UPPSALA
UNIVERSITET

IT 15 080
Examensarbete 30 hp
Februari 2016

Universal infrared adapter for air conditioners

Adeline Mercier

Institutionen för informationsteknologi
Department of Information Technology

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Universal infrared adapter for air conditioners

Adeline Mercier

This thesis explores how a phone app and an infrared remote control can be used at the same time in order to control air conditioners. It introduces the concept of a universal adapter able to recognize a command from an air conditioner's remote control in order to update the phone app. This project extends prior work on universal devices focusing on infrared communication by examining the relationship between the different infrared protocols used for air conditioners from different manufacturers.
This thesis proposes a model of a universal infrared adapter. This adapter is able to recognize a command from an air conditioner's remote control if this one is registered in the database. If not, the adapter can also learn the different commands of the new remote control and save it in the database. A sample of 4 air conditioners from 4 different manufacturers has been tested for this thesis. With a comparative case analysis, this research explored the role of the position and the size of a setting in the infrared message. The findings from the research illustrate how the different manufacturers are encoding infrared commands. It also show that the impact of the size of the message and the amount of data on the analysis of the data is more complex than previously thought. As predicted, similarities has been found between protocols, allowing the adapter to look for information in specific parts of an infrared message or learn the information from an unknown remote control.
The findings provide support for the key arguments and support the prediction that there are similarities between the infrared protocols for air conditioners, allowing the creation of a universal adapter.

Handledare: Jens Peter Schroer
Ämnesgranskare: Christian Rohner
Examinator: Wang Yi
IT 15 080
Tryckt av: Reprocentralen ITC

Contents

List of Figures

1 Introduction

Nowadays, most of the devices can be remotely controlled thanks to infrared communication. But now it becomes inconvenient having a remote control for each device e.g. TV, DVD player, air conditioning etc. So that is why you can now find universal remotes to replace most of them. Thanks to Internet of things, another solution is to control devices from your phone. That is how Scypho AB proposes to control your heat pump or air conditioner.
Scypho AB is Swedish start up based in Stockholm [19]. Myscypho is an app which allows controlling and planning the temperature of your home. But the app does not completely replace the initial remote control, it is only an additional option to control your heat pump or air conditioner. The issue is that using the initial remote control, the app is not aware of the changes. This thesis is treating about this issue.

1.1 Project Purpose and Goal

MyScypho allows the user to change the temperature of the house from his phone and reduce his costs. But the issue with this app is that it is not aware if the user uses the initial remote control to change a setting. So the universal adapter's goals are to receive the data from an air conditioner's remote control and analyse it in order to update the app when settings are changed with the remote control.
So this universal infrared adapter has to:
- be able to receive a command from any air conditioner
- analyse and understand the command in order to update the app (thanks to a comparison between the command received and the database)
- learn commands from a remote control which is unknown

1.2 Motivation

Users can now control their air conditioner from their phone, but the application is not aware when settings are changed using the existing remote for the system. Using the application Myscypho, the customer is only able to use his phone. The goal of this project is to build an adapter using infrared communication, which analyzes the command in order to be able to recognize what kind of command is sent and update the app. This adapter has to be able to recognize any remote control since it can be installed on any air conditioner's remote control. All remotes are using infrared communication but manufacturers are not using the same infrared protocol to communicate. Databases have already been created for universal remotes in order to work with any system. Thanks to these databases, they recognize the protocol used and then you just has to select the specific model that you are using. A new option has been added to the new universal remotes, it is called the "learning" mode. Indeed, if the remote control used is unknown, it is possible to add this remote to the database. Most of the codes for remotes can be found on databases on Internet and offer

the possibility to update universal remotes. Knowing that the databases for the infrared protocols (for multimedia devices) are pretty complete since some brands are using the same protocol, this is not the case for air conditioners. Even if multimedia devices and air conditioners are using infrared communication, the way of sending data differs. The first step will be to analyze the communication between a multimedia device and its remote since commands for this type of device are less complex than the commands for air conditioners.
Then the next step is to analyze if these protocols for multimedia equipments, are also used for your air conditioner.

1.3 Scope

The scypho sensors already have Bluetooth Low Energy communication set so this project is going to focus on the communication between the adapter and the controlled system. Then many other ways of communication can be used e.g Wifi, Bluetooth for sending the data from the adapter to the system but since all the systems are controlled by a remote using infrared communication, the focus will be on this one. This project will also include the management of a database to store the recorded brands.

1.4 Structure of the report

This report is divided in 5 main sections. First, the background is introducing infrared communication and the different works related to universal devices using infrared communication but also "universal" systems able to use several infrared protocols. The second part concerns the methodology. It explains the different steps to understand the different protocols used first for multimedia devices and then for air conditioners. The third section concerns the realisation of the universal adapter. It is divided in 2 subsections: the hardware and the software. It explains how the board is designed and the different functions contained in the program. And the fourth section shows the tests done during this project. Each tested device is introduced and the several experiments show the similarities and differences that will help for the creation of the universal adapter. The last section concludes the different tests and results got from the previous part and includes some potential future work that can be done in order to improve this new device.

2 Background

Before working on the creation of the adapter, it is first important to understand how the communication between the infrared remote control and the air conditioner works. So this first section describes how the infrared communication is set and how a signal is sent and modulated but also a description of the protocols and how they are encoded.
The second part of this section is dedicated to the related work. Some projects and devices have already been realised using infrared communication. The one used and described in this section show similarities that will support this project.

2.1 Infrared communication

Infrared communication has many advantages. It requires low power and it has high noise immunity. But also a portable infrared system is cheap and simple to build.
Despite this, the main problem remaining is the line of sight: although the emitter has an angle of tolerance, the remote and the controlled device have to be almost aligned. The transmission can drastically be affected by the presence of objects, persons but even dust or sunlight [8]. It is shown in this part how an infrared command is built and sent but also how the infrared receiver analyzes the signal.

2.1.1 Modulation

The infrared communication is very efficient since it does not suffer electromagnetic interferences. Its optical wavelength is between 870nm and 950nm. However, the infrared signal is sensitive to other kind of interferences e.g. lights, temperature and even bodies which can affect the signal. Here, the modulation of the signal is the key to avoid these disturbances generating errors in the message [1]. In order to recognize the signal, it is necessary to make the emitter blink at a particular frequency. Concerning remote controls, the frequency used is between 35kHz and 40kHz (mostly 38kHz). If the receiver is set to these particular frequencies, it is able to ignore all the others, and ignore all the other potential noise.

2.1.2 Protocols

In order to be recognized by the receiver, specific protocols are used. These protocols are referred by Commercial Infrared protocols (CIR). Most of the protocols have the same base: first a start bit, then the data. The difference between them is the encoding, or the different position of the information composing the data. There are two main protocols used by most of the manufacturers. RC5 or RC6 . They have been created by Philips and became a standard for infrared transmissions The other protocols commonly used for infrared communication

is the NEC protocol. Then some other protocols are often used due to their omnipresence on the market e.g. Samsung or Sony.

2.1.3 Encoding

For serial communications, encoding a 1 or a 0 depends on the protocol used. For each logical state, "mark" (low state) and "spaces" (high state) are used, as shown on figure 1. Pulses are sent every mark's event and the IR light is turned off while we are in the space state. Most of the time, the mark for a logical one and a logical zero have the same length, so only the spaces are compared.

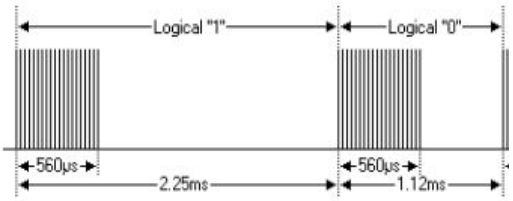

Figure 1: Logical states for the NEC protocol

However, from the receiver side, a mark will be represented by a low level of the receiver's output, and vice versa for the space. This phenomenon is represented on the figure 2. Notice that a mark does not represent a logical 1 and a space does not represent a logical 0 (see figure 1, 3) but both contains a mark and a space. It differs according to the protocol used.

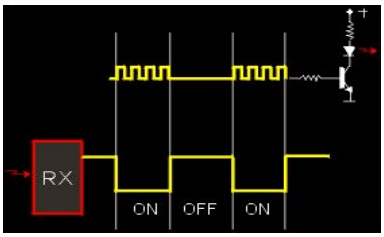

Figure 2: Interpretation of the signal by the sender/receiver

For representing the pulses, some protocols e.g. RC5 or RC6 are using Manchester encoding. For this type of encoding, 1 starts with a mark and 0 starts with a space as shown on figure 3.

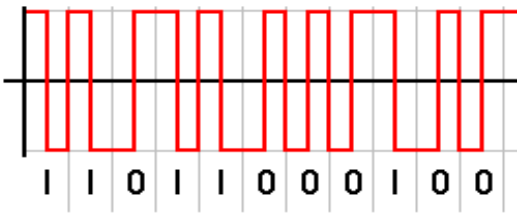

Figure 3: Manchester encoding

Recognizing a logical one from a logical zero depends on the length of a space.Marks for a logical 1 and a logical 0 are between 400 and 600ms whereas a space is between -1400 and -1650ms for a logical 1's space and between -400 and -600 for a logical 0's space.

2.1.4 Data frame format

Many different protocols are used, but if the data sent is analyzed, the same data frame format can be identify for all of them. First, the start bit, which contains one mark and one space that are much longer than marks and spaces for a logical 1 or 0. These mark and space have a different length compared to the other marks and spaces in the message. This differentiation indicates the beginning of the message. This starting part is very important for telling the receiver that this message is directed towards it. Then, the order between the address and the command differs according to the protocol. Protocols differ from one to another according to the time of the frame, the high and low part of the start bit, the one and the zero, or the number of repetition of the frame (See figure 4)

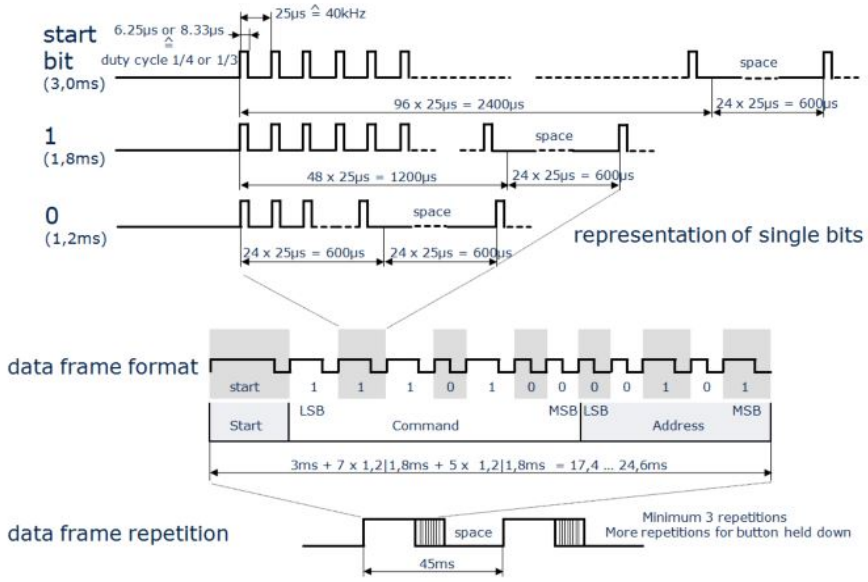

Figure 4: SIRC data frame format

2.2 Related work

This section introduces all the work at shows similarities with this project. The first three related works concern infrared protocols and libraries. Then are presented two devices similar to the sensors proposed by Scypho. The last related work concerns universal remote controls, more specifically the one with a "learning" function.

2.2.1 Ken Shirriff library

Ken Shirrif Library is an Arduino library able to recognize common infrared protocols (NEC, Sony, Philips RC5-6, Philips) [18]. These protocols are mainly used for multimedia devices. All the marks' and spaces' length are recorded for every element of the data frame (as explained in the section 2.1.3), and a decode function is written for each protocol.
This library has been useful for this project for understanding the composition of an infrared message and the position of the different information sent. It was used at the beginning of the test with a TV remote (See in section 5). After testing several air conditioners, it reveals that this library is not able to record long data frame (no more than 100 marks and spaces). So it is not possible to use Ken Shirriff library for this adapter.

2.2.2 Linux Infrared Remote Control (LIRC)

LIRC is a package used in order to analyse infrared signal [12]. This package can recognize, but also send infrared signals. It recognizes most of the common infrared protocols such as RC5, RC6, NEC etc. These known protocols covered more than 2000 devices from different brands. However, these protocols are mostly used for multimedia devices and not air conditioners.
It has been inspiring to see how the database with all the different protocols and brands were sorted, but this package will not be used for this project since it is a package for Linux and only multimedia devices are concerned.

2.2.3 AnalysIR

AnalysIR, is an Irish company, which is working on reverse engineering with infrared protocols [13]. They created a software that recognize an infrared protocol and shows its frame. This company is decoding new protocols on the go and has so far decoded more than 60 protocols. Compared to Ken Shirriff library, AnalysIR does not only focus on infrared protocols for multimedia devices but also for air conditioners. So they address the problem of recording long frames for air conditioners [14], by changing the baud rate and using interruptions. The sketch, which is freely available, is used for this project in order to record the data.

2.2.4 AirPatrol

Nordic air Patrol is a newly released device that allows users to control air conditioners from their phone [15]. Indeed, they propose a small device which can be positioned next to any air conditioner, and thanks to a SIM card, the user is changing the settings by sending SMS.This device can recognize around 20 brands. For some other brands e.g Daikin, Toshiba, the company Air patrol proposes another specific model for these brands [18]. Another variant of the model also allows to use the Wifi instead of a SIM card.
Air Patrol is a similar product as the Scypho sensors but they are facing the same issue: using the initial remote control, the app is not updated. It is interesting to see that this device, as the universal adapter, also contains a database with the commands of some defined brands. Air patrol Nordic proposes only the most common settings:
- ON/OFF
- mode: DRY / COOL / HEAT / AUTO
- temperature: from 16 to 30 degrees
- fan speed : AUTO / MIN / NORM / MAX
- Low heat: (8/10 C)

2.2.5 Tado

Recently released, Tado is an air conditioner controller [16].Designed in Munich,

this smart device that allows customers to control their air conditioner from their phone. This product is very similar to the scypho sensors. The app can work for any air conditioner, but the difference is that the app replaces completely the remote control.

2.2.6 Universal remote controls

Since 1985, Universal remote controls have been introduced in the everyday life due to the large amount of remote controls of every devices. The interesting part of the universal remote control for this project is that they are using infrared communication, but also their ability to now learn from another remote control. Indeed, they are now able, not only to replace a remote control thanks to their database, but also to learn the commands for another remote. Most of these universal remote control are used for multimedia devices since most of them are using the same protocols. Concerning air conditioners, most of the universal remotes that can be found do not have a learning mode, the protocols are already recorded by the manufacturer. That involves that the protocol of the air conditioner is not already recorded by the remote, it is impossible to use this universal remote unless the manufacturer updates it. So this project will focus on how the adapter can learn air conditioners' protocols directly from the remote.

3 Method

Before analyzing protocols used for air conditioners, first an analysis has to be done with protocols used for multimedia devices. Indeed, these protocols are already well known and documented, which is really helpful to understand infrared protocols and implement the functions receiving the data. Then, the next step is to compare these known protocols with the protocols used for air conditioners. After understanding how an infrared message is sent, the next step is to implement a solution. According to the results of this comparison, it will be possible to define the different criteria for implementing the "learning" function.
Several steps are defined for the two main functions of the adapter:
- Function 1: Recognize
Using the actual remote, send a command to the adapter. The infrared receiver will receive and analyse this command with the existing database to see if this system is already known. If it is, the adapter will look for similar information in the database in order to define the command that has been sent (temperature, fan speed...)
- Function 2: Update/ Learn
If the protocol is unknown, the adapter has to learn the remote used. In order to record the commands of this remote, and store the data in the database, each button has to be pressed in a specific order to be able to correctly store each command.

3.1 Infrared protocols for multimedia devices

Protocols used for multimedia devices are nowadays well known and documented. As said in the section 2.1.2, NEC, RC5 and RC6 are the commonly used. They all have the same carrier frequency (around 36KHz). After analysing these different protocols, it is possible to define how the different information are send through a single infrared message. here is an example of the similarities between the NEC protocol and the RC5 protocol.
- a start marker/ start bit which announce the beginning of a message
- an address (around 8 bits) for the receiver
- a command which is the content of the message
Information which can differ from a protocol to another are:
- the length of the command
- the addition of a final pulse in order to indicate the end of the message.
- the addition of logical inverse address or command which allows to verify if the address or the command does not contain any mistake.

3.2 Multimedia devices versus air conditioners

Since the infrared protocols for multimedia devices are nowadays well documented, it was important to analyse them as a first step to understand infrared protocols. But after the first tests with air conditioners (see section Tests and results), it reveals that infrared protocols for air conditioners are more complex than the ones for multimedia devices. Indeed, the length of the data for an air conditioner's commands most of the time are much longer due to additional information like the checksum, but also additional unchanged bytes included between different setting in order to avoid errors. Compared to multimedia devices, none of the air conditioners tested are using the same protocol.

3.2.1 Baud rate

One of the problems faced during the recording was the length of the message sent. In fact, during the tests, the length of the message is so far unknown and it is given by the program written for the receiver.
But during the tests for the air conditioner Coolman (see section 5.4), the messages were received and all the bytes seemed to correspond to a setting. However, when the same message was then forwarded to the air conditioner, the system did not recognize the command.
The Baud rate defines how fast the signal is sent.
In this project, the two first systems (KCC and Linda) did not need a rate above 9600 bits-per-seconds (bps) since the command sent was below 10 bytes. Then after several tests with different Baud rates, it reveals that for the next system tested (Coolman and Panasonic) the commands are sent with a Baud rate of 115200 bps.
Using the highest speed involves some limits concerning the hardware part. In fact, it might involve some errors while receiving because the speed is quite high to be supported by most of the microcontrollers. interruptions

3.2.2 Checksum

As explained in the previous section, each message sent starts with a start bit which has a random timing but this time is much more longer than a logical 1 or a logical 0 in order to be differentiate. The message also contains at the end 8 bits representing a checksum. It allows the receiver to check if the data has been received correctly. As the name introduced, it is a sum of all the bytes of the message. But it is not a simple sum; several analyses have been done to figure out how the checksum is calculated. After several explanations were found for different brands:
- a simple sum
- Counting the number of 1 (or 0) of the entire raw then compute this sum modulo 15. This new result is finally flipped and reversed.
- Doing the sum with all the bytes flipped (or reversed)
- using a XOR instead of a sum

But for the systems studied for this project, none of these solutions can be applied. After multiple calculation, the solution found for the 4 systems tested is:
- flip the bytes
- sum them
- flip the result to obtain the checksum

For example, with the 7 bytes shown in figure 5, the checksum which should be obtained is 11110011.

This method can be applied to the brands tested for this project: Coolman and Panasonic. KCC and Linda do not contain a checksum. It is important to understand each byte contained in a command, but since the checksum is representing all the settings at once and all the air conditioners have different

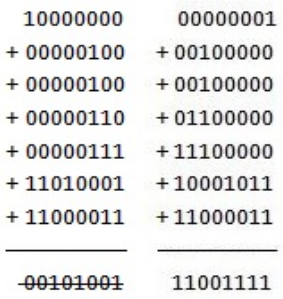

Figure 5: normal Checksum (left) and "flipped" checksum

settings, the analyse of the checksum will not be included in the program.

3.2.3 Settings

All the air conditioners have the same following settings:
- the temperature in degrees
- the vertical position of the fan (from 1 to 3 and auto)
- 4 modes (dry, cool, heat and auto)
- the speed of the fan (from 1 to 3 and auto)

However, some additional functions differ depending on the brand like the temperature in Fahrenheit, the horizontal position of the fan, additional vertical positions of the fan, additional speeds of the fan, a quiet mode etc. So the program has to include all these functions in order to be able to cover any kind of system. In this chapter are described how these settings are represented in the data frame depending on the brand of the system.

The temperature:
The temperature is usually represented as a flipped binary table
10000000
01000000
11000000
00100000
etc.

So a function should be able to recognize the temperature directly from the row received. However, this flipped table does not always start from the minimum temperature but from the second one or even the third one. These temperatures which are not included in the logical sequence differ according to the brand. So it is not possible to define a universal function recognizing the temperature from the sequence. The rows will have to be stored in the database. The temperature is always in the last position of the settings in the message sent.

The timer:
Timers for air conditioners differ from each other depending on the manufacturer: some can set a timer with only full hours and the more advanced one can set hours and half hours. For the first one, only 8 bits are needed whereas the other ones require 2 bytes: one for the full hour and the other one in order to represent the half hours.
Fan speed:
Fan speeds are represented on 1 byte or 8 bits depending on the protocol and the number of speeds that the air conditioner has. The number of speeds usually differs from 3 to 8.

Fan position:
For the position of the fan, it can be changed horizontally or vertically. On basic air conditioners, only the horizontal position can be changed. In order for the infrared adapter to cover these two types of air conditioners, the data collected for these information are on 2 bytes.

Mode:
4 modes can be found on an air conditioners: AUTO, DRY, HEAT and COOL. Depending on the mode chosen, it is possible to change more or less settings. For example, the auto mode only allows one fan position, one fan speed and one predefined temperature.
These modes can be represented on 4 bits: 00, 01, 10 and 11; but for security, they are represented on 8 bits. This setting is mostly the one in the first position in the message sent.

In addition to these common settings, some other ones can be found on air conditioners like a sleep mode or an "health mode". Since it is impossible to classify all the different functions for all the different air conditioners and the adapter will be use only on the user's air conditioner, the function written in order to record the commands is planned to propose to record 3-4 "additional commands". These additional commands do not have a precise name in the database so they can be used for any uncommon function of an air conditioner. It has been decided that this type of function would be defined as "additional", since the settings that users usually change are mostly the temperature, the timer and the fan speed, so it was not relevant to allow memory of the database for functions that can be specific to a single system.

4 Implementation

In this section are presented the hardware and software part of the universal adapter. So first will be presenting the board, the testing board and the different components used.

4.1 Hardware

In order to be able to receive the infrared signal, the universal adapter needs a board. This section describes the chosen board for this project, the board used for tests and it also includes a description of the components needed for the universal adapter.

4.1.1 Arduino board

In order to implement this project, an Arduino Uno board has been choosen. If the time allows us, switching from the Arduino board to a microcontroller or another smaller device would be an option in order to get a smaller device and a bigger memory for the database. The Arduino Uno has a microcontrooler ATmega328P, a 16Mhz quartz, a USB connection. Concerning the input/output pins, the Arduino Uno has 20 pins: 6 analog inputs, and 14 digital input/ouput. 6 of the 14 digital pins can also be used as PWM. These are the main components used for this project.

4.1.2 Testing board

Before testing the program for the adapter on an air conditioner, it is necessary to make sure that the board, and the program are sending and receiving properly. So a testing board with an infrared receiver and an infrared emitter is set. These components are also connected to a second Arduino board dedicated to the tests. This testing board allows to see if the data sent is and and/or correctly received.

4.1.3 Infrared receiver

An infrared receiver is needed in order to analyze the command. One receiver will be used for the adapter, another one will be needed for the testing board. IR receivers in general include a diode which receives the signal, an amplifier and a limiter in order to get a constant signal as an output, whatever the position of the emitter. Then a band pass filter to define which frequency the receiver can read. In this situation, the receiver has to be set for frequencies between 30kHz and 60kHz. This filter reduces the interferences from other devices which work with higher or lower frequencies. A detector, an integrator, and a comparator (which is pulled low in case of the presence of the modulation frequency) are also integrated to the component in order to detect the frequency.
The best match for IR emitters for all the requirements and also used by most of remotes control is the TSOP 38238 from Vishay [9]. This component has been miniaturized especially for this use. Moreover, the universal remote control is used in free ambient, so this receiver will be one of the most efficient and reliable since this component "improved immunity against ambient light" and its carrier frequencies are between 30kHz and 56kHz, with perfectly match with remotes' frequencies. This component gives directly a rectangular signal as an output,

so the output can directly be connected to the microcontroller to analyse the output.

4.1.4 Infrared emitter

An infrared emitter is needed on the testing board in order to test if the data has been correctly analysed. Indeed, after learning all the settings, the only way to make sure that they have been correctly understood is to send it to the air conditioner.
After having an overlook on the different remotes available on the market, and the different projects built around universal remotes, the Infrared Emitting Diode chosen for the adapter is the diode TSUS540 from Vishay [7]. This emitter offers the biggest angle of emittance e.g. +- 22 degrees. The peak wavelength is correctly in the infrared color range e.g. 950nm as shown on figure 6

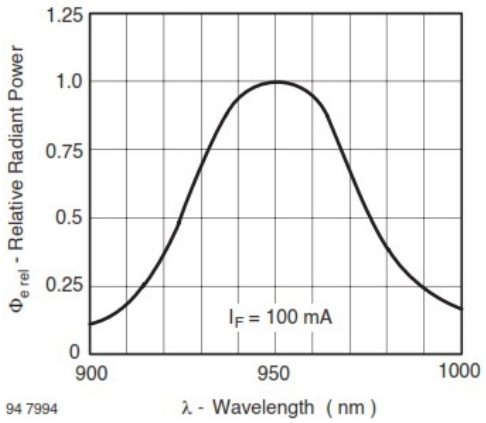

Figure 6: Relative Radiant Power vs. Wavelength

4.1.5 Range

Due to an ambient area, it is complicated to send a signal without interferences. Hence, several solutions have been thought in order to solve this problem. First, when the transmitter was directly connected to the Arduino board, the range was extremely short (<15cm). Then a NPN transistor has been added in order to amplify the current given from the Arduino board to the infrared emitter (from 20mA to 100mA). This solution extends the distance from 10cm to 30cm. But this distance is still too short for the use of a remote. The solution found for this was the Darlington configuration. This configuration consists in connecting 2 NPN transistors in series like the figure 7.

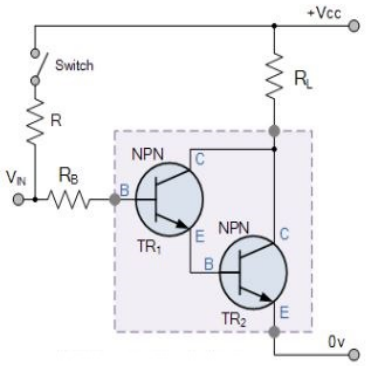

Figure 7: NPN Darlington Output

4.2 Software

Learning and recognizing are the main functions of the adapter. In any case, the adapter has to recognize the command from any air conditioner's remote control, but it is also "learning" if the system used is not already recorded in the database.
Arduino proposes its own software to program on the Arduino board. Thanks to this open-source software, it is easy to write compile and upload a program on the Arduino board [10]. The Arduino language is similar to C/C++. For this project, the version 1.0.6 is used. The following section will describe the different functions written for the universal adapter, but also the database used to store the data.

4.2.1 Learning

Due to many differences between the protocols used for air conditioner, creating a universal way of learning them is complex. The first difference is the length of the command, it is the easiest information to get from the message sent it will be the first information recorded. Then the start bits and the checksums

(so the beginning and the end of the message) will be ignored since they are not giving relevant information concerning the positions of the settings. The first common setting for all air conditioners is the temperature. So the user is asked to first select the lowest temperature and then the maximal temperature of the air conditioner. During these 2 records, none of the other settings should be changed. By receiving these 2 information and comparing the 2 rows of data received, the position of the byte(s) defining the temperature can be found. In order to deduce a relevant position of the setting, finding a bit that differs from a temperature to another is not enough. Indeed, settings are usually encoded on one byte minimum. So from the position of 1 bit, it can be deduced that the entire byte containing this bit represents the setting.
For example:
after the comparison of to rows, the bit in position 35 differs: Position of this bit in a byte:
i = (35 modulo 8)+ 1 = 4
So in addition to the bit 35, the next 8-i bit(s) and the previous i-1 bit(s) are also defining the same setting.

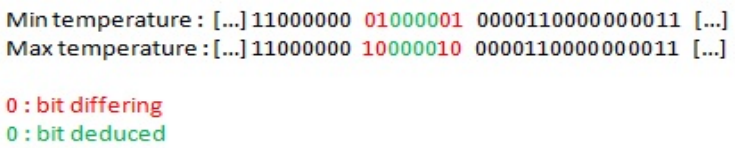

Figure 8: Raws comparison

Once the position is known, the user decreases the temperature degree by degree in order to record the value(s) of this/these specific byte(s) for each temperature. This method is showed on the figure 9 This procedure is used for all the different settings of the air conditioner: fan position, modes, fan speed, timer... When the common settings have been recorded, the program asks if other settings have to be recorded. In the database, 5 more tables have been created in order to allow additional settings.

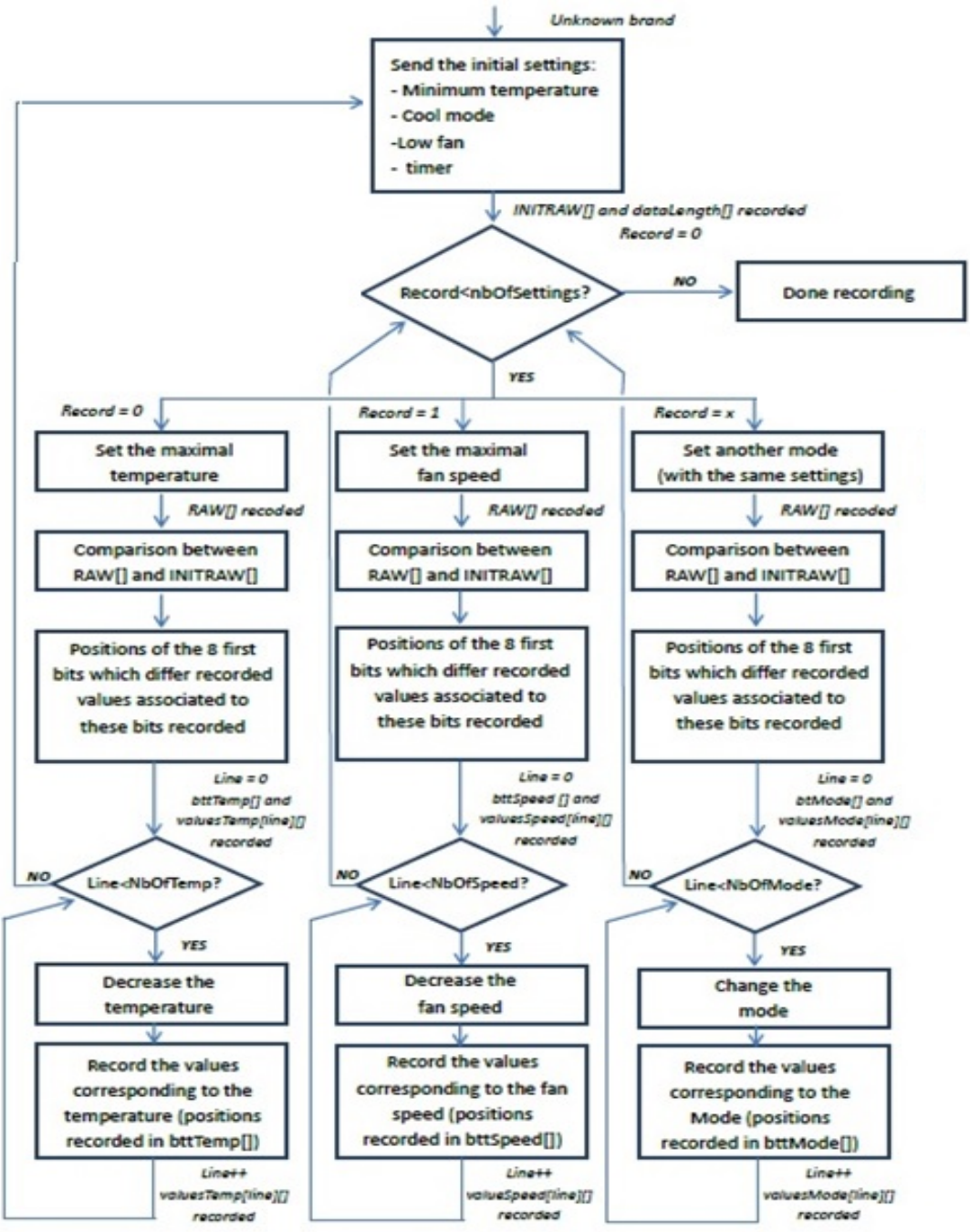

Figure 9: Final chart, learning

4.2.2 Recognition

To recognize the setting which is changed by the remote control, the new command sent has to be compared with the previous one saved in the program. Then it appears that several parts of the data differ: the actual setting desired but also the checksum at the end of the command. The first idea was to register both information as references for each setting. But since the checksum represents all the settings (mode, temperature etc.). It is impossible to record, for example, the data representing 17 degrees for each mode and every different fan speed. That will already represent at least 9 different combinations for only one temperature. So it is decided to ignore the checksum and only the part representing the setting itself.
From the comparison of the two rows, the position of the bits that changed is defined. This position is then compared to the information recorded in the database in order to deduce the parameter modified. Once the parameter has been found, the next step is to determine the new value of this parameter. For example, once the program knows that it is the fan speed that has been modified, it will now have to determine the new speed set. So from the known positions, the values of these bits are compared with another the corresponding table. This methodology is described on figure 10.
The methodology which is not describe in this figure is for the auto mode and the sleep mode. Since this 2 settings require a specific table, they also need a specific comparison. So before starting comparing the previous raw with the position of the bits which differs, the entire raw is first compared with the auto modes and sleep modes recorded. Then, if one of them is recognized, the program does not go through the other comparison.

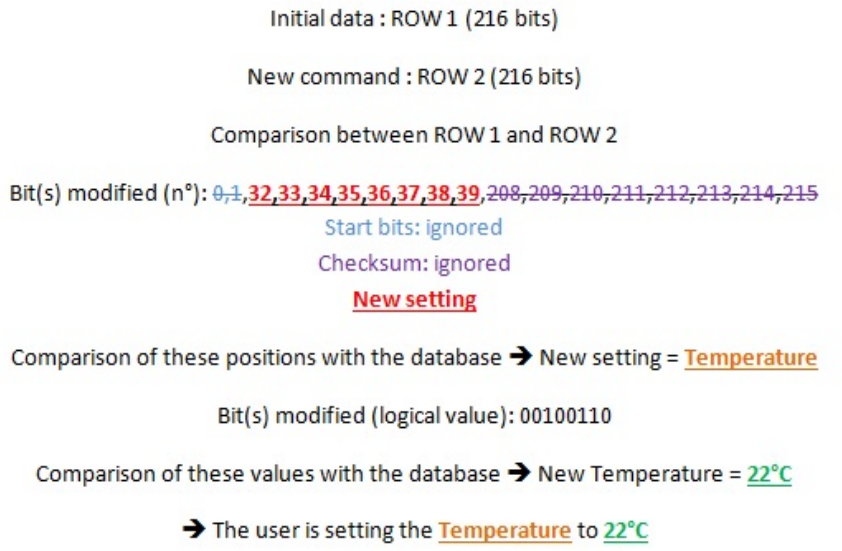

Figure 10: Recognition's steps

4.2.3 Combination

Once there are, on one hand, functions able to recognize a remote control (recognition) and, on the other hand, functions able to learn a new remote control (learning), a program which combines these 2 actions is needed.
So the first step is to, from a random command sent, determine if this protocol is already known or not. This can be deduced from the data length. Since several brands can have the same data length so after comparing the length of the command with the database, the user will see a list of the brands found with the same data length and will be able to choose the brand if it is in the list. If the air conditioner is on this list, the program will call the recognition functions. Otherwise, it will call the learning functions and then go back from the start (see figure 11) .

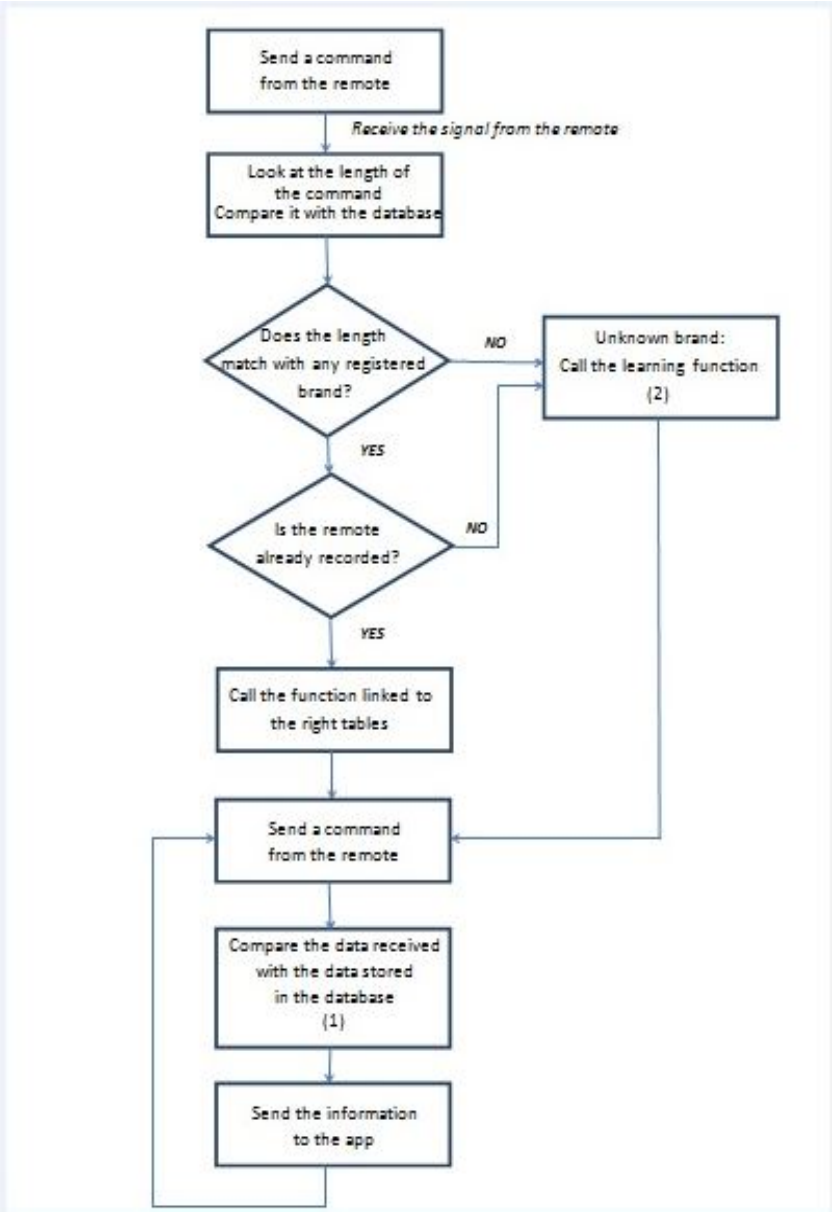

Figure 11: Final chart, combination

4.2.4 Arduino's Database library and extended database library

Because of the big amount of data that has to be saved in order to be able to recognize the different settings of several brands, it is necessary to use the Arduino database library. The Arduino database is using the EEPROM memory instead of the RAM memory which allows freeing the RAM for any additional functions. So for all systems, the same structure is defined :
- the positions of the bits defining a parameter (one table per parameter)
- the different values that these bits can have (one table per parameter which contains all the possibilities)
- the name of the brand
- the size of the command sent

Example:
Char name[] = "Panasonic";
int commandSize = 439;
int bitsSetting[] = 21,22,23,24,25,26,27,28 ;
char valuesSetting[][]=
'L', 'L','H','H','L','L','H','H',
'H', 'L','H','H','L','H','H','H',
'L', 'L','L','H','L','L','H','H',
'H', 'L','L','H','L','H','H','H',

BitsSetting and valuesSetting are generic examples. In the database, one table of each is created for every setting (bitsTemp, valuesTemp, bitsFanSpeed, valuesFanSpeed etc.). There are 2 exceptions for these table: the auto mode and the sleep mode. Indeed, when one of these modes is set, it does not matter which setting was set before, all the different parameter can be changed. So since the adapter can only recognize the command when one parameter is changed, these 2 modes have to be recorded into specific tables.
The limitation of the Arduino database is the number of records which can be saved in it. Indeed, this number depends on the EEPROM size of the Arduino board and the size of the records
(EEPROM size / record size = max records) [19]. In the case of the Arduino Uno, the Atmega328 has 1kb of EEPROM storage. Another known issue is the number of records per tables. But since the maximum of records for the adapter is 48 (for the timer), this issue does not concern this project. A solution found for the lack of memory is to use the extended database library.
Using the extended database allows more records and so solved the problem of recording all the settings of one remote control and having another one saved in the database. For this project, this database will still be used but another way to save the data has to be taken in consideration as a future work. Another issue concerning the program is the size of the sketch. Indeed, in order to record the commands in the database, it is necessary to first record the message in the RAM to analyse it. Since the tables allocated in the database have different sizes depending on the number of bytes needed to encode a setting, it is not

possible to create only one table to receive the data. So all the tables needed in the database for one remote also have to be created in the sketch to receive and analyse the data. Since the RAM on the Arduino Uno is only 2kb, It is not possible to create all these tables. The program has been completely written but only a part of it can be run due to that limitation. Forwarding the program to another platform would be a part of future works.

5 Tests and results

In order to realize an universal adapter, several air conditioners from several manufacturers have to be tested in order to understand how the commands are encoded. In this section are shown the different air conditioner tested and the way the data is sent for each of them. The differences and similarities are also explained.

5.1 Samsung TV

The fist device that has been tested is a Samsung TV. A multimedia device has been chosen in order to verify the board and the program with simpler and already well known protocols. As mentioned in the section 2, the most famous infrared protocols for multimedia devices are NEC, RC5 and RC6. But some big companies like Mitsubishi have their own protocol for their multimedia devices. Concerning the Samsung TV, the first step is to see if the remote control is using one of these 3 common protocols. After sending different commands to the receiver with the remote control, it has been deduced that Samsung is using its own protocol. Samsung is really expended in the multimedia devices' world, its protocol is also well documented via the manufacturer itself [22]. Once the Samsung protocol implemented, the goal of testing this device is to understand how the information are sent to the receiver. It is already know that the 2 first bits represent the starting bits in order to tell the receiver that it is the beginning of the message. The rest of the data has to be figured out. Because even if the protocol of a specific brand is know, the position of the information can differ from a device to another.
In order to find the different information contained in an infrared message from the Samsung TV's remote control: the following steps have been processed:
- send a first message to the receiver and save it
- send a second message changing one setting (.g. the channel)
- compare the 2 messages received and deduce the part of the data which has been modified

These actions have to be repeated for every buttons of the remote in order to find the different positions for the different settings (volume, channels etc.) and the different values for these settings (channel 1,2,3 etc.). This methodology is necessary in order to find a logical way to read and analyse the message sent by the remote. From this first tested device has been written a function able to recognize the differences between 2 messages. The program written during

this first part had to avoid being too specific because the protocols from air conditioners are slightly different and this first device was first used in order to understand and analyse an infrared message.

5.2 LINDA

The first air conditioner tested is a system from Loma Linda Heat and air conditioning.
This air conditioner only has the basic options: 3 fan speeds, timer with full hours, horizontal fan position.
So the first step is to see if the protocol is recognized by the program written for multimedia devices (with the NEC, RC5, RC6 or Samsung protocol). After sending different commands to the testing board, the results, see figure 12, showed that some protocols where sometimes recognized but were unknown most of the time.

Command	Number of command sent	Multimedia device's protocol recognized (%)	Unknown protocol (%)
Temperature	40	55	45
Fan Position	12	25	75
Fan Speed	12	41,67	58,33
Timer	50	18	82

Figure 12: recognition of multimedia devices' protocols

The results from this test show that the LINDA protocol is not similar to protocols used for multimedia devices, but some similarities can be found for some commands.
The next step is to understand how the data is sent, so where the information are placed in the raw of bytes sent by the remote. When a command is sent, the testing board receives 8 bytes. After changing the settings one after another, the positions of the commands in the data frame can be deduced, see figure 13

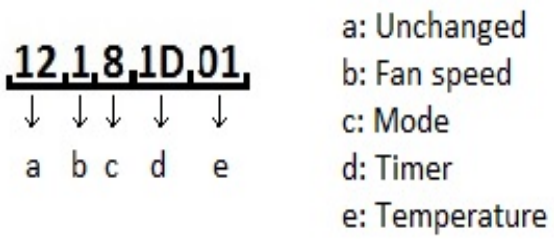

Figure 13: Linda's data frame

Once the position of each setting is found, the function that can recognize the command sent can be written. But this function will be modified later in order to be able to match with any air conditioner's protocol. For now, the adapter can receive correctly an infrared message from a remote control.

5.3 KCC

The second air conditioner tested is a system from KCC. It has the same basic functions as the Linda system except the temperature can be given in degrees or in Fahrenheit. Testing this system will allow to see if the program previously written works with this system. The length of the data is also 8 bytes.
After the first test which consists of receiving the different command and see first if the protocol is already known. As shown on figure 14, the results of this test show that the protocol used for this air conditioner is unknown. So the next step is to analyse all the commands, and see what are the positions of the different settings in the message sent. This second test allows to find similarities with the previous air conditioner tested. Indeed, the amount of bytes used for the common setting are similar (e.g. 2 bytes for describing the temperature). This information will be relevant in order to create a function to recognize the different settings and to know how much space has to be allocated in the database. However, the position of these settings in a message sent by the remote control are sightly different.
So in comparison to the first system tested (LINDA), the lengths are the same (for the settings and the message) and both systems have a starting bit. So even if the settings are set differently in the message, it does not affect the plot of the

program. The program is now aware of the memory needed for each setting for an air conditioner. It can also recognize the difference between two messages and define the part of the message that differs and deduce the setting that has been changed. On the other hand, it can record the different settings in the database.

Command	Number of command sent	Multimedia device's protocol recognized (%)	Unknown protocol (%)
Temperature	40	25	75
Fan Position	12	8,3	91,7
Fan Speed	12	0	100
Timer	50	12	88

Figure 14: KCC's data frame

5.4 Coolman

The third air conditioner tested is a system from Coolman. This system has more advanced functions like a sleep mode, horizontal and vertical an position, a dry, cool, and an auto mode. The first step is to test all the commands in order to define their position in the row of data received.
After this first test, some positions have been defined but commands appeared to be similar. The reason is that the length of the command was too long compared to the length set for the receiver, so only the beginning of the data was received. This problem has been solved after testing a Panasonic air conditioner. As with Coolman the same problem was discovered, all the messages were the same, no matter the command. Hence, it has been deduced that a part of the message was not transmitted.This problem has been figured out by AnalysIR through a Panasonic air conditioner [14].
The two first air conditioners tested (KCC and LINDA), had a data length of 8 bytes whereas the air conditioner Coolman sends messages of 13 bytes and 27 bytes for Panasonic, this why the data received was not reliable since the program written for the receiver receives only 10 bytes.
In order to be able to receive more data, both the receiver and the remote control have to send/receive the data with the same rate. Hence, the Baud rate needs to be changed for the receiver part.
The other difference with the previous tested air conditioner is due to the amount of data sent, a checksum is used at the end of the message sent in order to verify if the command is correct. The checksum, as explained in section 3.2.2, is a byte

(or sometimes 2) which contains the information about all the settings. This part of the message allows the receiver to check if there is any error in the message received. In order to simplify the "universality" of the adapter, this part of the message is not taken in consideration during the analysis of the message.
Now the adapter can receive messages of more than 10 bytes and memory for more advanced functions of the air conditioner has been allocated.

Command	Number of command sent	Multimedia device's protocol recognized (%)	Unknown protocol (%)
Temperature	40	22,5	77,5
Fan Position	12	16,7	83,3
Fan Speed	12	25	75
Timer	50	2	98

Figure 15: Coolman's data frame

5.5 Panasonic

A Panasonic air conditioner is the last system tested. As shown on the figure 16, the Panasonic protocol is not recognize neither by the common protocols. The first analysis of the messages sent by the remote control revealed that the receiver was receiving an identical message no matter the command sent. This problem of reception has been solved by AnalysIR [14]. This topic has been treated since a lot of projects have been done about infrared communication but mostly with multimedia devices. And since messages sent via a remote control for air conditioner are much longer, this problem of reception has been recurrent.
Now that the infrared message can be received completely, it shows that if the data previously received was similar all the time because the first half of the message is a constant. All the information concerning the setting are contained in the second half of the message. After testing all the different functions of these air conditioners, it appears, one more time, that the common settings have the same size as the previous air conditioners tested. But compared to the previous air conditioners tested, the Panasonic air conditioner has more advanced settings like “health”, “sleep”, “quiet” or “powerful”. Since these additional functions differ from a system to another, the database will allocate 5 tables without a label in order to be able to record these kind of additional function for any air conditioner. The size of the setting will be unknown, but knowing that all the other settings are using between 1 and 2 bytes, 2 bytes will be allocated for these functions. Now that the 3 different systems have been compared, the structure of the database for the different setting can be set. In total, for one system, 14 tables are created. these are used for:
- the name of the air conditioner
- the length of the message
- the timer
- the fan speed
- the modes
- the auto mode
- the sleep mode
- the vertical air swing
- the horizontal air swing
- 5 tables for additional settings

Command	Number of command sent	Multimedia device's protocol recognized (%)	Unknown protocol (%)
Temperature	40	5	95
Fan Position	12	0	100
Fan Speed	12	8	92
Timer	50	8	92

Figure 16: Panasonic's data frame

6 Conclusion and future work

After the comparison of different air conditioners' protocols, the adapter is now able to recognize the commands from different air conditioners (temperature, modes, fan speed etc.) and record a new system in its database. Due to the large amount of different systems and protocols, this adapter cannot be guaranteed 100% since they have not been tested/recorded all. The infrared protocols for air conditioners are quite similar so it can be assumed that the learning function written according to the air conditioners tested will work for any air conditioner.
However, in order to be able to record all the settings, store the information in the database, and also be able to store more than 3 brands in the database, the problem of the memory has to be faced. So far, only the settings for one air conditioner is recorded in the database and one remote control can record the principal settings (the others have been written in the program but they have been disabled because of the lack of memory). As said in the section 4.2.4, the extended database is not big enough to record all the different parameters for different settings. In order to solve this problem, several solutions can be proposed:
- replace the Arduino Uno with an Arduino board with more
EEPROM
- create another board in order to be able to choose an adapted microcontroller and eventual additional components for the memory, such as an SD card
An optimization of the program can also be taken in consideration. First, because of the lack of memory for the sketch but also because the time needed to record a new remote is not really convenient since the user has to go through all of the settings. The first idea during this project was to find the logic in how the temperature was coded but it has not been found so far. If this logic can be recognized, the user will only have to go through 3-4 temperature before the logic is found by the program. Another optimization would be changing the way of storing the data. For example, instead of recording the 8 positions of the bits defining a setting ([23,24,25,26,27,28,29]), the idea would be to record the first position and the number of bits that have to be recorded after this one

([23,7]). Another step for the universal adapter would be to set up the connection between the adapter and scypho's app. Knowing that there is an existing communication between the scypho's sensors and the app, the easiest way would be to set a connection between the adapter and the sensors. Scypho's sensors can communicate via Wifi or bluetooth. From this statement, several options can be taken in consideration:
- using an Arduino Bluetooth [20] instead of an Arduino Uno
- using the LightBlue Bean [21] which has the advantage to read Arduino code, so the program does not have to be changed - or if an Arduino board is not used and the board is created from scratch, there will be a possibility to add a bluetooth module or a Wifi module.
In order to improve the learning part of the adapter, it would also be interesting to, instead of having a database for each adapter, have a global database which would allow implementing all the adapters when a new remote is recorded. The security of the data is not addressed in this project but this topic can also be a part of a future work.

References

[1] San Bergmans, "SB-Projects", available: http://www.sbprojects.com/knowledge/ir/index.php, [Accessed Feb, 2015]

[2] San Bergmans, "SB-Projects", available: http://www.sbprojects.com/knowledge/ir/rc5.php [Accessed Feb, 2015]

[3] Victor Flachs, Nimrod Peled and Yan Nosovitsky, "Multi-protocol infrared receiver", U.S. Patent 11,880,981, 29 Jan 2009, [Accessed April, 2015]

[4] Kai Di Feng, "Adaptive infrared communication apparatus",U.S. Patent 08,993,298, 18 dec 1997, [Accessed March, 2015]

[5] Joseph M.KAHN and John R. BARRY, "Wireless Infrared Communications", February 1997, Proceedings of the IEEE, Volume 85

[6] Wagner Lipnharski: "Infrared Remote Control", available http://ustr.sancaleve.com/infrared/infrared1.shtml, [Accessed Feb,2015]

[7] Vishay, "Infrared Emitting Diode", TSUS540 datasheet, [Re-vised Apr, 2004]

[8] Vishay, "General Overview of IR Transmission in Free Ambi-ent", 80073 Vishay Semiconductors, [Revised 11 March 2013]

[9] Vishay, "IR Receiver Modules for Remote Control Systems",TSOP38238 datasheet, [Revised Feb, 2015]

[10] Maureen Kaine-Krolak and Mark E. Novak, "An Introduction to infrared Technology: application in the Home, Classroom, Work-place, and Beyond...", Closing the Gap, 1995, Presentation Manuscript, [Accessed Feb 2015]

[11] Arduino, "Arduino Uno", available: http://www.arduino.cc/en/Main/ArduinoBoardUno, [Accessed March, 2015]

[12] Linux Infrared Remote Control, available: lirc.org/[Accessed March, 2015]

[13] AnalysIR, "AnalysIR blog", available: http://www.analysir.com/blog/, [Accessed April, 2015]

[14] AnalysIR, "Air Conditioners: Recording long Infrared Remote control signals with Arduino", available: http://www.analysir.com/blog/2014/03/19/air-conditioners-problems-recording-long-infrared-remote-control-signals-arduino/, [Accessed June, 2015]

[15] Air Patrol, "Control your heat pump with mobile", available: http://airpatrol.eu/[Accessed May, 2015]

[16] Tado, available: https://www.tado.com/gb/smart-air-conditioner, [Accessed July, 2015]

[17] Ken shirriff, "A Multi-Protocol Infrared Remote Library for the Arduino", available: http://www.righto.com/2009/08/multi-protocol-infrared-remote-library.html, [Accessed April, 2015]

[18] Air Patrol, "Compare products", available: http://airpatrol.eu/compare-products/, [Accessed June, 2015]

[19] Arduino, "Database Library", available: http://playground.arduino.cc/Code/DatabaseLibrary, [Accessed July, 2015]

[20] Arduino, "Arduino BT", available: https://www.arduino.cc/en/Main/ArduinoBoardBT?from=Main.ArduinoBoardBluetooth, [Accessed September, 2015]

[21] Punch through design, "The LightBlue bean", available: http://legacy.punchthrough.com/bean/, [Accessed June, 2015]

[22] Samsung Electronics, "Application Note, IR remote controller", October 2008 [Accessed March, 2015]

[23] Scypho AB, Join the evolution, available: https://www.scypho.com/en/, [Accessed March, 2015]