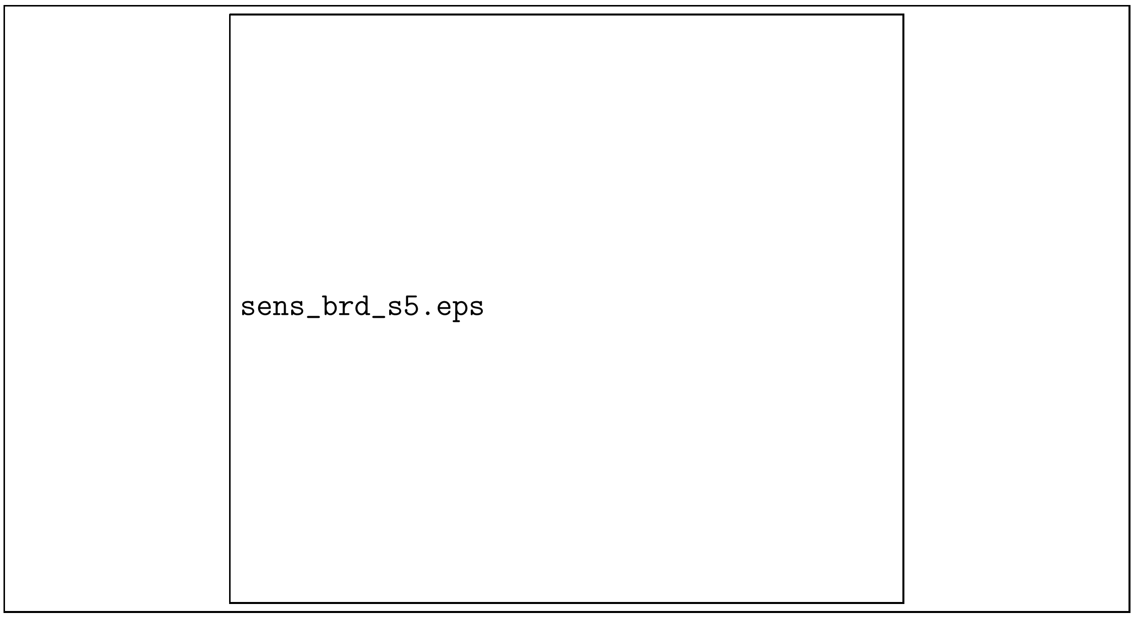

Figure 5.5: Sensor board - (Stage 5)

### 5.1.2 Board implementation

This is the stage in which the parts implemented in the previous step will be placed on the board. As the board will be installed in a visible part of a room, it is important for it to be as discrete as possible. In Figure 5.6, below, the components are ready to be spreaded in the available space. The final size of the board will be 5cm wide and 5cm high.

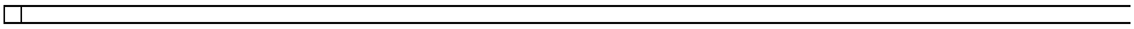
Figure 5.6: Sensor board - (Stage 6)

Once all components were spread with sufficient space between them, the *Ratsnest* tool was used to minimise the path of the of the wiring. This time the wiring is the conductive medium of electric current between the parts of the board – the copper.

Figure 5.7: Sensor board - (Stage 7)

The wiring routes were finally generated with the use of the *Autoroute* tool. To achieve an optimal result – shortest copper paths – *Autoroute* has an *Optimiser* option which was used. Moreover the settings of the *Optimiser* were set accordingly to achieve the least possible layers and bias.

Figure 5.8: Sensor board - (Stage 8)

Layers are the sides of the board used for the creating the copper paths. In Figure 5.8 the top layer wire paths are marked with maroon color and the bottom layer paths are blue. Biases are the points where the wire path changes layer. Biases add extra difficulty when soldering the board and therefore should being avoided when possible.

## 5.2 Control board

The same method as with *Sensor* was used for designing and developing the *Control* board. The following Figures 5.9 5.10 5.11 5.12 show the process.

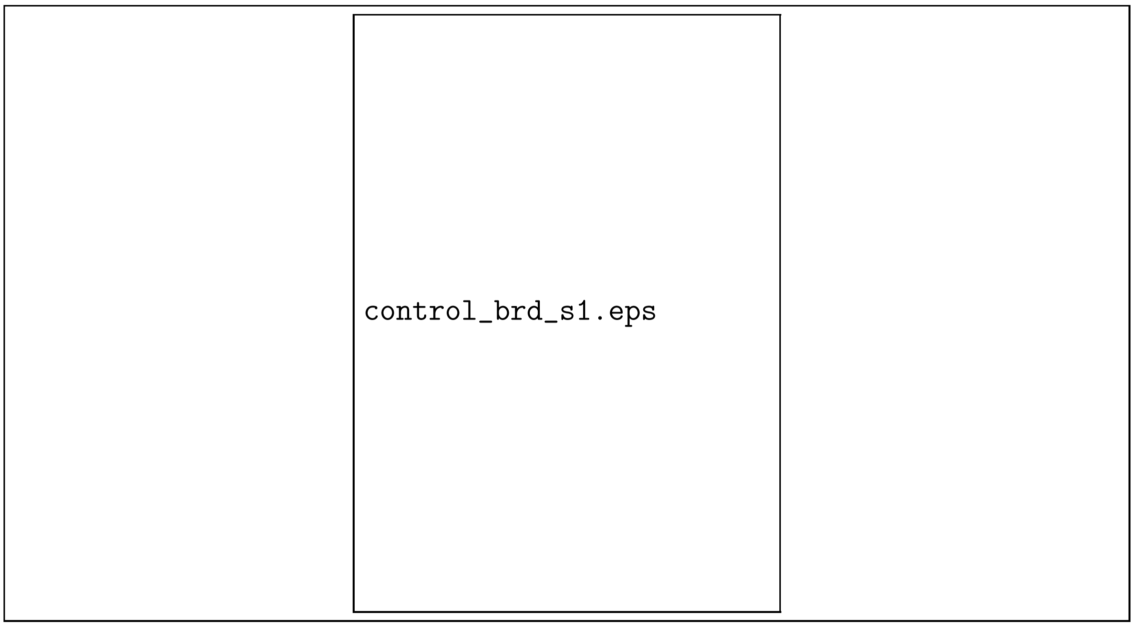

Figure 5.9: Control board - (Stage 1)

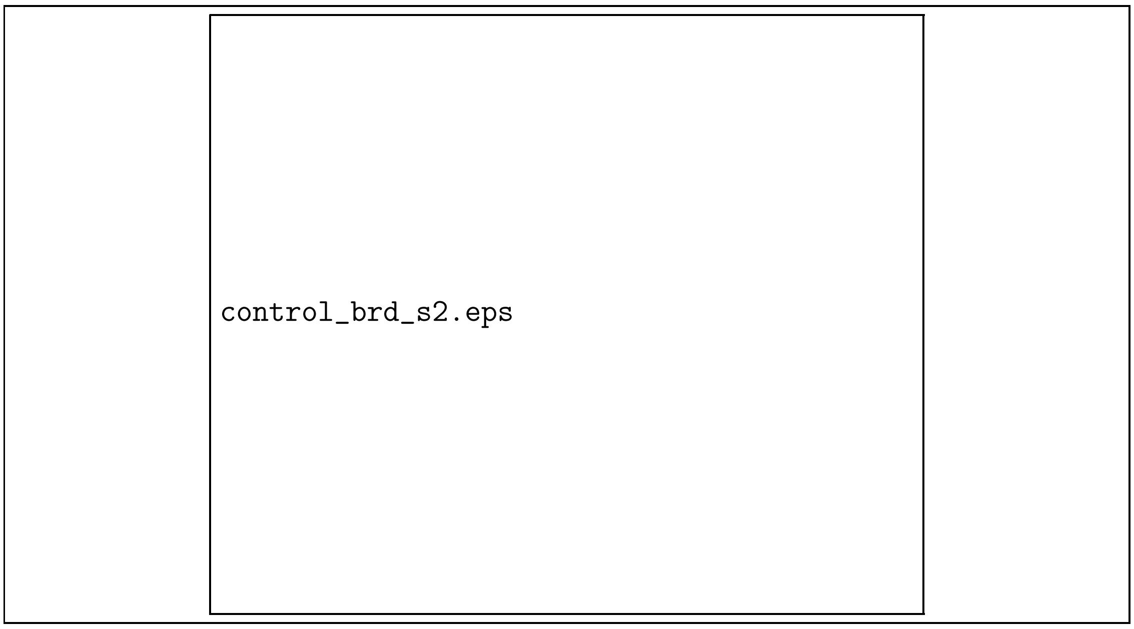

Figure 5.10: Control board - (Stage 2)

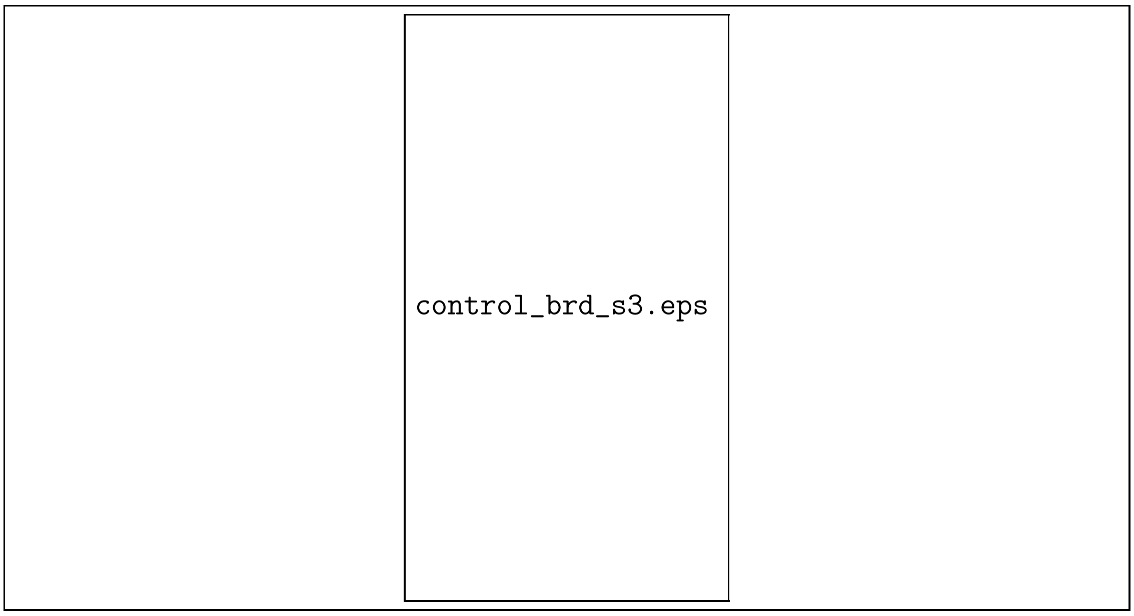

Figure 5.11: Control board - (Stage 3)

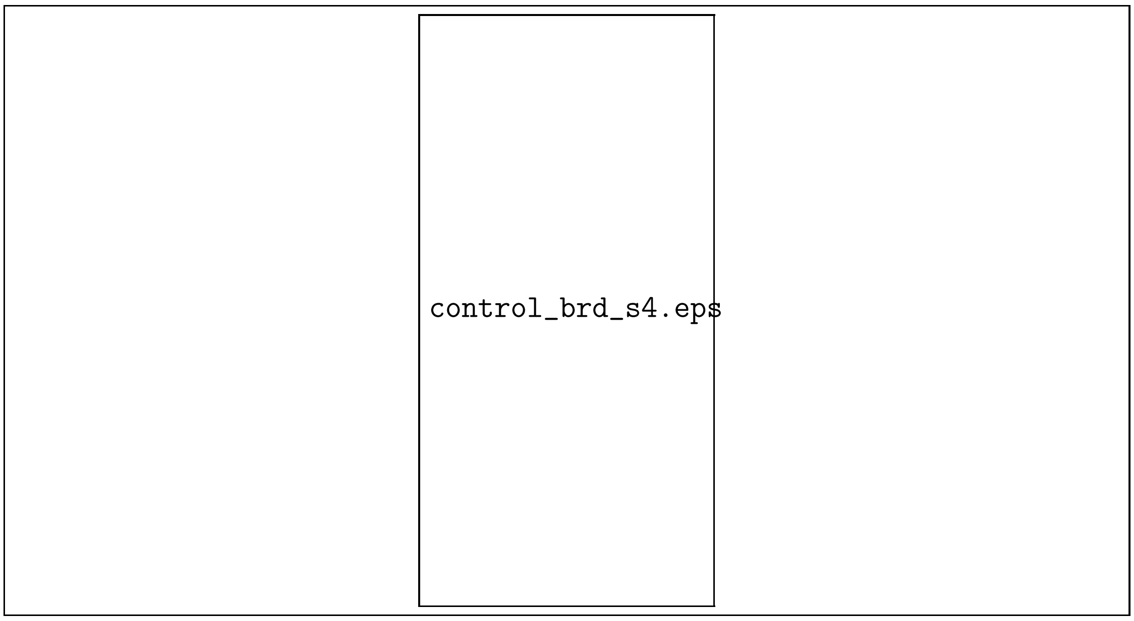

Figure 5.12: Control board - (Stage 4)

At this stage is essential to clarify the existence of the six pin holes (*JP8*) in the above Figure 5.12. These will host a six pin header (Figure 5.13). This is one of many practical ways to hold the *XBee* shield straight. With this method the shield's pins will not be bended, therefore any damage will be avoided.

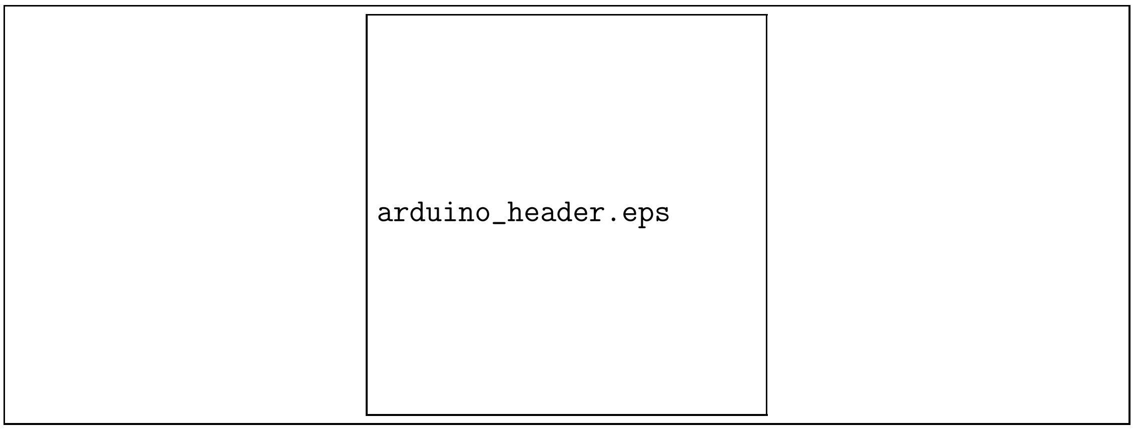

Figure 5.13: Arduino header

## 5.3 Action board

The same method as for the *Control* board was used in the development of the *Action* board. At this point, it is important to explain the existence of the four *LEDs* on the board. Their purpose is to have a quick visual check of the signal going into the relay. In this way, it is possible to detect if one or more of the relays have failed. With this technique, precious time is saved as the*action* board does not have to be removed.

Figure 5.14: Action board - (Stage 1)

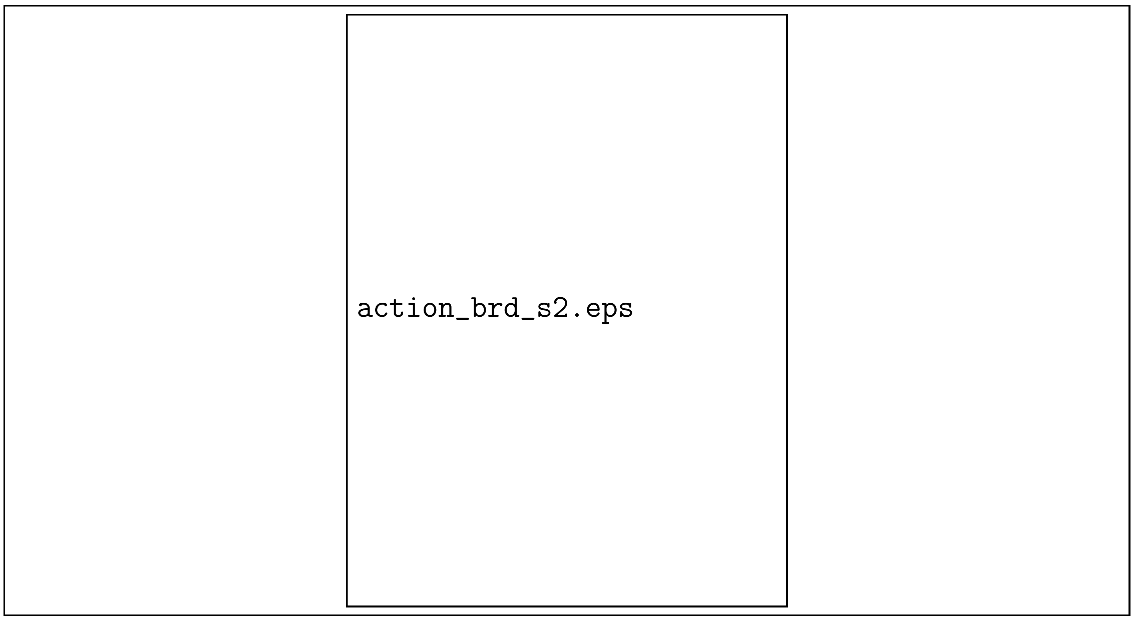

Figure 5.15: Action board - (Stage 2)

Figure 5.16: Action board - (Stage 3)

## 5.4 Arduino code

In this section an extensive description of the code that was created for *Apollo* will be presented. The code is divided in three parts; ***initialisation, setup(), loop()***

### 5.4.1 Initialisation

The very first part of the code consist in the *initialisation* procedures. These usually comprise the inclusion of the libraries as well as the declaration of the global variables which will be used across all the program.

The heterogeneous communication protocol which allows the Arduino to communicate with different devices is included as a header in this part. This is the *XbeeDario.h* and was provided by the Computer Technology Institute and Press "Diophantus".

Furthermore the global variables were declared at this stage as well. These include the Arduino's pins and to what they are connected to.

### 5.4.2 setup()

The connected pins can be set as input or output depending on their purpose. This is done in the *setup()* part of the code.

In addition the communication between the Arduino and the XBee is to be initialised. Moreover the XBee is connected to the pins of the Arduino. Therefore it is necessary to specify the communication serial port between them two.

### 5.4.3 loop()

*Loop()* is the final and longest part of the code. Firstly Arduino checks if the other two boards (*sensor, action*) are connected to it. The program can continue its execution if and only if this is the case. If on the other hand the two boards are not connected the program will be halted.
If the *action* board (relays) are connected, the number of relays are checked. As the specifications of the room in which the system is installed may vary, the *action* board may not have all four relays installed. To check how many relays are on the *action* board different resistances are used for each case. Therefore different current values will be measured in the *relayCheckPin*. The *relayCheckPin* is a dedicated analogue pin (A4) on the Arduino Mini Pro board. These values are mapped in the code and the system reports back how many relays are installed.

The next function controls the output of the relays. Before responding to the relays, the Controller needs to check which of the lamps are on and which are off and act accordingly. The system also reports any change on the status of the lamps to the Controller.

The following group of statements checks the status of the motion sensor (*PIR - Passive InfaRed*). Every half of a second the status of the PIR sensor is checked and its status is reported as well.

Following, the light sensor is called every three seconds and its value is being reported at the time of measurement.

The value of temperature is measured and reported, via a dedicated analog pin, at the same rate.
The value of methane ($CH_4$) is also measured every three seconds. This is simply done by reading the analogue value of the sensor's (*MQ-4*) output pin.

It was not possible to measure the *CO* levels as there was an ambiguity on the values returned from the *MQ-7* sensor. The issue was determined to be related to the heating time and the working cycles before sensor's values were evaluated.

## 5.5 Database

*Überdust* is a database based on the Hibernate [?]. Hibernate is a Java framework based on an object relational mapping library [?]. The following (Figure 5.17) UML diagram is used to visualise the structure of *Überdust*.

`wisedb2.eps`

Figure 5.17: *Überdust* - UML description diagram

Furthermore the entities (Setup, Node,Link, NodeCapability, etc) are MySQL tables of Java classes used to represent sensor data objects. For each of these Java classes there is an XML file which describes the MySQL table. Once the database is created by using *controllers*[1], which manage the MySQL tables, some specific functions are called. These functions are used to add values into the database as well to add nodes etc.

[1]controllers in this case are functions of Java and irrelevant of the Controllers mentioned in 3

# Chapter 6

# Results

*Apollo* was fully built as shown in the following Figures 6.1 6.2 6.3.

Figure 6.1: Sensor board - Completed with all parts soldered

Figure 6.2: Control board - Completed with all parts soldered

Figure 6.3: Action board - Completed with all parts soldered

The following figure shows the connections between the three boards:

Figure 6.4: *Apollo* - All boards are connected

The system was installed into the rooms 0.I.1, 0.I.3, 0.I.9 and 0.I.11 of the lower ground floor of the Computer Technology Institute and Press "Diophantus" building in Patra, Greece. The rooms were chosen based on their size and number of occupants. This was an extra test for the reliability of the system. For example, the number of lighting systems and air-conditioning units varies depending on the room size. This means that no room should have the same size-occupants-units configuration.

The moment each system went live it started reporting the environmental values and storing the recording into the *Uberdust* database. The data collected into the database for each room separately, categorised by the type of sensor, is shown below.

## 6.1 PIR

### 6.1.1 Room 0.I.1

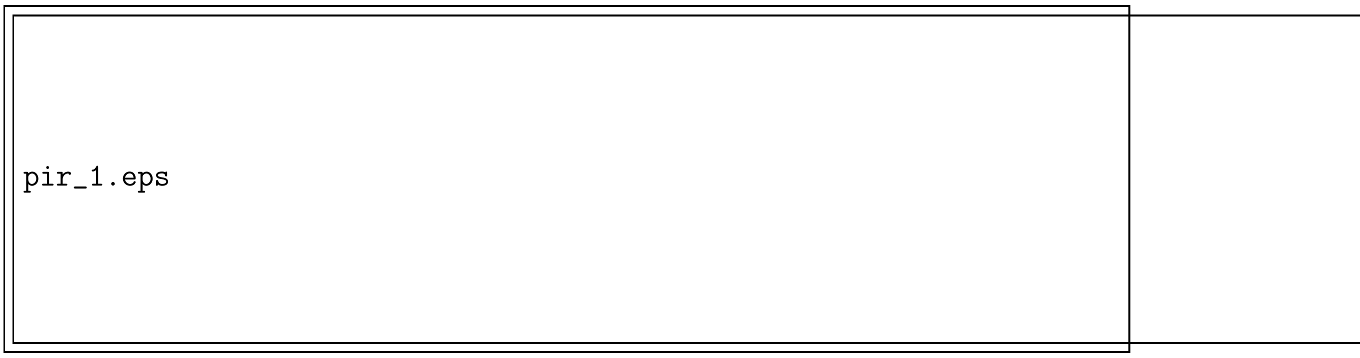

Figure 6.5: *Überdust* chart for for PIR sensor in room 0.I.1

### 6.1.2 Room 0.I.3

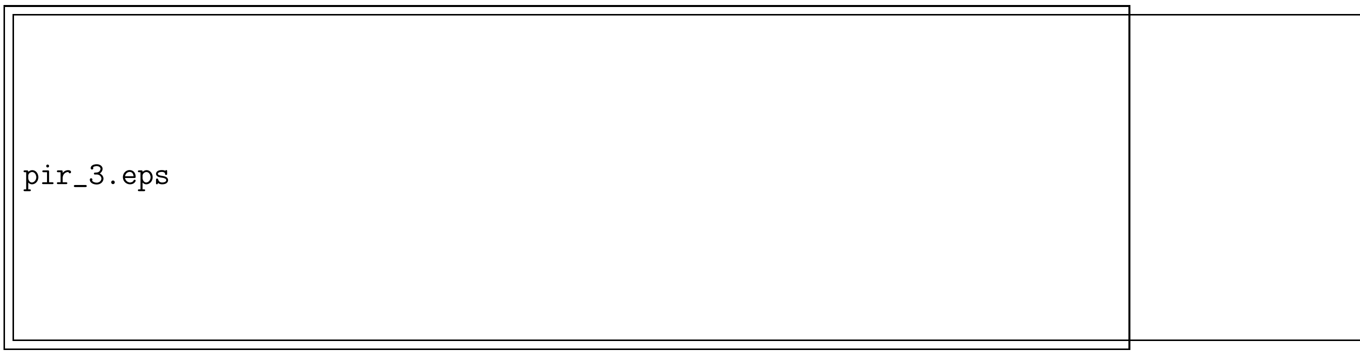

Figure 6.6: *Überdust* chart for for PIR sensor in room 0.I.3

### 6.1.3 Room 0.I.9

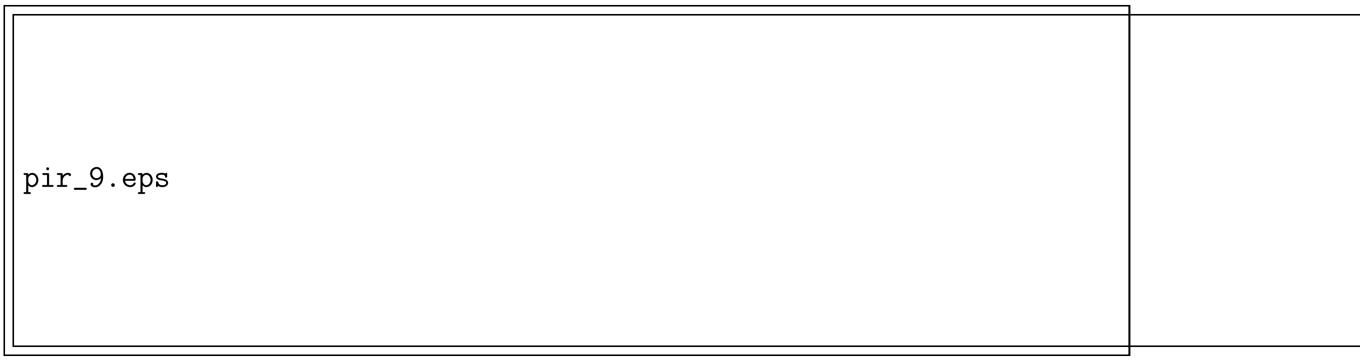

Figure 6.7: *Überdust* chart for for PIR sensor in room 0.I.9

### 6.1.4 Room 0.I.11

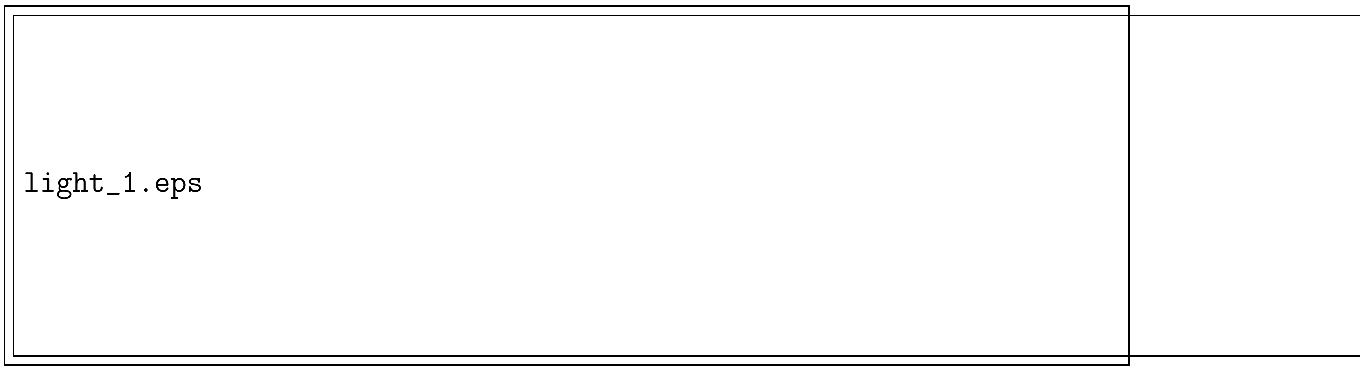

Figure 6.8: *Überdust* chart for LDR sensor in room 0.I.11

## 6.2 Light

### 6.2.1 Room 0.I.1

The following Figure 6.9 displays the pattern of the light emittance that is followed each day.

light_1.eps

Figure 6.9: *Überdust* chart for LDR sensor in room 0.I.1

### 6.2.2 Room 0.I.3

The light's strength during a day is visualised in the expanded chart in Figure 6.10.

light_3.eps

Figure 6.10: *Überdust* chart for LDR sensor in room 0.I.3

### 6.2.3 Room 0.I.9

In this case the illuminance of different room is shown.

pir_9.eps

Figure 6.11: *Überdust* chart for LDR sensor in room 0.I.9

### 6.2.4 Room 0.I.11

The occupants of this room have a preference of low light emittance. This can be seen from the low values reported from the light sensor.

light_11.eps

Figure 6.12: *Überdust* chart for LDR sensor in room 0.I.11

## 6.3 Temperature

The temperature plots show a general trend of high temperatures during times in which rooms are unoccupied – i.e. after working hours. During working hours – circa 08:00h to 18:00h – the temperature oscillates between 20°C and 30°C. The air conditioning systems seems to start once the temperature reaches the "critical" value at 26°C.

### 6.3.1 Room 0.I.1

The temperature shows a steady value around 24°C during the whole day.

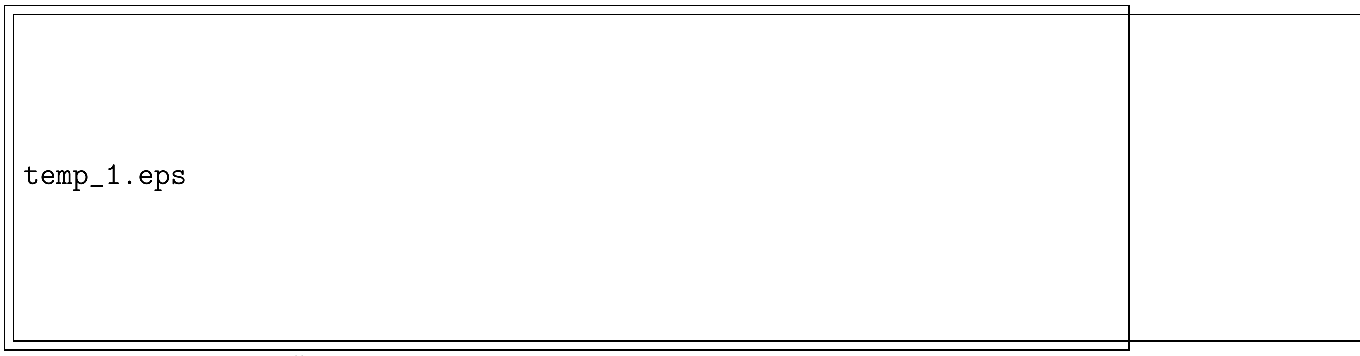

Figure 6.13: *Überdust* chart for Temperature sensor in room 0.I.11

### 6.3.2 Room 0.I.3

As all rooms contain several machines with high computing power the levels of dissipated heat are quit high.

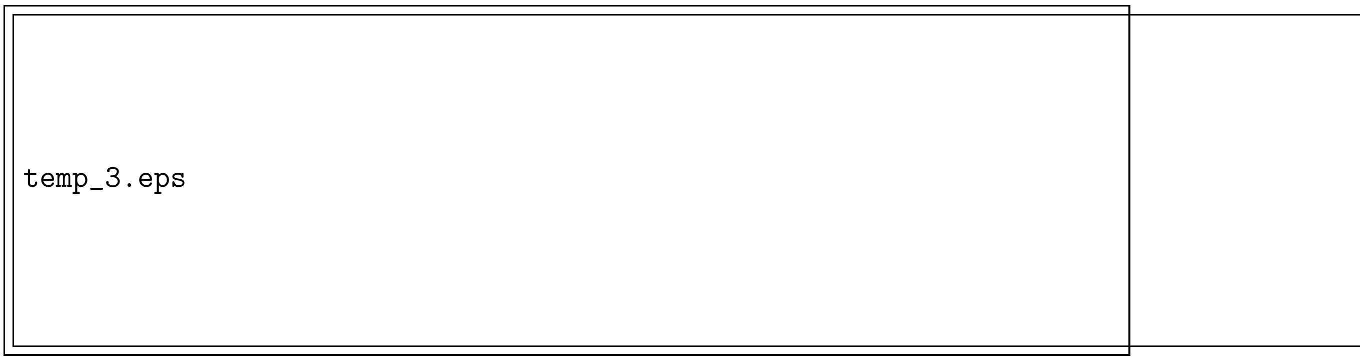

Figure 6.14: *Überdust* chart for Temperature sensor in room 0.I.11

### 6.3.3 Room 0.I.9

The same situation occurs in this room.

`temp_9.eps`

Figure 6.15: *Überdust* chart for Temperature sensor in room 0.I.11

### 6.3.4 Room 0.I.11

The temperature drop at around 14:00h in Figure 6.16 indicates that the ventilation was suddenly turned on.

`temp_11.eps`

Figure 6.16: *Überdust* chart for Temperature sensor in room 0.I.11

## 6.4 $CH_4$

The *Methane* $CH_4$ levels are constant and at a low value. The *glitch* noticed between 08:00h and 12:00h is due to a sensor's fault in the return value.

### 6.4.1 Room 0.I.1

ch4_1.eps

Figure 6.17: *Überdust* chart for PIR sensor in room 0.I.11

### 6.4.2 Room 0.I.3

ch4_3.eps

Figure 6.18: *Überdust* chart for PIR sensor in room 0.I.11

### 6.4.3 Room 0.I.9

ch4_9.eps

Figure 6.19: *Überdust* chart for PIR sensor in room 0.I.11

### 6.4.4 Room 0.I.11

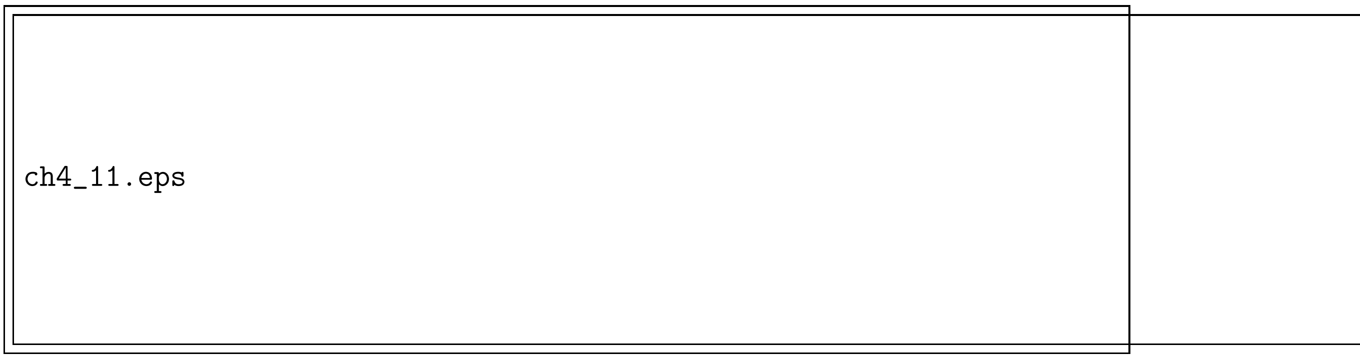

Figure 6.20: *Überdust* chart for PIR sensor in room 0.I.11

## 6.5 CO

Unfortunately, as explained in section ***5.4.3 loop ()***, the *CO* sensor is not yet functioning properly.

# Chapter 7

# Conclusions & Future Improvements

To the author's opinion the project can be characterized as successful due to the fact that all the set aims have been achieved. The system is installed and working in the Computer Technology Institute and Press "Diophantus" building, lower ground floor. Evidence provided by the database feed and everyday working status clearly supports the success of the system.

Moreover *Apollo* is able to collect the environmental values from the room is installed in and report them back to the *Controller* and, in extension, to the database. It is also capable of operating the lighting as well as the air-conditioning of the room. This is a novel approach for such a system, as with the use of the IEEE 802.15.4 protocol, the communication with other systems can easily be achieved. On that point the system can work along with other systems such as TelosB, SunSpot, iSense and any other system that uses the same communication protocol.

*Apollo* has a great potential of expansion. A first step could involve the development of a pressure sensor system to detect which chairs, and consequently which desks, are occupied. This addition would then lead to the possibility of personalising the working enviromnent according to the each user separately. On possible scenario could be the controlling the intensity of desk lights, the computer screen brightness and all those work environmnent settings which are very personal.

In addition, the integration of a blue-tooth module connected to the *action* board could further expand the system's personalisation options. Nowadays every mobile device incorporates a blue-tooth connectivity option which could be used to detect the presence of a specific user in the room to set the working environment accordingly. This would mean handling the lighting conditions along with the temperature level to be set upon user's entry in the room, for example.

Introduction of security alerts via e-mail could be considered a further evolution. The system could recognise abnormal activities – for example fire hazards, thiefs, etc. The owner of the office could then be notified for this activity via e-mail.

Finally, the devised system could be further developed to fully integrate itself with existing Building Management Systems. These tend to be highly energy consuming mechanisms which control the building in a global not-so-smart manner – turning the central heating on in the entire building even though certain rooms are empty is definitely not a smart behaviour. Integrating *Apollo* to provide a smarter point of view could lead to remarkable energy savings.

# Appendices