MASTER'S THESIS

Submitted in partial fulfillment of the requirements
for the degree of Master of Science in Engineering
at the University of Applied Sciences Technikum Wien.
Department of Embedded Systems

The Lipmouse: A Labially Controlled Mouse Cursor Emulation Device for People with Special Needs

By: Ibáñez Flamarique, Alberto

Supervisor 1: Dipl.-Ing. Veigl, Christoph
Supervisor 2: MSc, Deinhofer, Martin

Vienna, June 07, 2014

FACHHOCHSCHULE
TECHNIKUM WIEN

upna Universidad Pública de Navarra Nafarroako Unibertsitate Publikoa

Declaration

"I confirm that this thesis is entirely my own work. All sources and quotations have been fully acknowledged in the appropriate places with adequate footnotes and citations. Quotations have been properly acknowledged and marked with appropriate punctuation. The works consulted are listed in the bibliography. This paper has not been submitted to another examination panel in the same or a similar form, and has not been published. I declare that the present paper is identical to the version uploaded."

Vienna, July 14th, 2014 — Alberto A

Place, Date — Signature

Kurzfassung

Menschen mit Behinderungen sind Teil unserer Gesellschaft. Für manche dieser Menschen können einfache Tätigkeiten des Alltags unüberbrückbare Hürden darstellen - z.B. die Verwendung eines Computers. In vielen Fällen ist die Hilfe weiterer Personen nötig, um diese Hürden zu überbrücken. Unterstützungstechnologien dienen dazu, Menschen mit Behinderung mehr Autonomie und Unabhängigkeit zu geben.

Im Jahr 2010 wurde das "AsTeRICS" Projekt gestartet, mit dem Ziel, ein frei verfügbares Open-Source System zu Erstellung flexibler Assistenzlösungen zu entwickeln. Das AsTeRICS Projekt wurde von der Europäischen Kommission finanziell unterstützt und in Zusammenarbeit 9 internationaler Partnerorganisationen umgesetzt. Im Zuge des AsTeRICS Projektes wurde ein lippengesteuertes Eingabegerät für Computer entwickelt, die sogenannte "Lipmouse". Die Lipmouse dient dazu, einer Person mit Einschränkungen der oberen Extremitäten einen alternativen Zugang zur Verwendung eines Computers, Tablets oder Notebooks zu ermöglichen. Die Lipmouse konnte im Zuge des AsTeRICS Projektes nicht fertiggestellt werden und wurde in der vorliegenden Arbeit verbessert und erweitert.

Die durchgeführten Verbesserungen können in 3 Hauptaufgaben zusammengefasst werden: Im ersten Teil der Arbeit wurde eine portable Version des Lipmouse-Prototypen entwickelt (die originale Version musste via USB-Kabel angebunden werden). Zu diesem Zweck wurden ein Bluetooth-Modul und eine aufladbare Batterie zum bestehenden Design hinzugefügt. Der zweite Teil der Arbeit beschäftigt sich mit der Planung und dem Layout einer Leiterplatte für das Lipmouse-System. Im letzten Teil der Arbeit wurden die Software bzw. Fimware des Limouse-Moduls in folgender Weise verbessert: die Verwendung der neu hinzugefügten Hardware-Funktionen wurde in die Firmware eingearbeitet, und für die Verwendung der Lipmouse im AsTeRICS System wurde ein geeignetes Software Plugin erstellt.

Abstract

People with disabilities are part of the society. However, simple actions in their daily life, such as using a computer become a challenge. In many cases they need the assistance of another person. Assistive technologies help performing these tasks and help people with special needs to live more independently. In 2010, a collaborative project called "AsTeRICS" (Assistive Technology Rapid Integration and Construction Set) was initiated and partly funded by the European Commission, where 9 international partner organisations worked together to develop a free, open-source, flexible and affordable assistive technology tool.

A mouse cursor emulator controlled through the lips was developed for the AsTeRICS platform. This device was called "Lipmouse" and was intended to provide a way of accessing a computer, tablet or notebook to a person with an impairment in her/his upper limbs. This thesis is a continuation and enhancement of the Lipmouse development.

The improvements accomplished can be summarized in three blocks: The first part of this thesis consisted of the development of a portable version of the Lipmouse. The initial version worked via USB cable. For that purpose, the new version incorporates a Bluetooth module and a battery power supply system. The second part was focused on the design of a PCB where all electronic components of the Lipmouse were integrated. Finally, the Lipmouse source code was enhanced in two ways: The firmware of the Lipmouse was modified for supporting the new functionalities described above. On the other hand, a specific software plugin for the integration of the Lipmouse into the AsTeRICS framework has been developed.

Keywords: Mouse emulator, Lipmouse, Assistive Technologies, AsTeRICS

Acknowledgements

First of all, I would like to express my most sincere gratitude to Mr. Dipl.-Ing. Veigl for trusting and believing in me to accomplish this incredible project. It has been an honour for me to work under his supervision. Without his guidance, support and wisdom, it would not have been possible.

A special mention for his invaluable help during the PCB design and the soldering process deserves Benjamin Aigner, MSc. He has my most honest gratefulness.

I should thank the University of Applied Sciences Technikum Wien for accepting me as an exchange student and allowing me to develop this Master Thesis, especially the Embedded System Department for giving me this opportunity.

I also will not forget and thank all the marvellous people I got to meet this year in Vienna. Thanks for being always there.

Table of Contents

1 Introduction

1.1 Assistive Technologies

Society is very complex and each human being has different necessities than the others. A part of the community has difficulties carrying out daily and simple tasks, like using a computer, reading a book, opening a door, getting up stairs, switching on lights, making a phone call, and so on. Such people with special needs require dedicated tools which allow them to equally participate in our society. They need special devices and tools in order to perform their regular activities. These accessories are called "Assistive Technology devices".

Official organisations like the Assistive Technology Industry Association (ATIA), [1], Assistive Technology Partnership (ATP), [2], or the Individuals with Disabilities Education Act (IDEA) American federal law, [3] , define "Assistive Technology" like the field involved in developing "any item, piece of equipment, software or product system that is used to increase, maintain or improve the functional capabilities of individuals with disabilities. Not including a medical device that is surgically implanted or the replacement of such device".

This field has seen a great development in recent years. Despite their efforts, many manufacturers work on particular aspects of an impairment solving only a part of the issue. For example an on-screen keyboard can facilitate the access to a computer for a quadriplegic person, but this application does not help him/her in switching on the light of the room. In addition, each person with disabilities has different necessities than someone else with the same problem. Therefore these devices need to be flexible and adaptable. The lack of versatility makes these solutions inappropriate for creating a single, unique system.

Another problem is the price. In most cases the devices are not affordable by the fact that only a small group of people need them so the market is small compared to mainstream consumer electronics. Accordingly, these gadgets are very expensive or are not easy to get.

There is a diversity of types of impairment with different grades of disability. As a result, the range of available AT solutions is very wide. They can be categorized in four general classes: mobility (including muteness), auditory, vision, and psychology/neuronal (learning disabilities, memory problems, narcolepsy).

This project is considered being part of the computer access field. Its goal is to guarantee computer access for people who have their upper-limbs paralysed, who can only move their lips or for whom a computer input solution based upon lip/mouth interaction is more comfortable than for example an eye gaze system. A computer is not limited to surfing the

Internet like on-line shopping, checking emails and such activities. It is also capable of either interaction or taking control over other systems, e.g. controlling an “intelligent house” or using a wheelchair as well as a radio or TV via remote control. Thus, providing computer access opens a door to an easier, more independent, and more comfortable life for people with special needs.

1.2 State of the Art

There is a range of alternative ways to access a computer apart from the common tools, such as keyboards and mice. Switches are the simplest solution for accessing all types of electronic devices. Many of them are just buttons. There is a great variety of them: dual or multi switches, foot switches, sip-puff switches, wireless switches, and so on as can be seen in Figure 1.1.

(a) Dual switch (b) Bluetooth dual switch (c) Foot switch

(d) Pal Pad switch (e) Sip-puff switch

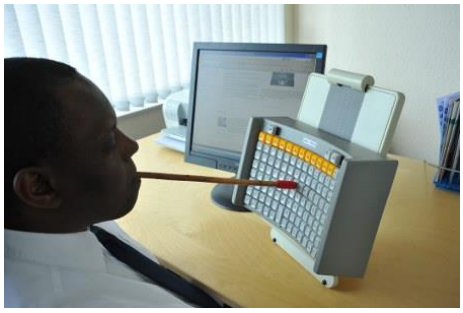
Figure 1.1: Different types of switches

The keyboard combined with the mouse is one of the most important input peripherals for computer access. However, a lot of people with problems in their upper limbs cannot use them. There is a wide spectrum of alternative keyboards, from an ergonomic keyboard to an on-screen keyboard. Some of them are depicted in Figure 1.2.

Another example would be the joystick, which is commonly known as controller for playing video games, but it also can replace a mouse, a switch or a keyboard, i.e. a simple joystick becomes an AT device. Like these previous devices, there are also adapted joysticks for people with disabilities, from a regular hand used joystick to a mouth controlled joystick. Besides computer access, joysticks can be found in wheelchairs or adapted cars. Some examples are illustrated in Figure 1.3.

(a) Ergonomic keyboard

(b) Alternative keyboards



(c) Mouth stick keyboard

(d) Single hand keyboard

(e) On-screen keyboard

Figure 1.2: Different types of keyboards

(a) Simple joystick

(b) Adapted joystick

(c) Mini joystick

(e) Head joystick

(f) Wheelchair with chin joystick

(g) Steering wheel adapted with joystick

Figure 1.3: Different types of joysticks

There are also other products like touchscreens or voice recognition input systems. The touchscreens do not necessarily need the use of fingers; they could also be accessed with a special stick held in the mouth. The voice recognition devices are promising systems, because they can be configured in many ways to perform different actions with a computer.

Focusing on the mouse cursor solutions, there is also a great variety of these devices depending on the users' necessities. As depicted in Figure 1.4, there are ergonomic mice, trackball solutions, head or eye controlled, switch adapted, touchpads, and so on.

(a) Roll-mouse (b) Foot mouse (c) Head controlled

(d) Button-mouse (e) Touchpad

Figure 1.4: Different types of mice

(The images of switches, joysticks, adapted keyboards and mice were taken from the following AT vendor web pages: Infogrip [4], EnableMart [5], AbleData [6], Attainment Company [7], Enabling Devices [8], and AbleNet [9]).

However, the Lipmouse user profile is very specific. It was designed for individuals who cannot move their arms and head-camera mouse solutions do not fully satisfied them. Nowadays, similar solutions can be found on the market. An example would be the IntegraMouse Plus from LIFEtool [10], the Jouse2 or the Jouse3 from Compusult [11], the Lipsync from Adaptive Computer Control Technologies Inc [12], the QuadJoy 3 from QuadJoy [13] or the TetraMouse XS or TetraMouse XA from TetraLite Products [14]. A picture of each one is shown in Figure 1.5. These solutions are characterized by a plastic mobile stick that the user controls with the lips. Thus, the movement is reflected in a movement on the computer screen cursor. Additionally there are clicking features. Many of these devices incorporate either a sip and puff sensor to emulate the click or just a simple switch. Most of them are wired and plugged through a USB cable. Nevertheless, the IntegraMouse device has a wireless version. Several of them have replacements for the

mouthpiece due to saliva and bacteria accumulation. The QuadJoy costs approximately 800$ and the IntegraMouse nearly 2500$.

(a) IntegraMouse, [10]

(b) Josue 2, [11]

(c) Lipsync, [12]

(d) QuadJoy 3, [13]

(e) TetraMouse XA, [14]

Figure 1.5: Different types of mouth controlled mice

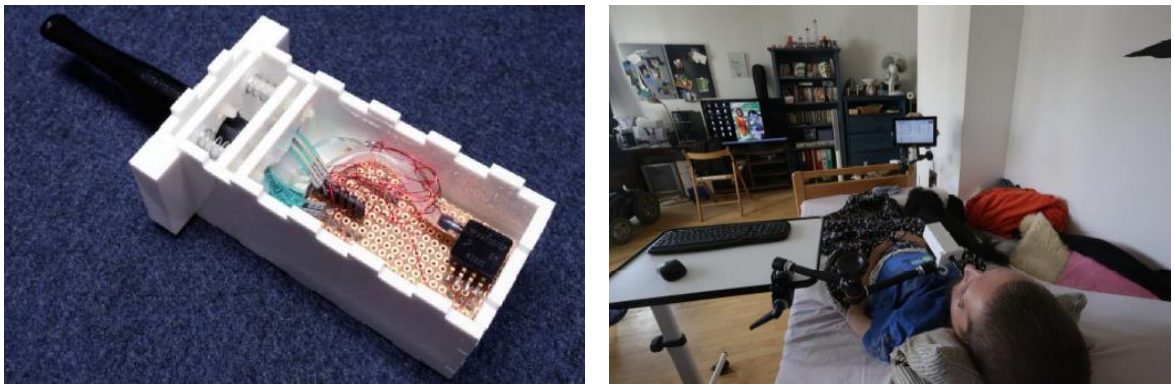
1.3 First Version of the Lipmouse

A first version of the Lipmouse was developed in 2013. The device was especially designed for one person, although it had multiple future possible users. The person mentioned can only use his facial muscles, so he needed a dedicated and specific solution. The device and the person using it are depicted in Figure 1.6

Although devices like IntregraMouse were available on the market, the IntregraMouse was designed for people who can move their head and the strength required for moving the mouth pieces is higher than the Lipmouse. In addition, the sensitivity is lower. (Reference [15])

The current Lipmouse has a mouth stick made from plastic that is connected to an acrylic glass plate. When the mouth piece is moved with the lips, the plate accordingly follows the movement. It transmits the force through four screws to another plate where the pressure sensors are placed. The force between the plates is adjusted with four springs. The tail of the sensors is wired with air wires to the Teensy microcontroller (see Section 2.5.1) via a voltage divisor made with 10kΩ resistors. The mouth stick is hollow inside allowing the air

flux to the sip and puff sensor. The stick end is connected to the sensor through a plastic tube. The values measured in the five sensors are transmitted to the microcontroller, which processes and sends them to the computer. The electrical components are mounted and soldered in a perfboard. The board and the plates are covered and fixed to an acrylic enclosure.



(a) The first prototype build of the Lipmouse

(b) A person testing the first prototype in his home

Figure 1.6: The first prototype of the Lipmouse

In order to be operative, this device needs a special software tool, the AsTeRICS framework, described in the Sections 2.1 and 2.2.

1.4 Motivation

The purpose of the Lipmouse prototype was providing a device to emulate a computer mouse for people with severe motor conditions who can only move the lips. The computer mouse is one of the most important peripheral devices for interacting with computers. Therefore, these people are not able to use a computer. Nowadays many tasks of our daily life involve using a computer: work, media, communication via email or videoconference, checking bank accounts, shopping and so on. Therefore it is very important to develop an AT device to facilitate computers access also for people with severely reduced motor capabilities. In addition, the Lipmouse could be also used together with a tablet or a notebook.

This thesis is a continuation of work in the Lipmouse project, accomplished by Robert Haderer, (Reference, [15]), as it was described in the previous section. The particular targets were improving some parts of the written software code, designing a portable solution (a wireless communication plus a battery powered system), implementing power saving techniques, designing a printed circuit board (PCB), and testing it. The modification of the existing enclosure, anti-bacteria and saliva membrane protection for the stick of sip and puff sensor, and other enhancements will be achieved in a future project.

2 Methods

In this section the different tools for the development of the project will be presented.

2.1 Introduction to AsTeRICS Project

This work builds upon a research initiative which has been started in 2010 and was co-funded by the European Commission: the “Assistive Technology Rapid Integration & Construction Set”, abbreviated AsTeRICS, [16]. AsTeRICS is an international free open-source project which allows creating customisable tailored Assistive Technologies (AT) solutions. This system provides an alternative way of interacting with electronic devices like a PC, a tablet, a smartphone, a TV remote control or a home environment control for people with disabilities.

Figure 2.1: AsTeRICS logo (Source: [16])

The AT market is still very small and narrow. Many solutions just satisfy a single specific purpose. They cannot be combined or adapted for other similar situations. In fact, two users with the same impairment may require different solutions depending on their wishes or needs. Moreover, the available solutions are usually expensive. The AsTeRICS platform was developed in order to address all these issues.

AsTeRICS can combine the simplest input systems like switches, buttons and adapted keyboards with novel methods such as brain computer interfaces, head- and eye tracking, inertial measurements and proximity sensors under a unique system covering multiple scenarios.

The measurements of these sensor modules are delivered to an embedded platform where they are processed. Depending on the configured solution, the embedded platform sends a response over the corresponding actuators to perform an action or shows it on a display. There is also a wide range of actuators that allow the emulation of a keyboard, phone calls, a stereo control or a KNX home environment regulation.

In summary, AsTeRICS is endowed with a high level of flexibility, which makes it possible to develop high adapted solutions for the users’ necessities in an economic way.

2.2 AsTeRICS Platform

The main component of the AsTeRICS framework is the computing platform, where the AT application is hosted and executed. For this purpose, a specific OSGi based middleware

was created, the AsTeRICS Runtime Environment (ARE). The Open Services Gateway Initiative (OSGi) defines a set of specifications in order to create a dynamic modular Java system. The ARE model is composed of smaller independent sub-blocks, which represent the sensor-, the processor- and the actuator-modules. The performance of each part is independent, taking advantage of the modularity and the flexibility that OSGi provides. The ARE loads and deploys the entire configured model. It acquires the sensors data, processes it and generates the actuators response. Any device running a Windows operating system, like a PC or a tablet can be used to host the ARE.

The models are created with the AsTeRICS Configuration Suite (ACS). It is a graphical model based editor. The developer establishes which sub-blocks will be used for the model, its properties and the relationship between sub-blocks. After the creation of a model, the ACS sends the model to the ARE. The communication runs over a TCP network connection, so the ACS and the ARE can operate in the same device or in different devices which are connected through the Internet. Therefore, a developer can configure a users' model from distant locations. Moreover, the ACS can receive information and modify some properties during the ARE's execution.

The ACS also provides an auto-guided tool for the creation of these sub-blocks called "plugins". The plugin is the software part that interacts directly with the actuator or the sensor. In addition, some processing functions are encapsulated in these sub-blocks. This system allows building complex and customizable solutions only with dragging and dropping the blocks and interconnecting them. The user does not need to know how these blocks work internally, in the same way as other model based editors like LabVIEW or Simulink work.

The external hardware modules, i.e. the sensors and actuator, must implement a special interface called Communication Interface Module (CIM) which allows the communication with the ARE, described in Section 2.2.3. These devices are connected to the ARE via USB, Bluetooth, ZigBee or other communication systems implementing this protocol in their upper layer. Figure 2.2 shows an overview of the AsTeRICS framework.

2.2.1 AsTeRICS Runtime Environment

As mentioned, the ARE is the core of the AsTeRICS software, where all sensors, actuators and processors source code is executed. The ARE was built in Java and needs a Java Virtual Machine installed in the embedded platform. It requires at least JAVA 1.7 (JDK/JRE 7).In most cases the embedded platform is a variant of the Windows operating system. A basic version of the ARE already runs on Linux and for other operating systems, a Windows virtual machine could be installed and runs over it. The OSGi framework and the ARE middleware are responsible for loading and deploying the model designed in the ACS as plugins and their interconnections. Some modules may need additional software (i.e. drivers) for working. They may require a legacy code and the Java Native Interface can be

used for the interoperability. The ARE coordinates the CIM modules through the CIM protocol: it detects the virtual COM port or a serial interface, identifies the CIM, manages and controls the exchange of data between the module and the ARE.

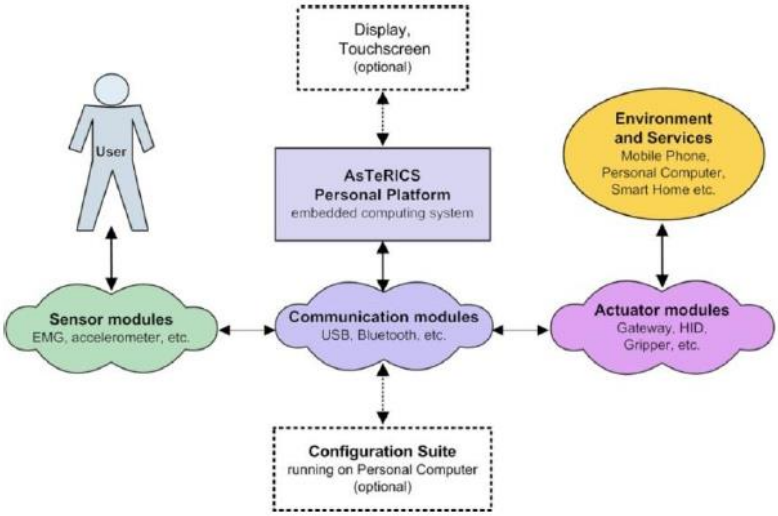


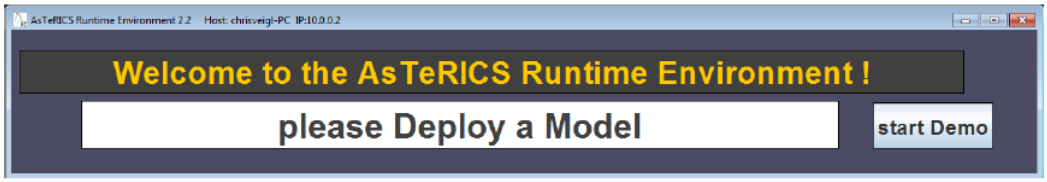
Figure 2.2: AsTeRICS system outline (Source: [17] , page 7)

Furthermore, the ARE disposes a desktop front panel, which allows the user to interact with the system. For example: pressing a button to begin the execution, pause or stop it. It can also show the status of the ARE: disconnected, connected, synchronized, running and paused. Depending on how the model was configured and which GUI plugins it contains, the ARE can also display the results of calculations, the signals acquired, messages, etc. to inform about the execution process. Two examples of front panels can be seen in Figure 2.3. Image (a) corresponds to the front panel, showed at the beginning when the ARE is launched. Picture (b) shows the front panel during the execution of a model. The user can change some parameter values via this panel and modify the behaviour of the model. For instance, during a mouse emulation model, it is possible for the user to change the sensitivity or speed of the cursor. If the user needs more accuracy in his/her movements, he/she can reduce the velocity during execution. In addition, the user may need special functions, such as double click or hold the click for dragging and dropping items. This can be done by placing buttons and by pushing them during the execution in order to perform the associated action. (References [17] and [18]).

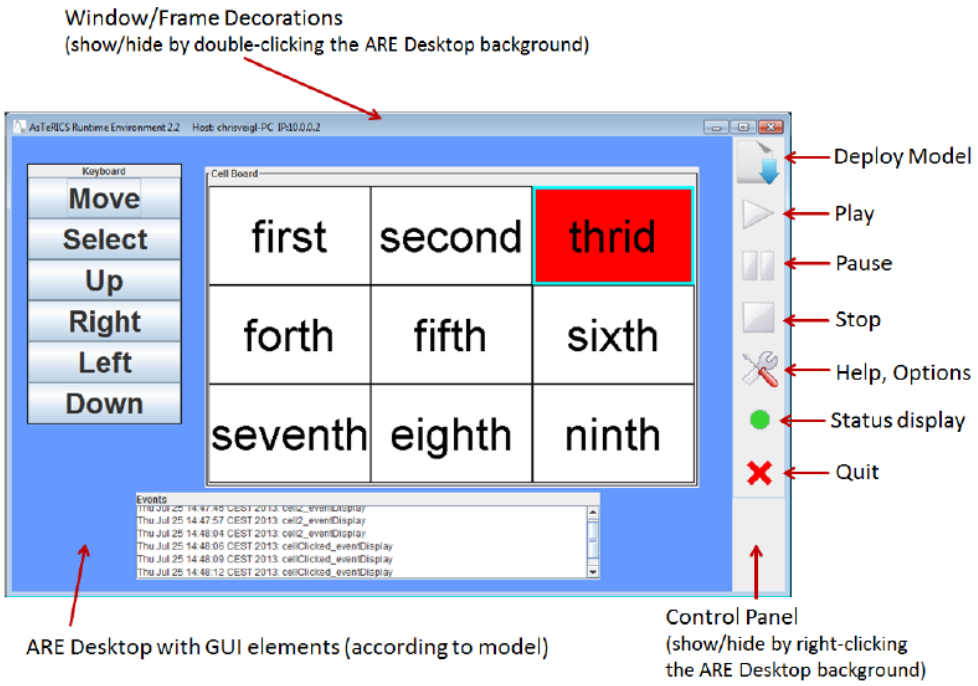
2.2.2 AsTeRICS Configuration Suite

The creation and configuration of the ARE's models are made with the ACS editor. All sensor-, processor-, and actuator-modules available can be selected, configured in their parameters, or interconnected with other plugins, defining the relationship between them. The ACS can be connected directly to the ARE. Thus, the ARE can be controlled from the ACS. The designed model can be uploaded to the ARE, or the current loaded model can be downloaded to the ACS in order to save or modify it. The ACS can launch, pause,

resume and stop the model execution in the ARE. Moreover, the status of the ARE is displayed inside the ACS. Errors, logging information and status messages are sent from the ARE to the ACS, allowing debugging the model.



(a) At the beginning of the execution



(b) During execution of a model

Figure 2.3: ARE front panel (Source: [17], page 7 and 9)

Each plugin has one or more ports classified as input-, output-, event listener- and event trigger-ports. The input- and output-ports are used to exchange data between the modules. The event listener- and event trigger ports are employed to control and synchronize. When a defined condition is reached in one module, it can send a signal to inform another plugin to perform an action. One of the aims of this thesis was the creation of a specific plugin for the Lipmouse, see Section 3.4. Until now, the previous prototype used the Arduino plugin, for implementing the proof-of-concept with this generic microcontroller. An example of a plugin is shown in Figure 2.4

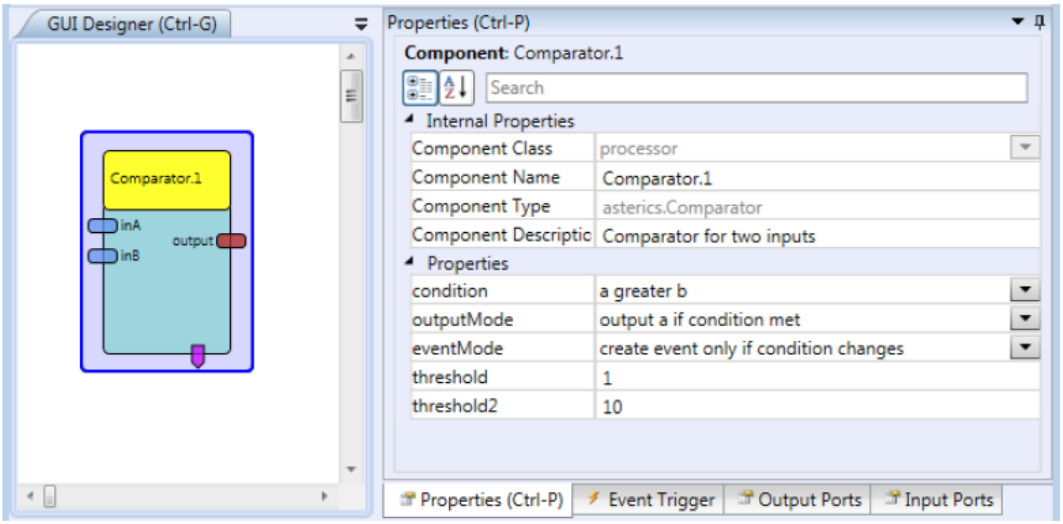


Figure 2.4: Module example (Source: [17], page 21)

The output ports (red) can only be connected with input ports (blue); event trigger ports (purple) can be connected with event event listener ports (green). After each connection, the established channel must be configured accordingly. For example, if the input signal must be synchronized with other incoming signals or an event trigger signal is paired with an event listener in the other module. Moreover, the default module properties can be defined. Many of them can be modified when the model is running. An example of a finished model is depicted in Figure 2.5.

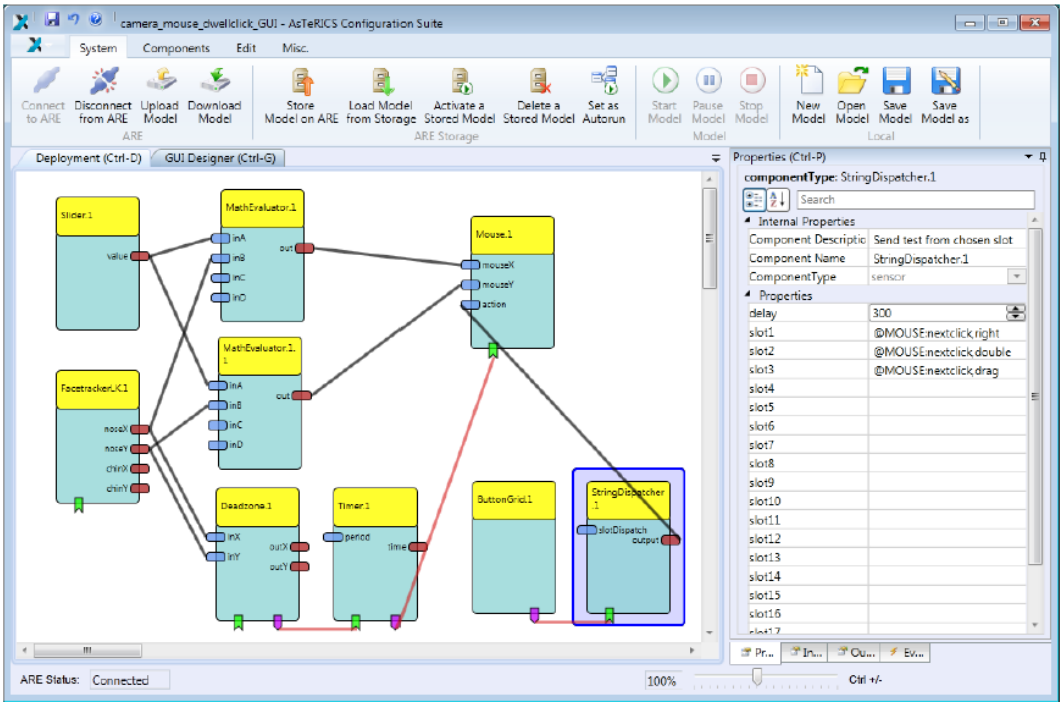


Figure 2.5: Example of a finished model (Source: [17])

Apart from the "Deployment" area in the ACS editor, there is another tab called "GUI Designer" where the ARE window layout shown during runtime can be set. The size and the appearance of the displayed elements can be arranged and adjusted, depending on the users' preferences.

For more specific information about the model creation consult [17].

2.2.3 Communication Interface Modules Protocol

As previously mentioned, the ARE model plugin and the firmware of the device must implement a protocol named Communication Interface Modules (CIM). This provides a way of exchanging data between the running firmware in the module and the ARE. The protocol runs over a serial communication, such as (virtual) COM port using USB, UART, RS-232, etc. If the device cannot support it or the developer does not want to implement it, the plugin can use other external protocols like the raw port implementation, providing the standard I/O stream classes of Java to communicate with the device. The devices which implement the CIM-protocol are simply called CIMs.

The ARE-middleware provides a separate interface called CIM Communication, where all CIM communication services for device communication are managed.

Each plugin has its own commands, called features. Each feature can be read to know the state of the feature and/or written to perform an action. For instance, a CIM can have a temperature sensor, which has to read the temperature value (the feature) periodically. The ARE processes the value and sends it for example to another plugin such as a threshold detector. Thus, if the temperature reaches a critical level, the threshold detector sends a signal to the CIM, and writes to another CIM device to switch on the fan (another feature). An example of this feature list is on Figure 2.6. It is the feature list of the HID-actuator CIM, a module which can emulate mouse/keyboard/joystick devices on USB HID level.

CIM-ID	Feature-address	Access	Descritption	Data
0x0101: HID actuator version 1	0x0000	r	Unique serial number	4 bytes
	0x0001	w	MOUSE x/y pos (relative change)	4 bytes: xxyy
	0x0002	w	MOUSE buttonstate	1 byte: Bit 0=left click, Bit 1=right click, Bit3=middle click
	0x0003	W	MOUSE wheel	1 byte: wheel displacement
	0x0010	w	KEYBOARD keypress	2 bytes: keycode, modifier
	0x0011	w	KEYBOARD keyhold	2 bytes: keycode, modifier
	0x0012	w	KEYBOARD keyrelease	----------
	0x0020	w	JOYSTICK joy1pos-analog	4 bytes: xxyy
	0x0021	w	JOYSTICK joy2pos-analog	4 bytes: xxyy
	0x0022	w	JOYSTICK joy3pos-digital	1 byte: Bits 0-3: left/right/up/dwn
	0x0023	w	JOYSTICK joybuttonstate	2 bytes: Bits 0-9: button pressed 0/1

Figure 2.6: Example of a CIM feature list, the HID actuator (Source: [18], page 56)

The ARE must know which features are implemented by the device. According to this, when the device is plugged in, the first step is its identification. Each CIM has a unique identification number and a unique serial number. The identification number identifies the type of CIM such as an accelerometer, an HID device, a Lipmouse, an Arduino microcontroller, and so on. The serial number identifies the device. It is like the MAC address of the device. The ARE sends a packet making a request and the device must answer it with a reply packet. The ARE acts like a master and the device as a slave, only answering when the ARE sends a reply. However, there is one option for sending periodical information from the CIM to the ARE without an ARE request packet. There are 6 pairs of request/reply packet types and one more type for these periodical updates. These categories are shown in Figure 2.7.

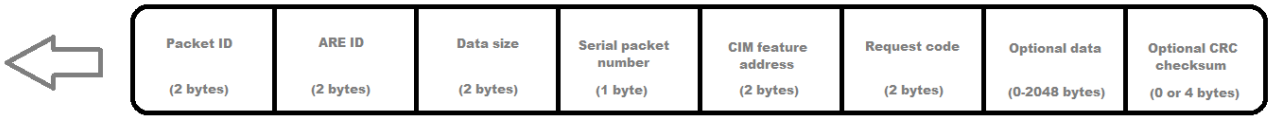
Request / *Reply* code	**Direction**	**Description**	**Expected Data**
0x00	ARE→CIM	request feature list	-
0x00	*CIM→ARE*	*reply feature list*	*list of supported features (eg. 8 bytes for 4 feature addresses)*
0x10	ARE→CIM	request write feature	bytes according to feature
0x10	*CIM→ARE*	*reply write feature*	*bytes according to feature*
0x11	ARE→CIM	request read feature	bytes according to feature
0x11	*CIM→ARE*	*reply read feature*	*bytes according to feature*
0x20	*CIM→ARE*	*event reply*	*bytes according to feature*
0x80	ARE→CIM	request reset CIM	-
0x80	*CIM→ARE*	*reply reset CIM*	-
0x81	ARE→CIM	request start CIM	-
0x81	*CIM→ARE*	*reply start CIM*	-
0x82	ARE→CIM	request stop CIM	-
0x82	*CIM→ARE*	*reply stop CIM*	-

Figure 2.7: Different types of CIM packets (Source: [18], page 55)

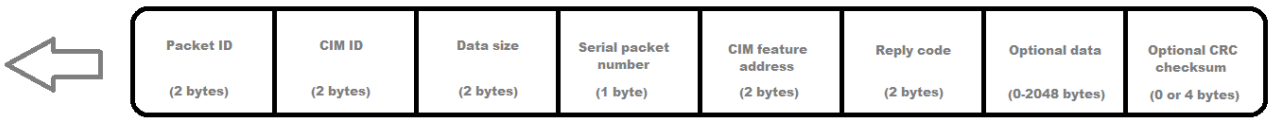
The reply/request code identifies the type of packet. Thereby, the ARE and the CIM know which kind of information is contained in the packet, the data format and how to manage the data according to the packet's type. The entire structure of the CIM protocol packets is depicted in Figure 2.8.

The packet ID is always the same combination of two bytes. It identifies the beginning of a CIM packet. Thus, the ARE and the CIM know when a packet is being received. It also has a synchronization purpose. These bytes are 0x4054, in ASCII format "@T". The next field is the ARE ID or the CIM ID. The ARE ID identifies the ARE version. If the CIM detects that the version of the ARE is below the minimum required version for compatibility, it can refuse to reply to some features. The CIM ID identifies the CIM type and its version. The data size notifies to the ARE/CIM how many bytes are in the optional data field, because it can have different amounts. The minimum value is 0 (0x0000) and the maximum is 2048

(0x0800). If a device receives a value out of this range, it will consider the packet as defective. All request packets are enumerated sequentially with a packet number, which is incremented in each request. The reply packets have the same number as their corresponding request, identifying each request with its reply. As the event replies do not need a request, they have their own sequence. The request/reply packets sequence ranges from 0x00 to 0x7f and the event replies range from 0x80 to 0xff. The CIM feature address identifies the corresponding CIM feature. The request/reply field is subdivided in two parts. The high-byte identifies the mode in the ARE-to-CIM packets and the error status in the CIM-to-ARE packets. The mode informs if a CRC checksum is implemented or not, and the error status reports to ARE if an error was detected by the CIM and which type of error (lost packet, invalid ARE version, CRC mismatch, and so on). Despite the error detection, the ARE does not resend faulty packets. The low byte consists of the request/reply code. The optional data is the data sent according to the feature and the desired action to perform. The optional CRC checksum is only added in the respective mode. The minimum size of a packet is 11 bytes (without the optional fields) and the maximum size is 2063 bytes. The fields with two or more bytes are stored in little-endian format. The complete information about the CIM protocol is in the Section 5 and 6 of the AsTeRICS Developer Manual, [18].



(a) ARE-to-CIM packet



(b) CIM-to-ARE packet

Figure 2.8: CIM packets field structure

2.2.4 Plugin Design Tool

In this project, a specific plugin for the Lipmouse was developed. The plugin's internal structure contains information of where the code is stored, the descriptor files, the directories paths or the programming of common part as the output, inputs functions (the source code skeleton) can be complex to accomplish. For creating it, the ACS has a tool

that helps the developer: the Plugin Creation Wizard. Figure 2.9 shows a screenshot of the Plugin Creation Wizard.

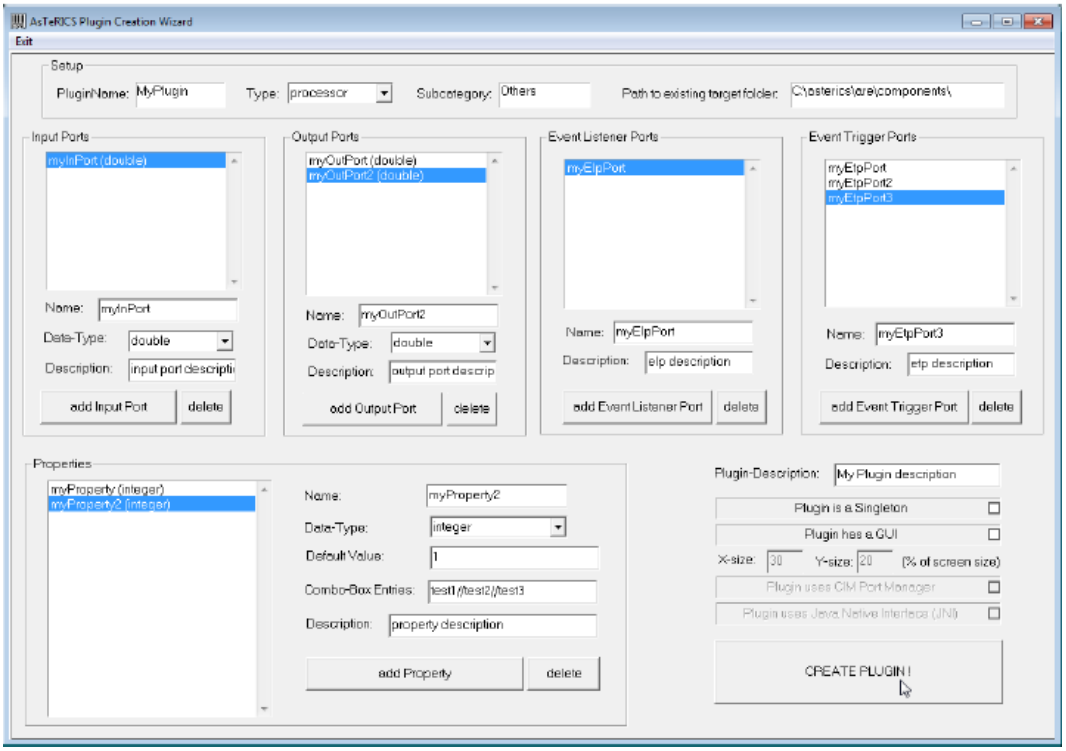


Figure 2.9: The Plugin Creation Wizard (Source: [18], page 20)

The developer has to fill the gaps according to its necessities and the whole structure of folders, files and the source code skeleton are created automatically. Then the developer only has to program its own functions and features in the main file.

After the creation of a new plugin, the last step is activation and registering the plugin in the ACS/ARE repository for being used. This can be done manually or using the Plugin Activation Tool.

2.3 Programming Tools: AVR Studio and Eclipse

The Lipmouse has an integrated Atmel microprocessor, the AT90USB1286. This microprocessor belongs to the AVR family implementing a RISC architecture. For Windows platforms there is an open source software tool that allows application development for this series of microprocessors, the WinAVR ([19]). This project provides the compiler (avr-gcc), the programmer (avrdude), basic libraries (avr-libc), the assembler (avr-as), the make utility for the source code generation, the debugger (avr-gdb), and much more tools. The AVR Studio is an Integrated Development Environment (IDE) which uses the WinAVR tools. All source files and header files needed for the Lipmouse's firmware were created with the AVR Studio 4. The AVR Studio facilitates the task of programming and offers additional

functionalities, such as code debugging and simulations. In addition, the microprocessor can be flashed using supported programmer tools. The AVR Studio also allows loading files that contain definitions and functions to facilitate the programming task, such as the use of interrupts, I/O peripherals, the EEPROM memory management, sleep modes, mathematical functions, types definitions, and so on.

Figure 2.10: AVR Studio logo

After programming the code, it must be compiled. If the compilation is successful, the WinAVR GCC will assemble and link the files. Then it will create the *.hex* file for the microprocessor. Before the compilation and the building process, some parameters must be specified: the frequency used by the microcontroller CPU, if the project uses a Makefile or not, if the compiler has to make any optimization or not, the type of device (AT90USB1286) and the path where the output files will be stored. The last step is downloading the firmware to the microprocessor.

The first setups where made using a Teensy++ 2.0 microcontroller. This board has its own program for flashing and rebooting the microprocessor. However, for the final prototype that only uses the AT90USB1286 chip, the Atmel FLIP tool was employed. It is a tool for flashing the microprocessor through a USB, CAN or RS232 interface. The chips must contain a special bootloader for this process and if it is erased or the CPU does not jump to the bootloader section, the firmware cannot be actualized. Moreover, there are other options for downloading the firmware to the microcontroller as via SPI or JTAG interfaces. The JTAG also provides on-chip debugging and allows changing the fuses values of the microcontroller. With the FLIP or the Teensy flashing program, the fuses cannot be modified. Therefore, the final prototype board incorporated a JTAG interface. For that purpose, the AVR Dragon programmer was used. It is a development tool from Atmel that allows programming an AVR microprocessor using AVR Studio. It has various types of interfaces supporting different ways of programming, like In-System Programming or JTAG. Figure 2.12 shows a picture of the AVR Dragon used for programming the fuses.

As seen in the previous section, the ARE was programmed in Java and runs in a Java Virtual Machine. One of the objectives of this thesis was creating a new plugin for the ARE. For this task, the Eclipse software was employed. Eclipse is an IDE that can be used for programming in Java and other languages and it facilitates the ARE framework management, [20]. The ARE folder can be loaded including all subfolders and files in order to provide an easy access to the all ARE source code.

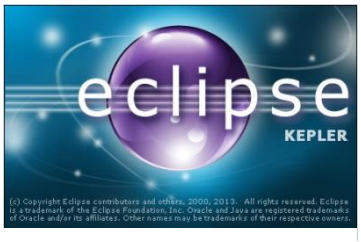


Figure 2.11: Eclipse logo (Source [20])

After the plugin creation through the Plugin Creation Wizard tool, the subfolders structure of the plugin will appear in the workspace containing the descriptor files and the source code skeleton for the plugin instance which has been created by the Plugin Creation Wizard as described in Section 2.2.4 (*ARE**components**sensor.lipmouse*). From this point, the plugin programming process can continue using Eclipse.

The descriptor files: *build.xml*, the bundle*_descriptor.xml* and the *manifest.mf* can also be edited from Eclipse. The Eclipse provides a debugging functionality that can be used for debugging the ARE and all plugins. The last step is compiling the project and building it. After that, the plugin is ready for operation.

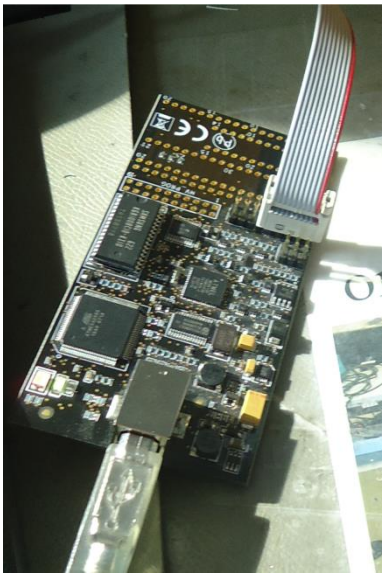


Figure 2.12: The AVR Dragon

2.4 PCB Design: EAGLE CAD Software

For designing the Printed Circuit Board (PCB), there was the need of a special computer-aided design (CAD) software. For this project, the free version of EAGLE CAD [21] was chosen. EAGLE CAD has three main tools: the EAGLE Schematic Editor, the Layout Editor and the Autorouter functionality. To create the schematics and the board layout, EAGLE procures a set of libraries with the most common electrical components from many manufacturers. Apart from the default libraries, there are many projects on the Internet and these components can be downloaded and integrated in the EAGLE repository. The element14 community (from Farnell) has a wide list of libraries with many components. If a component is not available in any library, EAGLE provides a tool for creating new libraries, allowing the user to create its own components. The EAGLE free version has some restrictions: the board cannot have a size over 100 x 80 mm and no

Figure 2.13:EAGLE logo (Source: [21])

more than two layers (top and bottom). Though, it was enough for designing the Lipmouse board.

Returning to the Schematic Editor subject, the first step is to add all essential components for the project to the workspace. Then the electrical connection between them must be drawn, including the ground and supply traces. The appropriate names and the values for the components must be defined. When the electrical circuit is finished, the Design Rules for the circuit must be checked. EAGLE has a checking tool to test if all connections are right to satisfy the design rules (DRC – design rule check).

After the schematic design, the next task is the layout process. The board dimensions must be defined according to the board requirements. Afterwards, the components must be placed in their exact positions on the correct layer (top or bottom). The following step is routing all components. The physical traces must be drawn in the board layout. The process must be realised regarding to some consideration, such as avoiding right angles traces, the thickness of the traces (for example the supply traces must be thicker because they carry more power), placing the decoupling elements as close as possible to their corresponding components, providing an adequate heat dissipation, and so on. A good recommendation is to cover the entire board with a ground plane. In the same way of the schematic, the last part is checking if the design satisfies the design rules.

When the design was finished, the last step was to send the EAGLE design to a board manufacturer for the elaboration of the physical board.

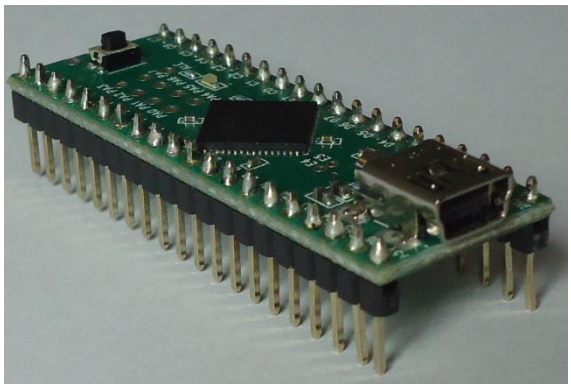
2.5 Electrical Components

For the development of this thesis, some integrated electronic components were used for creating the new Lipmouse prototype. This section will introduce and describe these elements and their most important characteristics.

2.5.1 Teensy++ 2.0 and AT90USB1286

A microcontroller was necessary to sample the signals of the sensors, convert them to the digital domain (with an analog-to-digital converter), process the data and interact with the embedded platform. The first prototype employed the Teensy++ 2.0 microcontroller from PJRC, [22]. This microcontroller uses as core an AT90USB1286 microprocessor from Atmel. A picture of both can be seen in Figure 2.14. The Teensy was a good option because it facilitates the access if a protoboard or perfboard solution is used. Besides, it provides the USB connector, a 16 MHz crystal for clocking the system and a reset button. Moreover, it has a solder pad for a 3.3 V regulator (MCP1825 from Microchip) to optionally supply the board with 3.3V instead of 5 V. However, for this new version, a PCB was built and it was considered that it was better to use the AT90USB1286 chip without any kind of

evaluation PCB. Anyway, the main characteristics of this microprocessor are reported in Table 2.1.



(a) Teensy ++ 2.0 board

(b) AT90USB1286 microcontroller

Figure 2.14: The eval board and microcontroller employed during the project development.

Parameter	Value
Flash (Kbytes)	128
Pin Count	64
Max Operating Freq (MHz)	16 MHz
CPU	8-bit AVR
RAM (Bytes)	8192
EEPROM (Bytes)	4096
Max I/O Pins	48
External Interrupts	16
SPI	1 interface
TWI (I2C)	1 interface
UART	1 interface
USB	1 interface, Full Speed, USB 2.0 compliance
ADC channels	8, 10-bits resolution, 15 ksps
Timers	2 x 8-bit and 2 x 16-bit
Comparators	2
Temperature Range (°C)	-40 to 85
Operating Voltage Range (V)	1.8 to 5.5
PWM channels	9
Debug Interface	JTAG

Table 2.1: Main features of AT90USB1286 microprocessor (Source: [23])

2.5.2 Bluetooth 4.0 Low Energy Devices

The communication between the microprocessor and the software running in the PC in the first prototype was done via USB. However, in this project a wireless solution was incorporated in order to elaborate a portable solution. The technology chosen was Bluetooth 4.0 Low Energy. Depending on the users' necessities, the Lipmouse could be used either connected with the USB cable or with the Bluetooth wireless link.

Two devices were used for the Bluetooth connection, both from JNHuaMao Technology Company, [24]. One was the HM-10 BLE 4.0 module, [25]. The other device employed was the HM-15 BLE USB dongle. The dongle is similar to the HM-10, but it has already soldered a male USB connector and it is ready to use (see Figure 2.15). The key features of these devices are:

- RF transmission power of -23dBm, -6dBm, 0dBm or 6dBm.
- 6kBps speed, both synchronous and asynchronous.
- Master (central) or slave (peripheral) role configuration.
- Authentication and encryption security.
- Point-to-point serial communications (UART and/or USB).

Moreover, the module supports a "sleep mode" in slave role. The module needs to be supplied with +3.3V and 50mA. The dongle incorporates a LED that blinks when it is not paired and lights when it is paired. The module has an output pin configured for this purpose, but the LED has to be provided from the user. This way, it can be quickly noticed when there is a problem with the connection establishment.

(a) HM-10 BLE 4.0 module (Source: modified from [25])

(b) HM-15 BLE USB dongle (Source: [26])

Figure 2.15: The BLE devices employed for the wireless communication

The main component of both devices is the chip CC2540 form Texas Instrument. This chip integrates a large part of the modules needed for developing BLE applications: a radiofrequency transceiver, an 8051 microcontroller unit, an in-system-programmable flash (128 KB or 256 KB), an 8 KB SRAM, a 12-bit Analog-to-Digital Converter (ADC), a UART and a USB interfaces, three general purpose timers, among other characteristics. This integrated circuit only needs a few external components as the antenna or a crystal oscillator to build a BLE device. Furthermore, a vendor specific firmware was programmed in the chip developed by JNHuaMao that implements the BLE protocol stack.

2.5.3 Sip and puff Sensor: MP3V7007

The Lipmouse was built to be controlled only with the lips and the mouth. Therefore, the clicking function was accomplished with a sip and puff sensor. It needs a mouth piece for inhaling and exhaling the air and a plastic tube for conducting the air flux from the mouth piece to the sensor. The requirements for the sensor were being capable of measuring negative and positive pressure values and using a voltage supply of 3.3 V. The pressure range should have been approximately between -5 and 5 kPa. Unfortunately there was a lack of available solutions on the market, which would fit these conditions. There were many different pressure sensors, which were able to measure the difference of pressure between two inputs. The first Lipmouse prototype used the MPXV7007GP device (with 5V supply voltage) from Freescale Semiconductor. Nevertheless, for this project, the MP3V7007GP device was used, which has the same pressure range between -7 and 7 kPa, but features a voltage supply of 3.3V as needed. Figure 2.16 shows a picture of the utilized device. The most important characteristics of this sensor are illustrated in Table 2.2.

2.5.4 Force Sensors: FSR400

Mouse cursor movements are performed via the mouth piece: The user moves this piece with the lips. The movement is transmitted to a small plastic plate, where four pressure sensors convert the mechanical movements into electrical signals. The sensors chosen were FSR400 physical force sensors from Interlink Electronics. These are basically resistors, which decrease the resistance according to the force applied to sensor. The range is approximately between almost infinite (i.e. an open circuit) to 1 kΩ. For a proper operation they must be connected with another resistor to form a voltage divider configuration. A variation in the resistance produces a change in the voltage that can be converted into digital data by the microcontroller's ADC peripheral. Therefore, the measured values depend on the used resistor and the voltage supply. The sensors are depicted in Figure 2.17 and the main characteristics are summarized in Table 2.3.

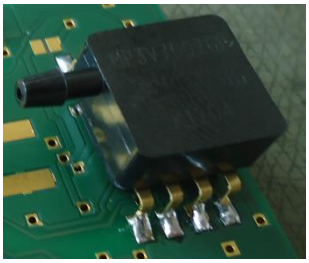


Figure 2.16 MP3V7007GP sip and puff sensor

Parameter	Min	Typ	Max
Pressure Range (kPa)	-7	-	7
Supply Voltage (V)	2.7	3.0	3.3
Output Voltage (V)	0.2	-	2.8
Supply Current (mA)	-	7.0	10
Accuracity (%)	-	-	±5.0
Sensitivity (mV\kPa)	-	171	-
Response Time (ms)	-	1	-
Warm-Up Time (ms)	-	20	-
Temperature Range (°C)	0	-	80

Table 2.2: MP3V7007GP main features (Source: [27])

If it would have been possible, a sensor with shorter tail would have been used, because the tail had to be bended, so it could be attached to the plate and also hindered the assembly inside the enclosure. Nevertheless, it was not possible to find such shorter sensors during the realization of this project.

Figure 2.17: FSR400 pressure sensors

Parameter	Value
Force Sensitivity Range	0.2 N to 20 N
Force Resolution	Continuous (analog)
Non-Actuated Resistance	>10 MΩ
Rise Time	3 μs
Operating Temperature	-40 °C to 85 °C
Durability	10 million actuations

Table 2.3: FSR400 main features (Source: [28])

2.5.5 Battery

A portable solution requires also a portable power supply. In other words, the new Lipmouse prototype needs a battery system. There are many types of batteries from diverse chemistries with different performances for different applications. The discussion of the election will be given in Section 3.2. In this subsection, the selected battery will be described briefly.

The aim was to power the system as long as possible. This means the battery should have a large capacity. Higher capacity means bigger in size. One of the purposes of the Lipmouse project was to keep the physical dimensions as small as possible. Therefore, the space inside the enclosure was limited. Due to its dimensions, a 1000mAh Lithium-Polymer battery was chosen. Its features are illustrated in Table 2.4 and a picture of it can be seen in Figure 2.18.

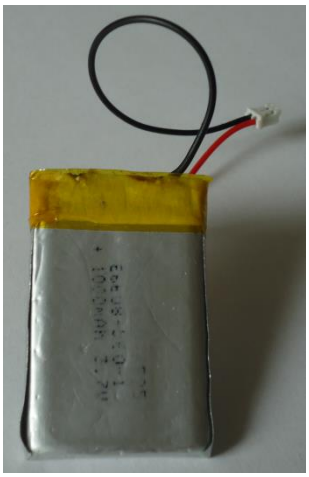


Figure 2.18: The battery employed for powering the Lipmouse

Parameter	Value
Nominal Capacity	1000mAh
Nominal Voltage	3.7 V
Charge Current (standard)	200 mA
Maximum Charge Current	1000 mA
Charge Cut-off Voltage	4.20±0.03 V
Discharge Current (standard)	200 mA
Maximum discharge Current	2000 mA
Discharge Cut-off Voltage	2.75 V
Impedance	≤300 mΩ
Operating Temperature	-20 °C to 60 °C
Weight	20 g.
Dimensions	6 mm x 34 mm x 50 mm

Table 2.4: Battery specifications (Source: [29])

This battery provides a built-in protection circuit to avoid overcharge and over-discharge. The voltage range should be between 2.75 – 4.25 V. Moreover, this circuit has a short circuit protection. However, the battery is charged via a USB connector (5 V). Therefore, it is required another circuit to adjust the voltage from 5 V to battery's voltage. This function is accomplished by the charger as will be explained in Sections 3.2 and 3.3. Moreover, nowadays chargers regulate the charging voltage and integrate other protection features

as overcharge and over-discharge. Hence, this built-in protection circuit is not necessary and any single cell li-polymer battery can be used to power the system.

2.5.6 Charger

A battery requires a charging solution. When the battery is depleted, it must be charged. There are many commercial electronics devices in which the user must remove the battery and connect it to an external charger. It would not be practical for the Lipmouse user to retire the enclosure, to disconnect the battery and to plug it into an external charger. Therefore, the idea was to integrate the charging circuit with the rest of the electronic components. The aim was to build a device which allows charging the battery through a USB cable. Thereby, when the device is plugged into a USB port, the device is powered by USB and charges the battery simultaneously. So of course if the USB is disconnected, the system is again powered by the battery. The charger itself needs protections against undervoltage, overvoltage or overheating. There are specific integrated circuits that facilitate this type of designs. These types of topics will covered more deeply in Section 3.2.

2.6 Mechanical Components

The Lipmouse does not only consist of electronic components and software, it also has mechanical components: an enclosure, a mouth piece stick, three plastic plates for placing the sensors, a plastic tube, studs, springs, screws and nuts. As mentioned, this work was a continuation of another one. The new version reused the same mechanical structure. To provide an overview of the Lipmouse project, the physical design will be shortly described in this subsection.

The enclosure, the three plastic plates, the screws and the nuts were made of acrylic glass. It is a cheap material and very easy to manipulate. The acrylic glass was cut with a laser cutter. The design was elaborated using the Inkscape and Corel Draw X6 software. Then the patterns were loaded to the laser cutter, which cut and engraved the pieces. One of the aims was to make the Lipmouse as small as possible. The final housing dimensions were 80 mm x 35 mm x 35 mm, which gave enough space to fit all the internal parts.

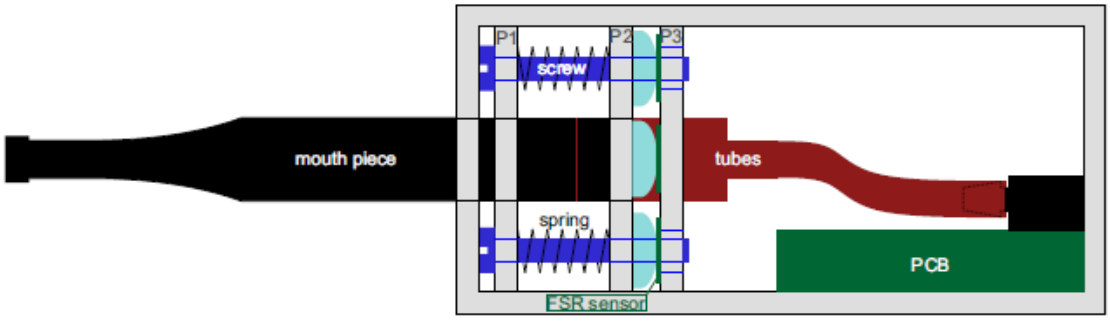


Figure 2.19: Lipmouse's scheme (Source: [15])

The Lipmouse's structure is depicted in Figure 2.19. The mouth piece breaks through the enclosure and is glued to the second plate. The movements in this part are transmitted to the small plates and the studs, applying a slight force over the FSR sensors surface. The other purpose of the mouth piece is to conduct the air flux to the sip and puff sensor. The mouth piece has a gap and at the end of it there is plastic tube connected to the sip and puff sensor. The four springs guarantee that after any movement, the mouth piece recovers its original position. The screws hold the springs and support the inner plastic plates. When they are adjusted via the nuts, a default force is applied to each sensor. Unequal forces at diametral sensors produce a slight drift. This undesired drift must be neglected, so these values should be corrected via a default offset value via the software. Behind the plate with the sensors, the PCB is placed above the battery. In the first Limpouse version, the FSR sensors where connected to the microcontroller through an air wire. However in this project, the plate where the FSR sensors are located, was modified to add the resistors of the voltage divisor and a connector for the Lipmouse PCB was installed.

3 Implementation

One main goal for this new model of the Lipmouse was wireless operation. Inside of the Wireless Personal Area Networks (WPANs) standards, (i.e, the wireless networks of a few meters of coverage) there are various technologies: Bluetooth, ZigBee, Wireless USB (WiMedia), RFID, Z-Wave, IrDA, NFC, etc. For this project, the technology chosen was Bluetooth, in its new version 4.0 Low Energy specification. However, that does not imply that it is impossible to use another technology instead. Each standard has its own advantages and disadvantages.

For the power supply system, there are many types of batteries of different chemistries and architectures. The space inside the Lipmouse limits the solution chosen, but the capacity must be big enough to operate during various hours, because the user cannot recharge the battery continuously.

When the Lipmouse is connected through a USB cable, there is no concern for power, because the USB port can usually supply up to 500 mA at 5 V (2.5 Watts). However, when it is unplugged, the device employs the battery as a source of energy. For enhancing the life cycle of the battery, adopting power saving strategies were needed. There are two types of strategies: software and hardware strategies. The microcontroller can be entered in a "sleep" mode out of a reduced power consumption state when the device is not being used. The unused microcontroller’s internal peripherals (timers, TWI and SPI interface and the USB module) can be disabled for saving more energy. In addition to this, some actions, like disabling the brown-out detector, the watchdog and the internal voltage reference or configuring the port pins to use the minimum power help to reduce the total power consumed by the system. Otherwise, taking into account this issue in the electrical design

can also save energy. The new version of the Lipmouse worked at 3.3 V versus the 5 V of the first prototype. Another essential consideration was to switch off the Bluetooth module during the "sleep" state.

Though, there is the issue of charging the battery when it is depleted. The charger must be designed and be incorporated with the rest of the electronics of the Lipmouse. Beside the charging circuit, there is the safety protections cited in Section 2.5.6 that the battery needs for a proper operation.

The other purpose of this thesis was to design and build a PCB for the electronic components of the device instead of a perfboard like the previous version of the Lipmouse, as was mentioned in Section 1.3 and 2.6.

The other part of this project was to improve the source code of the Lipmouse. On one hand, there was the necessity of developing a specific Lipmouse plugin for the ARE. A new plugin was created based on the Arduino plugin. On the other hand, the microprocessor's firmware was reedited according to the new improvements discussed in Section 3.5.2.

3.1 The Bluetooth Radio-Link

3.1.1 Introduction to the Standard

Bluetooth is a standard of WPAN developed by a consortium of enterprises: the Bluetooth Special Interest Group (SIG). Part of its specification is regulated by the IEEE in the 802.15.1 standard. The main purpose is to create a short range wireless communication technology for removing the electronic device's cables. The most important features are robustness, low complexity, small and low cost electronics, security, interoperability with other devices and low power consumption. Many parameters are optional and depend on the manufacturer's implementation, allowing a wide range of possibilities. Nowadays, more than 20000 companies are members of the SIG. (References [30] and [31])

The number of applications of Bluetooth technology is very large. It is used in many industries, like in the automotive sector, for consumer electronics, healthcare devices, mobile telephony, home automation or tools for sports and fitness monitoring.

With the version 4.0 of Bluetooth, a new concept appeared: the Bluetooth Low Energy (BLE) standard, also called Bluetooth Smart. Although Bluetooth Classic, i.e. Bluetooth Basic Rate (BR) and Enhanced Data Rate (EDR) share a lot of characteristics with the BLE specification, there are still many differences. In fact, the BLE version is not compatible with the previous ones. However, devices can implement both standards (Bluetooth Smart Ready), allowing a dual-mode communication.

The Bluetooth technology implements an entire protocol stack from the physical layer up to the application layer following the OSI model as shown in Figure 3.1.

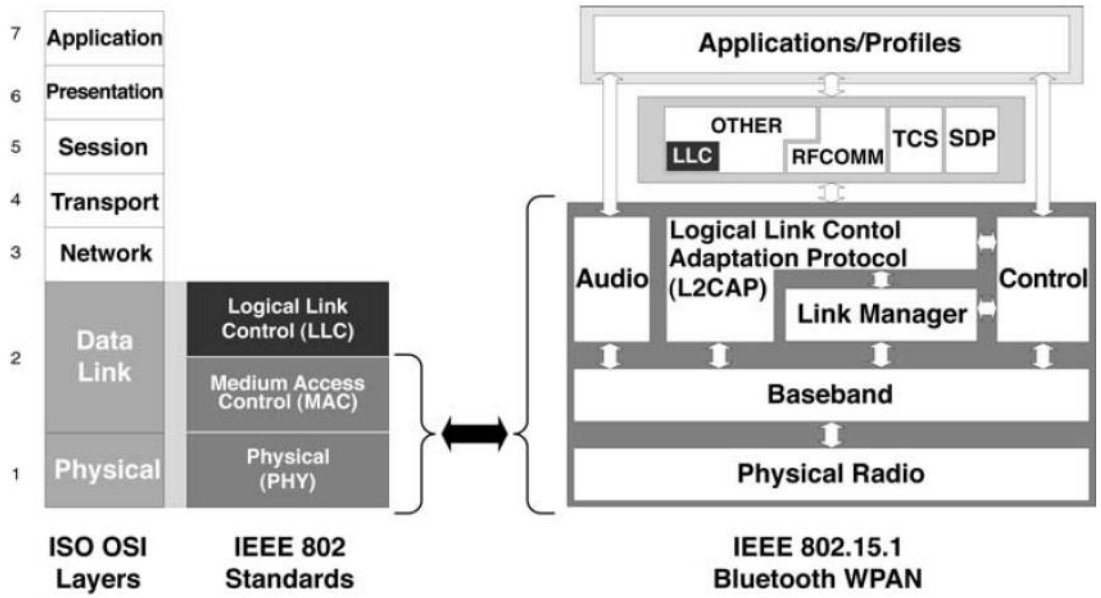


Figure 3.1: The BT protocol stack (Source: [32], page 23)

The Bluetooth system operates in the unlicensed 2.4GHz ISM band, between 2.400-2.4835GHz. In BT Classic, there are 79 physical channels with a bandwidth of 1MHz (all channels are not available in all countries, e.g. France and Spain). Nevertheless, in the BLE specification the bandwidth of the channels is 2MHz, allowing only 40 channels. The modulation in the BR case is a GFSK (Gaussian Frequency Shift Keying). In the EDR, the access code and packet header employs the same GFSK modulation (EDR is compatible with BR), whereas the rest of the packet uses a PSK (Phase Shift Keying) modulation (for enhance the data rate). The BLE also uses a GFSK modulation, but the range of the modulation index is different. In other words, BLE and BT Classic are incompatible. Their own radio specifications make them incompatible. All specifications define a Frequency Hopping Spread Spectrum (FHSS) modulation technique for avoiding interferences and fading. The transceiver changes the RF channel 1600 times per second. These specifications were fixed by the Bluetooth Core Specification, [33] and must be followed by all Bluetooth devices' manufacturers. A complete description about the GFSK and PSK modulation schemes and FHSS technique is in [34], chapters 6 and 7.

Based on the transmitted power, three power classes were defined in BT Classic. The main characteristics are summarized in Table 3.1, whereas in the BLE specification, the allowed power range is different (to reduce the power consumption), see Table 3.2.

Power Class	Maximum Output Power (Pmax)	Nominal Output Power	Minimum Output Power[1]	Power Control
1	100 mW (20 dBm)	N/A	1 mW (0 dBm)	Pmin<+4 dBm to Pmax Optional: Pmin[2] to Pmax
2	2.5 mW (4 dBm)	1 mW (0 dBm)	0.25 mW (-6 dBm)	Optional: Pmin[2] to Pmax
3	1 mW (0 dBm)	N/A	N/A	Optional: Pmin[2] to Pmax

1. Minimum output power at maximum power setting.
2. The lower power limit Pmin<-30dBm is suggested but is not mandatory, and may be chosen according to application needs.

Table 3.1: BT Classic power classes (Source: [33], page 320)

Minimum Output Power	Maximum Output Power
0.01 mW (-20 dBm)	10 mW (+10 dBm)

Table 3.2: BLE power range (Source: [33], page 2483)

The Bluetooth specification determines two possible network architectures: point-to-point or point-to-multipoint. In a point-to-point scenario, two BT units establish a physical channel, where one device is the master and the other the slave. In a point-to-multipoint scenario the physical medium access is shared by various devices, up to 7 active devices. In this topology, one is the master which coordinates the communication and the rest are slaves. These network architectures are called piconets. The communication is always master to slave, never slave to slave. Nevertheless, one slave device can be connected to two different masters, i.e. it can be a member of two piconets. This situation produces a physical connection between two piconets and is denominated scatternet. In addition to a scatternet, one device can be a master in one piconet and a slave in another piconet. There can never be two masters in the same piconet. The scenario described above is depicted in Figure 3.2.

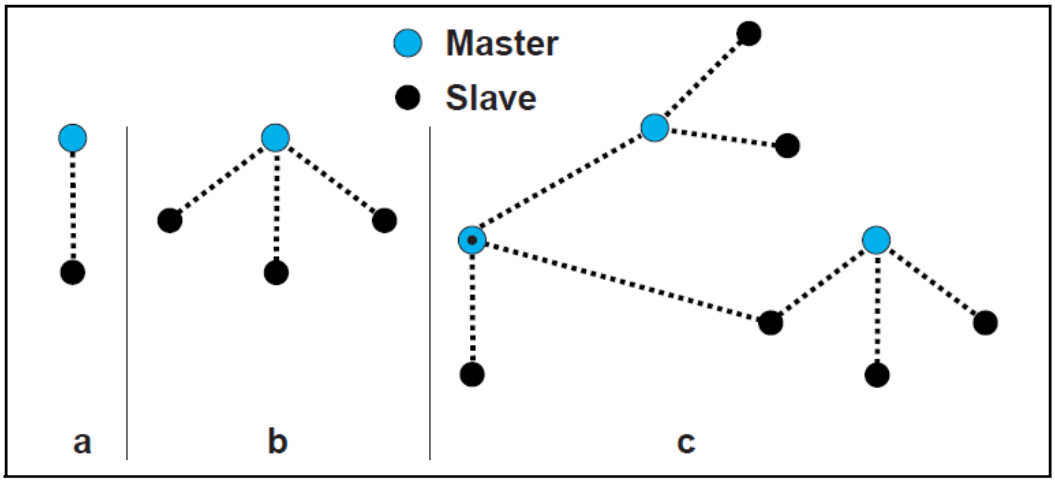


Figure 3.2: The BT network architecture (Source: [33], page 348)

The (a) situation corresponds to a point-to-point scenario, (b) represents a point-to-multipoint scheme and (c) is a plot of a scatternet. In this project, the devices employed only support a point-to-point configuration.

The BT specification defines a Time Division Duplex (TDD) scheme for the medium access. Each physical channel is divided in time slots. The duration of the time slots is 625 µs and each time slot has assigned one RF frequency (FHSS). The BT packets starts at the beginning of a time slot and can occupy 1,3 or 5 time slots. The channel is used alternatingly between the master and a slave. Moreover, BT distinguishes at the logic level two types of traffic, synchronous or asynchronous: Synchronous Connection Oriented (SCO) and Asynchronous Connection-Less (ACL), depending if it is Circuit Switching or Packet Switching traffic.

The BT state diagram specifies three principal states: standby, connection and park; and nine substates: page, page scan, inquiry, inquiry scan, synchronization train, synchronization scan, master response, slave response and inquiry response. The state diagram is depicted in Figure 3.3. The master response, slave response and inquiry response substates do not appear to simplify the picture. For a detailed analysis of how the connection process is established, consult the core specification [33] page 441 and ff. In addition, inside of the connection state, there are three modes: active mode, sniff mode and hold mode. The active mode is when the devices are actively involved in the communication. The sniff mode and hold mode are two states for reducing the activity and consequently the power consumption decreases. For example, the slave only listens the channel at certain points of time or the master refuses the asynchronous traffic during a period of time. For stopping transmitting for a long period but remaining synchronized in the channel, there is the park state. (Reference [33], page 457 and ff.)

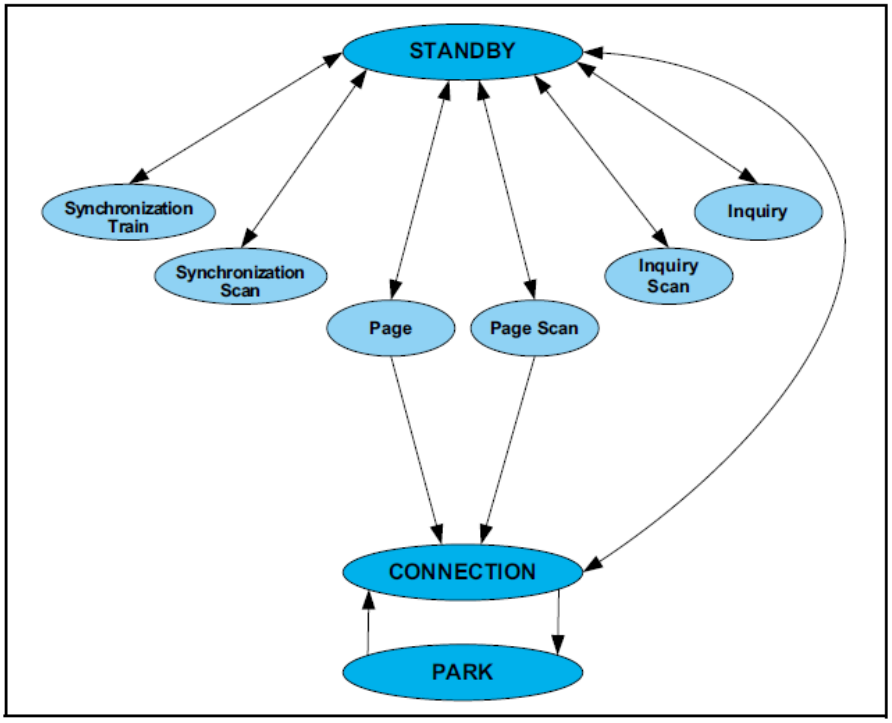


Figure 3.3: The BT state diagram (Source: [33], page 441)

Finally, at the top of the protocol stack, there are the BT Profiles. A BT profile is a set of particular definitions and specifications of possible BT application functions and features. Examples of profiles are:

- Advanced Audio Distribution Profile (A2DP)
- Basic Imaging Profile (BIP)
- File Transfer Profile (FTP)
- Headset Profile (HSP)
- Human Interface Device Profile (HID)
- Serial Port Profile (SPP)
- Video Distribution Profile (VDP)

The data sheets of the BT devices employed do not specify which profile implements. However, Serial Port Profile is usually the profile implemented by Bluetooth applications which emulate a serial communication.

3.1.2 HM10 BLE module and BLE-receiver USB dongle configuration

The HM10- BLE module was placed in the designed Lipmouse's PCB and wired with the Atmel microcontroller via the UART interface. Thereby, the microcontroller had to be programmed in order to send the configuration commands. These commands are vendor-specific and depend on the device's firmware developed by the vendor, JNHuaMao Technology Company.

Moreover, the BLE-receiver USB dongle, [35], has to be plugged to the embedded platform or the PC, where the ARE is running, in order to use the Lipmouse. When the ARE detects a newly connected USB device, it launches a scan process to detect if it is a new CIM device. The scan process mainly consists of sending a CIM packet to every COM port, asking for the device identification. If the receiver is a CIM device, it will answer to the identification packet and the ARE will register it. This fact is important, because if the Bluetooth connection is not established before the scan process starts, the CIM packet will not arrive to the device and the ARE cannot register it. If the Lipmouse is being used by the running model, the ARE stops automatically. Therefore, the Lipmouse's BLE module and the BLE-receiver USB dongle must be paired before clicking the ARE's start button.

The configuration of the HM10 BLE module and the BLE-receiver USB dongle is done by AT commands. The commands must be sent in upper-case string format without spaces and without any other symbol. In the case of the BLE-module, the serial communication is by default set to 9600bps, 8 data bits, 1 stop bit and no parity bit. The BLE-receiver USB dongle adjusts the data parameters automatically, since it opens a virtual serial COM port through the USB connection. Some AT commands are the same for the module and the dongle. However, there are only a few commands available for the module or the dongle. The most important commands are shown in Table 3.3. For more information, please consult the datasheets of the devices, [25] and [35].

One problem of the AT commands, which was not noticed in the datasheets, refers to the time needed to wait between sending two AT commands. It was proven empirically, that if two commands are sent too fast, the device does not interpret them and no response is received. The conclusion reached is, that e.g. if *AT+ROLE?* and immediately *AT+IMME?* are sent, the device receives *AT+ROLE?AT+IMME?*, which is not a valid command. Commands cannot be delimited by including special characters like for example \r (carriage return) or \n (new line). The solution for this problem was solved by waiting a few milliseconds, before sending a second command. For almost all commands waiting 50ms would be enough for a correct sending process. However, for commands which result in a device reset, as *AT+RESET,* more waiting time is needed.

There are two ways for the establishment of the BT link: automatic (default) or manual. In the automatic mode a master device detects the availability of slave devices (devices in slave mode which are not paired with another device); if found, the pairing is established immediately. When the connection is lost, the master will start again looking for the same slave. If the same slave is available, the connection will be successful (this statement was not defined in the datasheet but was deduced empirically by observing how the JNHuaMao devices work).

Send	Response	Parameter (Para)	Device	Brief description
AT	OK / OK+LOST	None	Both	If the module is not paired, it will be received OK. It is paired the connection will be lost and it will be receive OK+LOST only if the notifications are set. (see: AT+NOTI)
AT+ADDR?	OK+ADDR:MAC	None	Both	Ask the MAC address of the device
AT+BAUD? AT+BAUD[Para]	OK+Get:[Para] OK+Set:[Para]	0——9600 1——19200 2——38400 3——57600 4——115200 5——4800 6——2400 7——1200 8——230400	Module	Query/Set the baud rate of the module. By default, the baud rate is 9600bps.
AT+CON[MAC]	OK+CONN[Para]	A: Connecting E: Connect error F: Connect Fail	Module	Try to connect to the device with that MAC. Only master role devices. If the target is not a slave or is already connected, the attempt will fail.
AT+CONN[Para]	OK+CONN[Para]	0-1: Target device. In the response also: E: Link error F: Link failed	Dongle	Use it after *AT+DISC?* Try to connected in the "[*Para*]" position listed by the *AT+DISC?* command.
AT+DISC?	OK+DISC[Para]	S: Start discovery Address string E: End discovery	Dongle	Scan all available slave devices. The information about them is listed in the "Address string".
AT+IMME? AT+IMME[Para]	OK+Get:[Para] OK+Set:[Para]	0: Connect automatically 1: Connect after AT command	Both	Connect automatically after power on the device or wait until the corresponding command is received (*AT+START*, *AT+CON*/*AT+CONN* or *AT+CONNL*).
AT+RENEW	OK+RENEW	None	Both	Restore all default settings
AT+RESET	OK+RESET	None	Both	Reset the device
AT+ROLE? AT+ROLE[Para]	OK+Get:[Para] OK+Set:[Para]	0: Slave 1: Master	Both	Query/Set the role of the device. Slave by default.
AT+SLEEP	OK+SLEEP	None	Both	Send to sleep the device. Only is supported by slave devices.

Table 3.3: Most important AT commands extracted from the datasheets. To see the whole list, consult [25] and [35]

In the manual case, the connection process is done by sending AT commands. Here, there are also two possibilities to establish the connection manually. The first way is sending the command, *AT+CONNL* (this is, connect to the last device paired successfully). This command is available for both: module and dongle.

If it is the first time that the connection is established (no previous device was memorized), the second way of manual connection has to be used. In this case the command *AT+CON[MAC ADDRESS]* must be sent for the module. For example, if the MAC address of the device was 00:11:22:33:44:55, the command would be *AT+CON001122334455*.

For the BLE-receiver USB dongle, the utilized command for manual connection is *AT+DISC*. After receiving this command, the dongle will first search for all BLE slave devices inside its cover range. Secondly, the dongle sends a list with the available devices via the USB interface. Then a second AT command has to be sent for choosing a device of the list, *AT+CON[POSITION ON THE LIST - 1]*. If the connection to the first device is required, the command must be *AT+CON0*. If it is for the second device, the command would be *AT+CON1*, and so on. The command *AT+DISC* only displays the 6 first devices found.

Both solutions have disadvantages. In the automatic configuration, if more than one BLE slave device is available, the selection of the slave by the master is unpredictable. It could be a problem if there are other BLE applications (which are not AsTeRICS CIM devices) inside the cover range. The advantage is that, if the first pairing is successful, the device will always try to connect to this first device, avoiding the connection to other available slave devices.

The main drawback of the manual case is that during the connection process, if the module is the master, the MAC address of the dongle needs to be known. In order to find out the MAC address of the dongle, the user should open a virtual serial COM port with the dongle using a program which allows the serial communication with the COM ports, e.g HTerm [36]. After the COM port is opened, the MAC address query command has to be sent. Finally, incorporation of the address to the microcontroller's firmware code is mandatory.
In case of configuring the dongle as the master, first of all it is necessary to know the module MAC address or to change the device's name for a recognizable one. Secondly, the *AT+DISC* hast to be sent to look for the module. Finally the command for the connection is required. An additional problem is how to send the AT commands. In case of the module as master, it is easy via the microcontroller. However, in the case of the dongle as master, the use of a program as HTerm or a suitable configuration application (*.exe* file) is required. If instead of AT+DISC, the *AT+CONNL* command is used, the previous configuration process is required for the first pairing. In addition, if one of the devices breaks down and must be replaced by another, the procedure must be repeated. In

conclusion, in order to facilitate the use of the Lipmouse configuration procedure, the automatic option was selected.

Another critical decision is the determination of which device is the master and which is the slave. If the module is configured as master and the dongle as the slave, the dongle can be used with the default settings (in other words, it does not need any configuration). However, the appropriate baud rate and the master role have to be configured in the module. This can be done at the microcontroller's firmware. In addition to this, once the first connection is established, the module can only be paired with the same dongle, with which it was paired the first time. Therefore, if the dongle is substituted, the user should restore the module default factory settings for erasing the address of the previous dongle. This action involves modifying the firmware and flashing the Lipmouse's firmware again.

If the dongle is configured as master and the module as the slave, the baud rate and the role configuration also has to be done by the microcontroller's firmware. Moreover, the dongle has to be configured via HTerm or another similar program. As previous case, once the dongle has established the first connection, it will always try to connect to the same device. The command *AT+RENEW,* is used for restoring the default factory settings and forgetting the address of the device. Thereby, the dongle could establish another connection to a different device.

A reasonable requirement is that the firmware of the microcontroller should not need any changes even if a connection with a different USB dongle is required. Therefore, the dongle should be configured as a master and the module as a slave instead of the other way around. The resulting configuration process of the dongle is described in Appendix A.

The next configuration issue concerns the module's UART baud rate: The ARE opens the virtual serial ports at 115200 bps. Thus, the logic procedure would be to set the interface UART's baud rate at 115200 bps. Nevertheless, it cannot be done, because the microcontroller was supplied with 3.3V. At this operating voltage, the ALU's frequency clock should be prescaled from 16MHz (external's crystal frequency clock) to 8MHz in order to avoid over-clocking according to the AT90USB1286 specifications. On the other hand, the microcontroller's UART could not work properly at 8MHz and a speed of115200 bps. As it can be seen in Table 3.4, the error is too high at 115200 at this baud rate for both Normal and Double Speed modes. The speed of 76800 bps in Double Speed mode has a slight error (0.2 %). However, the BLE module cannot support this baud rate (see Table 3.3). Therefore, the best option was choosing 57600 bps in the Double Speed mode. As shown in Table 3.5, in the Double Speed mode, the recommended maximum error for 8 data bits and non-parity bit is ±1.5 %. At this speed the error is 2.1 % which is better than the -3.5 % error at 115200 bps and it is still inside the maximum range of total errors for a working data communication. Testing if the microcontroller's UART could support 115200 bps was performed elucidating that it was impossible to establish a communication at this

baud rate. Selecting a speed of 38400 bps has an error inside of the recommended range, but it would lead to a low baud rate which would limit the number of Lipmouse coordinated updates per second.

Baud Rate (bps)	f_{osc} = 8.0000 MHz				f_{osc} = 11.0592 MHz				f_{osc} = 14.7456 MHz			
	U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1		U2Xn = 0		U2Xn = 1	
	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error	UBRR	Error
2400	207	0.2%	416	-0.1%	287	0.0%	575	0.0%	383	0.0%	767	0.0%
4800	103	0.2%	207	0.2%	143	0.0%	287	0.0%	191	0.0%	383	0.0%
9600	51	0.2%	103	0.2%	71	0.0%	143	0.0%	95	0.0%	191	0.0%
14.4k	34	-0.8%	68	0.6%	47	0.0%	95	0.0%	63	0.0%	127	0.0%
19.2k	25	0.2%	51	0.2%	35	0.0%	71	0.0%	47	0.0%	95	0.0%
28.8k	16	2.1%	34	-0.8%	23	0.0%	47	0.0%	31	0.0%	63	0.0%
38.4k	12	0.2%	25	0.2%	17	0.0%	35	0.0%	23	0.0%	47	0.0%
57.6k	8	-3.5%	16	2.1%	11	0.0%	23	0.0%	15	0.0%	31	0.0%
76.8k	6	-7.0%	12	0.2%	8	0.0%	17	0.0%	11	0.0%	23	0.0%
115.2k	3	8.5%	8	-3.5%	5	0.0%	11	0.0%	7	0.0%	15	0.0%
230.4k	1	8.5%	3	8.5%	2	0.0%	5	0.0%	3	0.0%	7	0.0%
250k	1	0.0%	3	0.0%	2	-7.8%	5	-7.8%	3	-7.8%	6	5.3%
0.5M	0	0.0%	1	0.0%	–	–	2	-7.8%	1	-7.8%	3	-7.8%
1M	–	–	0	0.0%	–	–	–	–	0	-7.8%	1	-7.8%
Max. (1)	0.5 Mbps		1 Mbps		691.2 kbps		1.3824 Mbps		921.6 kbps		1.8432 Mbps	

1. UBRR = 0, Error = 0.0%

Table 3.4: Baud rate errors at 8MHz. Normal Speed mode (U2Xn = 0) and Double Speed mode (U2Xn = 1). (Source: [37] , page 205)

Apparently there was a speed mismatch problem. The maximum baud rate for the microcontroller was 57600 bps and the ARE worked at 115200. The ARE baud rate could not be modified, because this would affect the other CIM devices developed until that moment. All CIMs work at 115200 bps and when the ARE opens a serial communication port, it does not know which type of CIM is plugged in. Hence all communications have to be compulsory work at 115200 bps from the ARE side. One idea was to send a special sequence at 115200 bps, which could be recognised by a device working at 57600 bps. If the device detected this sequence, it would send a message to the ARE. Then the ARE would have realized that the CIM device was working at 57600 bps. Therefore, the ARE would change the speed of this port to 57600 bps allowing the communication with a slower CIM without affecting other possible CIMs attached.

However, this speed mismatch was not actually a problem. When the data arrives at the serial interface of the BT device, it is encapsulated in a BT frame. The frame will travel at various Mbps (in the air). As a consequence, the data is buffered between the layers of the protocol stack. Thereby, a BT channel in the application layer can work at one speed, but in the lower layers the data travels at more speed. Besides, the channel is not always transmitting. Due to these facts, one side of the communication can work faster than the

other side, if the data rate (the total amount of data per time, no the serial velocity) can be absorbed by the slower side of the communication. Otherwise, it will overflow and the communication will fail. It turned out that the baud rate specification is only necessary for the UART side (the microcontroller interface connecting to the BLE-module on the PCB) – and that the baud rate specification for the Virtual COM Port (for the connected USB dongle) is completely obsolete. Therefore, the BT module was configured to a baud rate of 57600 bps and the ARE still opens the VCP with 115200 bps. A test of the communication revealed that there was not any problem working at different speeds.

Finally, the definitive solution was: if it is the first connection, the role of the dongle must be set as a master. After changing the role, the device will try to connect to the module automatically when the two devices will be switched on. The procedure can be seen in Appendix A. The configuration settings are not cleared when the device is switched off. Thus, the configuration process has not to be repeated for future uses.

For the module, the configuration commands were programmed in the microcontrollers' firmware and they will be sent to the module via the UART interface. Therefore, the user does not have to intervene in the module's configuration. The configuration commands are sent every time that the firmware is reset / the Lipmouse is powered on. If the module was already configured, the configuration commands could be bypassed. However, it was easier to send the commands every time than to detect if the configuration was already done. In addition, in the unusual case that the Lipmouse (or BLE-module) would be replaced, it will also be configured without performing any special action. The resulting commands sequence is:

- Start the UART at 9600 bps (module's factory speed)
- Send the AT command for changing the baud rate to 57600 bps and wait 50 ms.
- Send the AT command for changing the baud rate to 57600 bps and wait 50 ms again. It was noticed in several cases that the baud rate change was not done properly if the command is only sent one time.
- Send the AT command for resetting the module. When the baud rate is changed, this it is not effective until the device is not reset.
- Wait 1000ms for the module restarting.
- Restart the UART at 57600 bps.

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	93.20	106.67	+6.67/-6.8	± 3.0
6	94.12	105.79	+5.79/-5.88	± 2.5
7	94.81	105.11	+5.11/-5.19	± 2.0
8	95.36	104.58	+4.58/-4.54	± 2.0
9	95.81	104.14	+4.14/-4.19	± 1.5
10	96.17	103.78	+3.78/-3.83	± 1.5

(a) Baud rate error depending of the number of data bits in Normal Speed mode (U2Xn = 0) (Source: [37], page 197)

D # (Data+Parity Bit)	R_{slow} (%)	R_{fast} (%)	Max Total Error (%)	Recommended Max Receiver Error (%)
5	94.12	105.66	+5.66/-5.88	± 2.5
6	94.92	104.92	+4.92/-5.08	± 2.0
7	95.52	104,35	+4.35/-4.48	± 1.5
8	96.00	103.90	+3.90/-4.00	± 1.5
9	96.39	103.53	+3.53/-3.61	± 1.5
10	96.70	103.23	+3.23/-3.30	± 1.0

(b) Baud rate error depending of the number of data bits in Double Speed mode (U2Xn = 1) (Source: [37], page 197)

Table 3.5: Baud rate errors

3.2 The Battery

3.2.1 Battery Selection

Developing a portable solution involves two topics: a wireless communication system and a battery power supply. To understand why some batteries are better than others, it is necessary to define the principal parameters that characterize them.

The first two questions to solve concern the suitability: a primary or a secondary battery and which type of battery is better. In other words, what is to prefer: a non-rechargeable or rechargeable battery and which chemistry is the best concerning specific capacity, specific voltage, safety, life span, maintenance and self-discharge. The question of choosing a non-rechargeable or a rechargeable variant is indirectly related to chemistry. Although primary batteries have better performance and reliability in some aspects, the price (considering a longer timespan) and the convenience of a rechargeable solution led to choosing a secondary battery.

A rechargeable battery can be acquired for less than 15 € and an alkaline AA or AAA cell can cost around 0.35 € (if they are bought in large amounts). This means, that for more than 43 uses, it is cheaper than a non-rechargeable solution. If the Lipmouse was used for a year, the increased cost would be amortized. Moreover, many users see an inconvenience in changing the cells or having new cells available when needed. It is easier for them to connect the device to the computer via a USB cable or to plug it in a local jack for recharging the battery, rather than to change the alkaline cells. Nowadays, many consumer electronics are recharged through a USB cable plugged in a computer. Hence, it was reasonable to develop a solution of this type since this project involves using a PC or another embedded platform with USB connectors. In addition, there was also the environmental issue with the chemical recycling problems of the batteries.

The battery characteristics are derived from the inner electrochemical process. Thereby batteries with the same chemistry, but with different internal designs have different performance parameters. The principal chemistries are:

- Lead-acid
- Nickel-cadmium
- Nickel-iron
- Nickel-zinc
- Nickel-hydrogen
- Nickel-metal hydride
- Zinc-silver oxide
- Zinc-bromine
- Sodium-sulphur
- Sodium-metal-chloride
- Cadmium-silver oxide
- Lithium-ion-cobalt
- Lithium-ion-manganese
- Lithium-ion-phosphate
- Lithium-ion polymer
- Redox batteries

Moreover, there are solutions based on supercapacitors and ultracapacitors used as power supply sources. Depending on the requirements of the application, some are more suitable than others. The most important parameters are:

- Electrical characteristics. The nominal voltage value of the battery determines the number of cells needed. The microcontroller chip needs a voltage supply of 2.7-5.5V. Thus, it would be necessary to use more than one battery cell for some types of chemistries. In addition, the minimum and maximum voltage range tolerable must be taken into account. A mismatch in the voltage between the battery and the rest of the circuit could damage some electronic components of the system, including the battery. Analogous considerations must be done for the electrical

current. Therefore, the charging and discharging I-V curve of the battery would affect the electronic design of the PCB.

- Capacity. The capacity of a battery is usually expressed in ampere-hour (or milliampere-hour). This capacity is given for a specific discharge rate (the C-rate), temperature, and cut-off voltage. The duration and the amount of current delivered to the load can be calculated using this value. For example, a 1000 mAh battery rated at a discharge current of 200 mA entails 5 h of autonomy. This means that if the battery was used at a discharge rate of 100 mA, the battery would supply the device during more or less than 10 hours, because, as cited above, the capacity also depends on the discharge rate. In addition, the battery has its limits and perhaps it cannot provide a high discharge rate, e.g. 2000 mA during 30 minutes. Another way of expressing the capacity is determined by the energy stored in the battery (in watt-hours). This quantity is defined in terms of mass and volume: specific energy or gravimetric energy density and energy density or volumetric energy density. Therefore, a battery with low specific energy and low energy density will be large and have high weight. In other words, the battery selected should have the highest possible specific energy and energy density.

- Duty cycle. The current supply can be continuous or intermittent. Some batteries are optimized for delivering a considerable amount of power during a short period of time, for example the flash of a camera. For this project, the application demands a continuous flux of energy.

- The temperature. The specific energy of a battery depends on the temperature. Batteries do not work properly below or above a certain temperature. For extreme conditions special batteries have to be used. This application is thought to be used by a person inside a building. Thus, the battery selected must work correctly between 15 to 30 ºC.

- Lifetime. Some batteries can be recharged more often than others. The more cycles, the longer the life of the battery. However, some batteries are more resistant to overcharging and deep discharging than others. These situations can decrease the battery's performance dramatically. Also, there is always a risk of catastrophic failure, i.e. suddenly, the battery breaks down due to a failure in the cell. Therefore, all batteries have a medium lifetime.

- Physical dimensions. The weight and the size of the battery are among the most critical aspects in this project. At the moment of the realization of this work, the enclosure design had the dimensions of 35 x 35 x 80 mm. The sensors, the PCB, and a part of the mouth stick had to fit inside the enclosure together with the battery. There are many types of packages available in the market, as some battery chemistries allow building different form factors.

- Self-discharge. The batteries have internal losses and some types are sold with a protection circuit that consumes a small amount of energy. Thus, these batteries are discharging if they are not used over a long period of time.

- Environmental conditions. Besides the temperature, other factors, such as the air pressure or the humidity affect the battery's performance. Moreover, if the application involves vibration, spinning, high accelerations or decelerations, an effect in the battery's life could be induced. However, this is not an issue for this project, since the Lipmouse will be stored under normal conditions of humidity and pressure and the device will not be exposed to excessive movements.

- Safety and reliability. As noted above, some batteries can be damaged out of inappropriate use. The batteries can break down and liberate corrosive and hazardous gases or liquids. In addition to this, the battery can overheat due to a malfunction and may damage other parts of the device. Also, some batteries are sensitive to deep discharge or under-/overvoltages. Hence, batteries need some electronic protections.

- Maintenance. Some components of the batteries need to be replaced after a prolonged use or cleaning of the electronic contacts may be required. Several battery chemistries imply the voltage depression effect (also called memory effect), which forces the user to perform a full charge and discharge cycle to recover the capacity of the battery.

- Cost. There is always a trade-off between performance and price. The best battery is usually the one that has good characteristics (not necessarily the best ones) at a reasonable cost, as one aim of the project is to develop a cost-effective solution.

Table 3.6 summarizes the most important parameters for typical battery chemistries.

The Li-ion batteries have a high energy density, high specific energy, more than 1000 charge/discharge cycles, low charge time and the cell voltage is higher than in other types. These batteries are good for portable and small devices like mobile phones, tablets, digital cameras, laptops, mp3 players, and others. Therefore, they became very popular in recent years. However, they have an important drawback: they need a protection circuit due to safety reasons. This battery technology does not tolerate overcharging or low voltages (undervoltage problems). All types of Li-ion chemistries are similar in performance and only have slight differences among them. However, in particular the Li-ion polymer batteries have greater energy density and they are more stable than the rest of Li-ion battery types. Moreover, one of the best advantages is that they are more flexible in cell sizes. Consequently, a wide variety of these batteries in different sizes is available on the market. (Reference [38] and [39])

Hence, the battery chosen was a Lithium-ion polymer battery. The battery was bought from SparkFun Electronics, [40] and it has a capacity of 1000mAh. It offers a low self-discharge rate and the nominal voltage is 3.7 V, so that only one cell is necessary. In addition, it was not too expensive ($8.95, without taking into account the possible shipping expenses), and its dimensions fit the requirements: 50.8 x 33.5 x 5.9 mm. As a result, it fitted perfectly in

Common name	Lead-acid				Nickel-cadmium				Nickel-iron (conventional)	Nickel-zinc	Zinc / silver oxide (silver-zinc)	Cadmium / silver oxide (silver-cadmium)	Nickel-hydrogen	Nickel-metal hydride	Rechargeable "primary"[c] types, Zn/MnO_2	Lithium ion systems[a]
	SLI	Traction	Stationary	Portable	Vented pocket plate	Vented sintered plate	Sealed	FNC								
Cell voltage (typical), V:																
Nominal	2.0	2.0	2.0	2.0	1.2	1.2	1.2	1.2	1.2	1.65	1.5	1.1	1.4	1.2	1.5	4.0
Open circuit	2.1	2.1	2.1	2.1	1.29	1.29	1.29	1.35	1.37	1.73	1.86	1.41	1.32	1.4	1.5	4.1
Operating	2.0–1.8	2.0–1.8	2.0–1.8	2.0–1.8	1.25–1.00	1.25–1.00	1.25–1.00	1.25–0.85	1.25–1.05	1.6–1.4	1.7–1.3	1.4–1.0	1.3–1.15	1.25–1.10	1.3–1.0	4.0–3.0
End	1.75 (lower operating and end voltage during cranking operation)	1.75	1.75 (except when on float service)	1.75 (where cycled)	1.0	1.0	1.0	1.00–0.65	1.0	1.2	1.0	0.7	1.0	1.0	1.0	3.0
Operating temperatures, °C	−40 to 55	−20 to 40	−10 to 40[e]	−40 to 60	−20 to 45	−40 to 50	−40 to 45	−50 to 60	−10 to 45	−10 to 50	−20 to 60	−25 to 70	0 to 50	−20 to 50	−20 to 40	−20 to 50
Specific energy and energy density (at 20°C)																
Wh/kg	35	25	10–20	30	20	30–37	35	10–40	30	50–60	105[f]	70	64 (CPV)	75	85	150
Wh/L	70	80	50–70	90	40	58–96	100	15–80	55	80–120	180	120	105 (CPV)	240	250	400
Discharge profile (relative)	Flat	Flat	Flat	Flat	Flat	Very flat	Very flat	Flat	Moderately flat	Flat	Double plateau	Double plateau	Moderately flat	Flat	Sloping	Sloping
Power density	High	Moderately high	Moderately high	High	High	High	Moderate to high	Very high	Moderate to low	High	Very high (for high rate-design)	Moderate to high	Moderate	Moderate to high	Moderate	Moderate: high in prismatic designs
Self-discharge rate (at 20°C), % loss per month[b]	20–30 (Sb-Pb) 2–3 (maintenance-free)	4–6	—	4–8	5	10	15–20	10–15	20–40	<20	5	5	Very high except at low temp.	15–25		2
Calendar life, years	3–6	6	18–25	2–8	8–25	3–10	2–5	5–20	8–25	—	2	3 (vented) 4 (sealed)	—	2–5		
Cycle life, cycles[d]	200–700	1500	—	250–500	500–2000	500–2000	300–700	500–10,000	2000–4000	500	50–100	300–800	1500–6000 40,000 at 40%	300–600	15–25	1000+

Table 3.6: Battery parameters of the most typical chemistries (Source: modified from [38], chapter 22)

the enclosure. In addition, the connection with the rest of the circuit was made by a standard 2-pin JST connector, which was simpler than, for instance, the use of metallic contacts. In conclusion, this battery suited for the project.

3.2.2 Circuit Protection Design

Not only the selection of most the suitable battery was important, but also the design of the charger for it. The Li-ion batteries need a protection circuit, which is directly related to the charging system. The operating principle of this type of batteries and their requirements will be discussed below, in order to understand how the charger and the protection circuit should be designed.

Most lithium batteries follow a constant current-constant voltage charging algorithm, depicted in Figure 3.4.

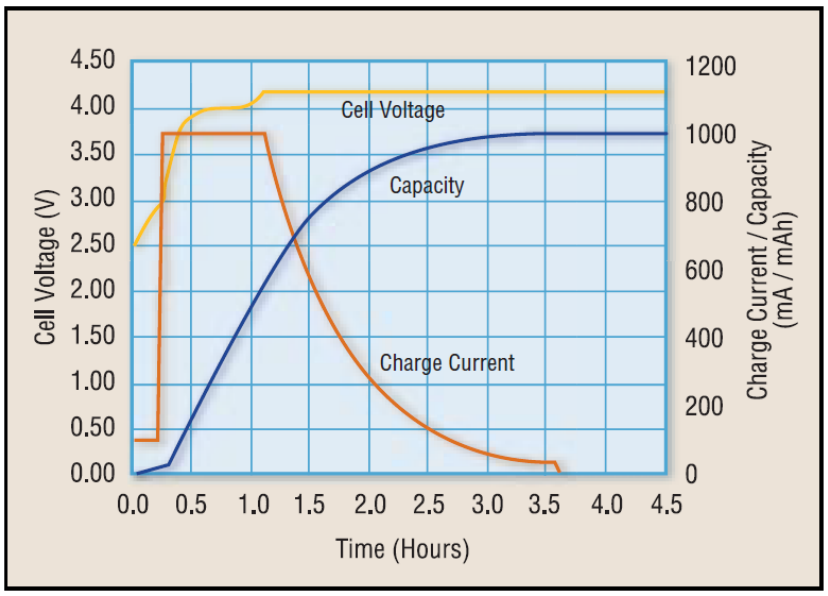


Figure 3.4: The Li-ion charging cycle (Source: [41])

The charging cycle can be subdivided in three stages. The first stage only occurs if the cell is completely depleted and the voltage of the battery cell is below 3 V. During this stage the cell must be supplied with a small constant current, until the threshold voltage is reached. In the second stage the current is raised up to the desired charging current value. This current must be maintained constantly during the whole stage, while the voltage of the battery is increasing. When the voltage is equal to the float voltage level, 4.2 V, the constant current phase finishes. In the third stage the current starts to diminish and the voltage remains constant. The charge cycle ends when the current is equal to a minimum level. After several recharging cycles, the battery may not achieve this minimal value. Thus, a timeout or other external mechanism is necessary to terminate the charge.

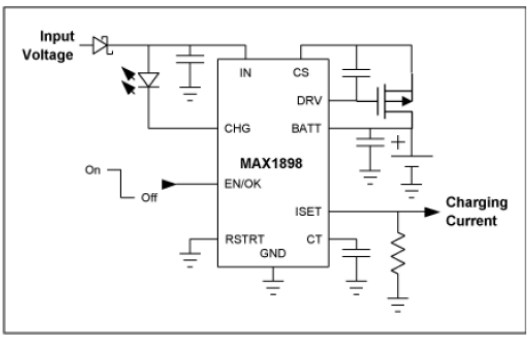
Notice that in the constant-current phase (second stage), the higher the current is, the sooner the float voltage will be reached. However, this does not shorten the charge cycle time significantly, since in the next phase more time will be required to reduce the current to the minimum level due to the charging current is higher. (References [41] and [42])

The battery must be protected in case of: under/overvoltage situations, battery overheating or when the minimum termination current is not reached. Therefore, the major electronics manufacturers have developed a wide range of integrated circuits (ICs), which control the charging process. These solutions can be classified in three groups, depending on the charging method performed: linear, switch-mode, and pulse charger. (Reference [44])

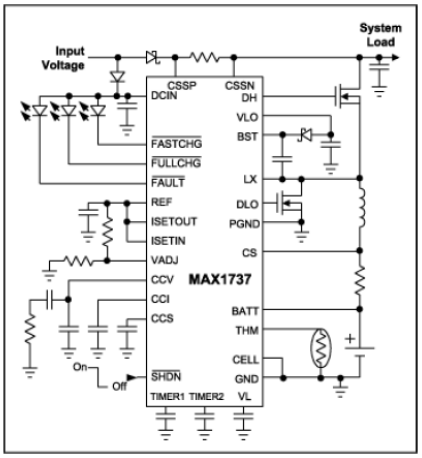
A linear charger is easy to implement, more compact and cheaper than the other two types of chargers. It uses a pass transistor to supply the voltage from the source to the battery. This configuration has a low efficiency, since the transistor dissipates a lot of heat. One advantage of this type of charger is that it needs only a few external components. In addition, it is immune to electrical noise. Figure 3.5 (a) shows an example of a linear charger, the MAX1898 by Maxim Integrated.

The switch-mode option is more efficient, seen from the energetic viewpoint and it can be used with a wide range of voltage inputs. However, it needs more external components, for example a LC filter, increasing the size of the PCB significantly. Another disadvantage is that the switching action produces electrical noise and the inductor generates spurious radiations, both potentially leading to interferences. An example of this type of solution is depicted in **¡Error! No se encuentra el origen de la referencia.** (b), the MAX1737 by Maxim Integrated.

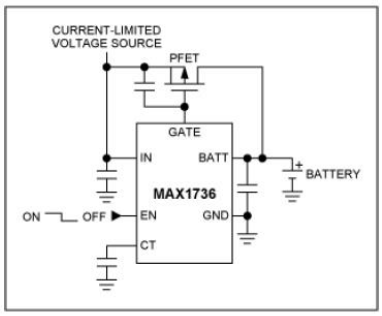
The third type, the pulse charger, is a combination of the two previous charging technologies. It works more efficiently than a linear charger, but less than the switch-mode option. The pulse charger also needs a pass transistor that connects the source with the battery. When the cell is depleted during the trickle charge, the transistor supplies the current like in the linear mode. However, in the constant voltage phase, the transistor acts as a switch (it operates by allowing or blocking the conduction of the current). The power dissipation is reduced, but it needs more time to charge. This configuration does not require a LC filter, so the space on the board is even more reduced than in the switch-mode configuration. The main problem with using this class of chargers is that the input source has to be current-limited and accurate. Consequently, these solutions are usually more expensive than the other charger types. The Figure 3.5 (c) shows an example of a pulse charger, MAX1736, also by Maxim Integrated.



(a) Linear charger (Source: [44])



(b) Switch-mode charger (Source: [44])



(c) Pulse charger (Source: [44])

Figure 3.5: The different charger types

The resulting size of the PCB was crucial in this project. The solution had to be as simple as possible. The linear chargers use less components and they do not need a LC filter. Also the charging time is much shorter than in the pulse configuration. Thereby, the best option was to search for a linear charger IC. In case the linear charger solution did not fulfil all requirements or the heat dissipation was a problem, a pulse or switch-mode chip would be selected.

Four of the major electronics manufacturers were consulted for the charger IC: Analog Devices, Texas Instruments, Maxim Integrated and Microchip Technology. The possible solutions and their main characteristics are summarized in Table 3.7.

The first column in Table 3.7 refers to the name of the chip and the second to the final charge voltage. Depending on the cell design, this voltage is different in each case. Most of the chargers work at 4.1 V or 4.2 V, but of course there are other types, too. As the charge

	IC name	Charge voltage (4.2V)	Control topology	Control interface	Charge source	Current charge (constant current phase) Ichg	Vin (operating input voltage)	Vin max > 5V	Battery and system load isolated/sharing	Vout (output voltage to the system)	Integrated pass transistor	Safety timer charging termination	Temp. Monitoring termination	Reverse protection	Status pins	Undervoltage lockout (UVLO)	Temp. range	Packing
Maxim integrated	MAX8934A/B/C/D/E	4.2V	Linear	Standalone	AC/USB	30mA-1500mA	4.1V-6.6V	9V (USB)	Isolated	5.3 (A) and 4.35 (B/C/D/E)	Yes	Yes (no configurable)	Yes	No information	Two (charging status and charging done)	4.00V	-40 to 85ºC	4mm x 4mm TQFN (28 pins)
	MAX8895V/W/X/Y	4.2V	Linear	Standalone	AC/USB	max. 1850mA	?-6.6V	14V	Isolated	-	-	Yes	Yes	-	-	-	-40 to 85ºC	WLP (25 pins)
	MAX8922L *with GSM Test mode	4.2V	Linear	Standalone	AC/USB	400mA (default), 100mA and 100mA (by low pulses)	4.45V-7V	30V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging) and LDO 4.94V, 100mA	Yes	No	No but has thermal protection	Yes	Two (charging status and power source)	3.90V	-40 to 85ºC	3mm x 2mm TDFN-Exposed Pad (10 pins)
	MAX8600A	4.2V	Linear	Standalone	AC/USB	300mA-1000mA	4.1V-6.0V	14V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	Yes	No information	Two (charging status and power source)	4.00V	-40 to 85ºC	3mm x 3mm TDFN-Exposed Pad (14 pins)
	MAX8845W/X/Y/Z	4.2V	Linear	Standalone	AC/USB	100mA-500mA	4.2V-7.0V	28V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging) and LDO 4.7-4.85V	Yes	No	No but has thermal protection	Yes	Two (charging status and power source)	No	-40 to 85ºC	3mm x 3mm QFN-Exposed Pad (12 pins)
	MAX8844Y/Z	4.2V	Linear	Standalone	AC/USB	88mA-440mA	4.2V-7.0V	28V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging) and LDO 4.7-4.85V	Yes	No	No but has thermal protection	Yes	Two (charging status and power source)	No	-40 to 85ºC	3mm x 3mm TDFN-Exposed Pad (14 pins)
	MAX8856	4.2V	Linear	Standalone	AC/USB	with 500mA input-> 250 sys and 225 batt	4.0V-5.8V	14V	Isolated	3.5V (with Vbat = 3.3V)	Yes	No	Yes	Yes	Two (charging status and power source)	4.00V	-40 to 85ºC	3mm x 3mm TDFN-Exposed Pad (14 pins)
	MAX8814	4.2V	Linear	Standalone	AC/USB	107mA-570mA	4.25V-6.80V	28V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	No	No	Yes	One (power source)	No	-40 to 85ºC	2mm x 2mm TDFN-Exposed Pad (8 pins)
	MAX8804 V/W/Y/Z	4.2V	Linear	Standalone	AC/USB	500mA (default) 450-700 configurable by serial pulses	4.15V-7.00V	28V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	No	No but has thermal protection	Yes	Two (charging status and power source)	4.00V	-40 to 85ºC	2mm x 3mm TDFN-Exposed Pad (8 pins)
	MAX8677 A/C	4.2V	Linear	Standalone	AC/USB	300mA-1250mA	4.10V-6.60V	16V	Isolated	4.35V	Yes	Yes	Yes	No	Two (charging status and power source)	4.00V	-40 to 85ºC	4mm x 4mm TDFN-Exposed Pad (24 pins)
	MAX8606	4.2V	Linear	Standalone	AC/USB	with 500mA input-> 250 sys and 225 batt	4.25V-5.50V	14V	Isolated	3.5V (with Vbat = 3.3V)	Yes	Yes (no configurable)	Yes	Yes	Two (charging status and power source)	4.00V	-40 to 85ºC	3mm x 3mm TDFN-Exposed Pad (14 pins)
	MAX8808X/Y/Z	4.2V	Linear	Standalone	AC/USB	465mA	4.25V-6.50V	15V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	No	No but has thermal protection	Yes	Two (charging status and power source)	2.74	-40 to 85ºC	2mm x 2mm TDFN-Exposed Pad (8 pins)
	MAX1551 MAX1555	4.2V	Linear	Standalone	AC/USB	100mA	3.70V-7.00V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	No	No but has thermal protection	Yes	Two (charging status and power source)	3.95V	-40 to 85ºC	SOT23 (5pins)
	MAX8600 MAX8601	4.2V	Linear	Standalone	AC/USB	95mA/475mA (USB)	4.15V-7.00V	14V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	Yes	Not information	Two (charging status and power source)	4.0V	-40 to 85ºC	3mm x 3mm TDFN-Exposed Pad (14 pins)

	IC name	Charge voltage (4.2V)	Control topology	Control interface	Charge source	Current charge (constant current phase) Ichg	Vin (operating input voltage)	Vin max > 5V	Battery and system load isolated/sharing	Vout (output voltage to the system)	Integrated pass transistor	Safety timer charging termination	Temp. Monitoring termination	Reverse protection	Status pins	Undervoltage lockout (UVLO)	Temp. range	Packing
Analog Devices	ADP5063	3.6V (LiFePO4)	Linear	I2C	AC/USB	400mA-1300mA	4V-6.7V	20V	Isolated	4.3-5V (programable)	Yes	Yes (no configurable)	Yes	Yes	One (charging)	2.35V	-40 to 125ºC	4mm x 4mm LFCSP
	ADP5062	4.2V	Linear	I2C	AC/USB	50mA-1300mA	4V-6.7V	20V	Isolated	4.3-5V (programable)	Yes	Yes (no configurable)	Yes	Yes	One (charging)	2.35V	-40 to 125ºC	4mm x 4mm LFCSP
	ADP5061	4.2V	Linear	I2C	AC/USB	50mA-1300mA	4V-6.7V	20V	Isolated	4.3-5V (programable)	Yes	Yes (no configurable)	Yes	Yes	One (charging)	2.35V	-40 to 85ºC	2.6mm x 2mm WLCSP
	ADP2291	4.2V	Linear	Standalone	not specified	depends of the input and the pass transistor	4.5V-12V	?? (minimum 12V)	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	No, external	Yes	Yes	No	One (charging)	No	-40 to 125ºC	8-lead MSOP and 3 × 3 mm LFCSP
Texas Instruments	bq24072/3/4/5/9	4.2V (*4.1V '79)	Linear	Standalone	AC/USB	95mA/475mA (USB) and adjustable	4.35V-6.4/10.4V('74)	28V	Isolated	Vbat +225mV (bq24072) 4.4V(bq24073/4) 5.5V(bq24075/9)	Yes	Yes	Yes	Not information	Two (charging status and power source)	3.30V	-40 to 85ºC	3 mm × 3 mm QFN (16 pins)
	bq24083	4.2V	Linear	Standalone	AC/USB	50mA-1000mA	4.50V-6.50 V	7V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes (no configurable)	No but has thermal protection	No	Three(2 charging status and power source)	2.50V	-40 to 120ºC	3-mm × 3-mm SON (10 pins)
	bq24085/6/7/8	4.2V	Linear	Standalone	AC/USB	50mA-750mA	4.35V-6.50V	20V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	Yes	No	Three(2 charging status and power source)	1.50V-3.00V	0 to 125ºC	3-mm × 3-mm SON (10 pins)
	bq24080/1	4.2V	Linear	Standalone	AC/USB	20mA-100mA	4.50V-6.50V	7V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes (no configurable)	No ('80) and Yes ('81)	No	Three(2 charging status and power source)	2.50V	-40 to 85ºC	3-mm × 3-mm SON (10 pins)
	bq24070/1	4.2V	Linear	Standalone	AC/USB	100mA-1500mA (1000mA typical)	4.35V-16.0V	18V	Isolated	4.4V('70) and 6.0V('71)	Yes	Yes	Yes	Yes	Three(2 charging status and power source)	2.50V	-40 to 85ºC	3,5 mm × 4,5 mm VQFN (20 pins)
	bq24060/1/3/4	4.2V	Linear	Standalone	AC/USB	100mA-1500mA	4.35V-16.0V	18V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	No ('61 and '63) and Yes ('60 and '64)	No	Three(2 charging status and power source)	1.50V-3.00V	-40 to 85ºC	3-mm × 3-mm SON (10 pins)
	bq24030 bq24031 bq24032A bq24035 bq24038	4.2V (*4.1V '31)	Linear	Standalone	AC/USB	100mA-1500mA (1000mA typical)	4.35V-16.0V	18V	Isolated	4.4V('32A and 38) and 6.0V('30 and '31) LDO 3.3V	Yes	Yes	Yes	Yes	Three(2 charging status and power source)	2.50V	-40 to 85ºC	3,5 mm × 4,5 mm VQFN (20 pins)
	LM3658	4.2V	Linear	Standalone	AC/USB	50mA-1000mA	4.35V-6.0V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	Yes (no configurable)	Yes	Two(2 charging status)	No	-40 to 85ºC	3 mm x 3 mm WSON (10 pins)
	bq24020 bq24022 bq24023 bq24024 bq24025 bq24026 bq24027	4.2V	Linear	Standalone	AC/USB	50mA-1000mA	4.35V-6.0V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	Yes (no configurable)	No	Two(2 charging status)	2.50V	-40 to 85ºC	3-mm × 3-mm SON (10 pins)
	bq24010 bq24012 bq24013 bq24014 bq24018	4.2V (*4.36V '18)	Linear	Standalone	AC/USB	100mA-1000mA	3.00V-16.5V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	No but has thermal protection	No	Three(2 charging status and power source)	No	-40 to 85ºC	3-mm × 3-mm QFN (10 pins)

Microchip	IC name	Charge voltage (4.2V)	Control topology	Control interface	Charge source	Current charge (constant current phase) Ichg	Vin (operating input voltage)	Vin max > 5V	Battery and system load isolated/sharing	Vout (output voltage to the system)	Integrated pass transistor	Safety timer charging termination	Temp. Monitoring termination	Reverse protection	Status pins	Undervoltage lockout (UVLO)	Temp. range	Packing
	MCP73811/2	4.2V	Linear	Standalone	AC/USB	85mA/450mA ('1) 50mA-500mA ('2)	3.75V - 6.00V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	No	No but has thermal protection	Yes	No one	No	-40 to 85ºC	SOT-23 (5 pins)
	MCP73831/2	4.2V (4.25V, 4.40V and 4.50V)	Linear	Standalone	AC/USB	14.5mA-505mA	3.75V - 6.00V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	No	No but has thermal protection	Yes	One charging status (DFN version)	3.38V	-40 to 85ºC	2 mm x 3 mm DFN (8 pins) SOT-23 (5 pins)
	MCP73830/L	4.2V	Linear	Standalone	AC/USB	20mA-200mA ('30) 100mA-1000mA ('30L)	3.75V - 6.00V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	No but has thermal protection	Yes	One charging status	3.30V	-40 to 85ºC	2mm x 2 mm TDFN (6 pins)
	MCP73833/4	4.2V (4.35V, 4.40V and 4.50V)	Linear	Standalone	AC/USB	100mA-1000mA	3.75V - 6.00V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	Yes	Yes	Three(2 charging status and power source)	3.45V	-40 to 85ºC	3 mm x 3 mm DFN (10 pins) MSOP (10 pins)
	MCP73826	4.2V (4.1V)	Linear	Standalone	AC/USB	depends Rsense + Ext FET	4.50V - 5.50V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	No, external	No	No	No	No one	No	-40 to 85ºC	SOT-23A (6 pins)
	MCP73828	4.2V (4.1V)	Linear	Standalone	AC/USB	depends Rsense + Ext FET	4.50V - 5.50V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	No, external	No	Yes	No	One (charge complete)	No	-40 to 85ºC	MSOP (8 pins)
	MCP73827	4.2V (4.1V)	Linear	Standalone	AC/USB	depends Rsense + Ext FET	4.50V - 5.50V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	No, external	No	Yes	No	One (charge complete)	No	-40 to 85ºC	MSOP (8 pins)
	MCP73113/4	4.2V (4.25V, 4.35V and 4.40V)	Linear	Standalone	AC/USB	130mA-1100mA	4.00V-16.0V	18V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	No but has thermal protection	Yes	One charging status	4.05V	-40 to 85ºC	3 mm x 3 mm DFN (10 pins)
	MCP73837/8	4.2V (4.35V, 4.40V and 4.50V)	Linear	Standalone	AC/USB	90mA/450mA (USB)	3.75V - 6.00V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	Yes	Yes	Three(2 charging status and power source)	3.35V (USB)	-40 to 85ºC	3 mm x 3 mm DFN (10 pins)
	MCP73123/223	3.6V/7.2V (LiFePO4)	Linear	Standalone	AC/USB	130mA-1100mA	4.00V-16.0V	18V	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	No but has thermal protection	Yes	One charging status	4.05V	-40 to 85ºC	4 mm x 3 mm DFN (10 pins) MSOP (10 pins)
	MCP73841/2/3/4	4.1V/4.2V('41 and '43) 8.2V/8.4V ('42 and '44)	Linear	Standalone	AC/USB	depends Rsense + Ext FET	4.50V - 12.0V ('41 and '43) 8.70V - 12.0V('42 and '44)	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	No ('43 and'44) and Yes ('41 and '42)	No	One charging status	4.40V ('41 and '43) 8.60V('42 and '44)	-40 to 85ºC	MSOP (8 pins '43 and '44) MSOP (10 pins '41 and '42)
	MCP73853/55	4.2V (4.1V)	Linear	Standalone	AC/USB	100ml/475mA (USB)	4.50V- 5.50V	Not information	Sharing	the voltage in the battery (4.2 charging and 3.7 discharging)	Yes	Yes	No ('55) and Yes ('53)	Yes	Two(2 charging status '53) One charging status ('55)	4.40V	-40 to 85ºC	4x4 mm QFN (16 pins '53) 3x3 mm DFN (10 pins '55)
	MCP73871	4.2V (4.10V, 4.25V and 4.35V)	Linear	Standalone	AC/USB	90mA/450mA (USB)	4.40V - 6.00V	Not information	Isolated	Not information	Yes	Yes	Yes	Yes	Three(2 charging status and power	4.27V	-40 to 85ºC	4 mm x 4 mm QFN (20 pins)

Table 3.7: IC charger proposal

voltage of the selected battery (see Section 2.5.5) is 4.2 V, the charger chip should work for this voltage in order to obtain a proper performance of the battery.

Column 3 of Table 3.7 exposes the type of charger: linear, switch-mode or pulse. The most common approach was the linear solution, so the solutions shown are only linear chargers, although there were also switch-mode and pulse ICs.

Column 4 describes the control interface. Some chargers offer configurable parameters or can report some information about the state of charge. These chips have I2C, SMBus or other types of control interfaces. For this project these kinds of interfaces were not needed. The solution designed had to be as simple as possible. Therefore, a standalone solution was a better option.

Column 5 depicts the IC power supply type. This can be an AC wall adapter, a USB port or a direct connection to the mains. However, the last type was not very popular due to the fact that such designs are more complicated than in the other two cases. Some chips could be used with either AC wall adapter or USB topologies. Others were optimized for one of these cases. For this project, the battery had to be charged through a USB cable, since this application involved a PC or an embedded platform with USB ports. Therefore, it was more convenient to use USB.

The charge current (the current during the constant current phase) can be set by an external resistor or by digital inputs, depending on the chip design. Many of the chips designed for USB applications have two digital inputs that allow choosing between the USB standard current values: 100mA or 500mA.

The next characteristic is the voltage range of the input source. The minimum voltage is approximately around 4 V and the maximum value depends on the chip design: 6 V, 7 V or 12 V. The USB electrical specification is inside these ranges. The following parameter is the maximum input voltage and it is related to the voltage input source. The voltage input source is the voltage in which the charger operates correctly. However, sometimes sudden overvoltages appear. Many ICs tolerate high voltages to a certain maximum value. This value constitutes the maximum input voltage and if it is exceeded, the circuit could be damaged or broken. The other dangerous situation is the exact opposite. If the voltage applied to the battery is below a minimum level, the cell could be damaged as well. Thus, the most ICs also feature an undervoltage lockout protection that stops the charging when the voltage reaches this minimum. This voltage is typically between 2.5 V and 4 V. In order to avoid unstable situations (e.g. a fluctuating signal around this value would constantly activate / deactivate the charger), many chips implement a hysteresis cycle. This means that for switching off the charger, the voltage has to be equal to the undervoltage lockout. But for switching the device on again, the voltage has to be above the undervoltage value plus a fixed threshold.

The manufacturers have implemented two variants of charging IC's: The first variant provides just one output pin which is directly connected to the battery. This results in two design options for the application: either both the charger and the rest of the circuit are separated or the charger is integrated into the system. The separate solution is less useful, because when the user needs to recharge the battery, he/she has to detach it from the device and plug it into the charger. After the charging, he/she must again connect the battery to the system. Therefore, many designers have developed a solution to include the charger inside the device. The charger has to be connected to the system load and to the battery, but it only has one output pin. As a consequence the designer has to build a switch circuit to select between supplying the power to the system load through USB or through the battery. Besides of supplying the power to the system load when the USB supply is connected, it must also charge the battery. The circuit will consume a part of the supply current provided to the charger. Thus, if the system load varies, the consumed current is modified which could be a problem for the battery in an integrated charging solution. The charging current must be constant during the constant current phase and the in trickle phase the current must remain low. Therefore, in such designs the battery and the rest of the circuit should not be directly connected. A regulation circuit must be placed between the battery and the system load. This circuit is also connected with the input source, thereby when the input source is active, the system load takes the power from the source and the battery is charged independently. If the input source is disconnected, the regulation circuit will switch to the supply source of the battery.

The other solution developed by the manufactures is the integration of the regulation circuit into the chip. In this case the charger IC has two output pins, one for the battery and another for the system load. Therefore, the designer can implement the charger into the system without regulation and switching circuit. The manufactures have patented their own solutions. Texas Instruments calls it "Dynamic Power-Path Management" and Maxim Integrated calls it "Smart Power Selector". This solution is rather preferred than the first one, since it is simpler and saves space on the board. The solutions shown in the tables differ in "sharing" when the chip has one output pin that must be shared by the battery and the system load and "isolated" when the chip has independent pins for the battery and the system load. Therefore, the output voltage (the voltage to the system load) will depend on the solution implemented. In the "sharing" configuration the voltage will be 4.2 V during the charge and 3.7 V when the battery is being discharged. The "isolated" case depends on the specification of the chip, although these values are usually in the range of 3.5 V and 4.5 V. This feature was important, because the BT module and the microcontroller operate at 3.3V. Consequently, the output voltage had to adapt this value through a low-dropout voltage regulator (LDO). (Reference [45])

The suitable solutions found were all linear chargers. As mentioned, they need a pass transistor to regulate the current from the input source to the battery. This transistor can or

cannot be integrated within the chip. A chip with the pass transistor integrated would be preferred, but this feature would not be essential for the design.

The following parameters in Table 3.7 refer to the protection features of the ICs. The first characteristic is the possibility of including a safety timer. As explained, after many uses, the battery might not reach the minimum level fixed by the charger IC in the last phase of the charge. Therefore, if the charge is not cut off by an external method, it will continue indefinitely. One solution is to finish the charge after a timeout determined by a timer. Many chips have a timer with a fixed value, but others can configure this value by an external capacitor. Another safety measurement refers to a possible overheating of the system. If the power dissipation is high and the chip cannot dissipate it fast enough, the charger starts to overheat. Moreover, when a Li-ion battery is not working properly or is damaged, overheating is also a possible outcome. Thus, many chips have a thermal protection, consisting of an input pin for monitoring the temperature of the battery through a thermistor. If the battery is too hot, the charger will stop the charging cycle. This rarely happens, so many manufacturers include only a thermal protection, instead of a temperature monitoring. The temperature monitoring is usually included in systems submitted to stress, where the battery cell could result in damage. The last protection parameter is the reverse blocking diode. Many pass transistors have a diode between the drain and the source pins that allow a reverse current flowing back to the input source. In order to avoid this situation, some chips have an integrated reverse blocking diode. Others do not have it, so that the designer must be aware of placing one diode between the input source and the charge IC input pin.

Moreover, these chips have output pins for showing the status of the charge through one or more LEDs. They usually incorporate one output pin for a “charging status”-LED and/or an output pin for a “power-on”-LED that displays the connection of a power supply (in this case the USB). The charging status LED starts to blink to indicate an error in the charge in most sophisticated solutions. This characteristic was not critical for the project, so if the chosen IC did not have these LED output pins, it would be a minor concern.

ICs chargers have a temperature operating range. If it is taken outside the limit boundaries, it could be permanently damaged. Thus, the application must work inside this range. The requirement was not a problem, because many of these chips have a range between -40 to 85 ºC. The critical parameter was the battery temperature.

Finally, the last parameter of the table is the IC package. The chips are encapsulated in standardized packages, but the offer is large: QFN, TDFN, DFN, SOT-23 (among others). In general the dimensions for charger ICs are very small (not bigger than 4 x 4 mm). As these packages are standardized, they also have a fixed number of pins, but not all of them are used. There are solutions with 8, 12, 14, and up to 28 pins.

Some charger ICs also incorporate a special metallic pad that has to be soldered to the ground plane of the board in order to faster dissipate the heat from the chip.

There are also other types of characteristics which are not shown in Table 3.7, for example the prize. The prize is important when it comes to cost effectiveness. However, the final IC was chosen because it fulfilled the previous requirements, even if it was not the cheapest one.

The best options were the MAX8606 from Maxim and one type of the bq2403X family from Texas Instruments, although there were other possible solutions, too. These circuits can be used with USB power supply; they are systems with "isolated" system loads and have an internal pass transistor, thermal monitoring, safety timers and a reverse blocking diode. Moreover, the current range for the charge is meets the project requirements and both solutions offer under- and overvoltage protections.

Nevertheless, the final selection was the MCP73831 from Microchip Technology. The electrical design of the charger and the problems appeared during the design phase will be discussed in Section 3.3

3.3 PCB Design

In this section, the different PCB versions which have been designed in course of the work will be explained. The whole process and the problems that emerged until the definitive version was created will be discussed.

3.3.1 First Version

For the first design, some ideas were taken from the AsTeRICS Open EEG project made by Fabian Schiegl. Both projects have the BLE module in common as well as the utilized microcontroller (AT90USB1286) and a battery powered system. For the first version of the PCB, the selection and the connection of the components will be explained in detail. In the following versions, only the changes will be discussed.

The previous Lipmouse prototype used the Teensy++ 2.0 microcontroller board. However, this board would occupy too much space on the PCB. For this project, only the AT90USB1286 microcontroller from Atmel (this microcontroller is also the core of the Teensy++ 2.0) was used. . Thus, the reset line, the decoupling caps, the USB connection and the external oscillator needed to be provided separately on the PCB.

The design was done using the EAGLE CAD software (see Section 2.4). The first part consisted of the elaboration of the schematic. All decisions about the electrical components and their interconnection with the other components were made during the schematic

design. Therefore, the considerations made concerning each component will be discussed one by one in the following pages.

As it was seen in Section 3.2, the Lipmouse had to incorporate a USB connector which would allow charging the battery and communicating from the device to the ARE through a USB cable instead of using the Bluetooth radio-link, which could be desired due to the fact that the wired connection is more reliable than the wireless one. A mini USB connector was chosen, since it is smaller than the common “Type B”-USB connector. This connector has five pins: VCC, D+, D-, ground and the mode pin. The mode pin is used by mini- and micro connectors to support the USB On-the-go specification where the device could act like a host. For this project, the Lipmouse had to work in a "device" mode, so this pin remains unconnected. The VCC or +5 V pin had to be connected to the charging system and the microprocessor USB VCC pin. In addition, a LED with its corresponding current limiting resistor was connected to the VCC trace, in order to show if the system is powered by the USB. However, the VCC line was connected to the rest of the circuit through a 1 µH ferrite bead to remove the high frequency noise that could come from the USB cable. The ground pin had to be wired to the device's ground line and the D+ and D- pins had to be connected to their corresponding microcontroller pins through a 22 ohms resistor.

When the first version of the schematic was designed, the MAX8606 from Maxim Integrated was chosen for the charging system. The characteristics are summarized in Table 3.7 and the complete description can be found in the datasheet [46] . This chip did not have an EAGLE model. Therefore, the symbol and footprint had been designed using the EAGLE library creation tool. A footprint of the component following the physical dimensions of the datasheet was made for the board design. After that, the schematic symbol was created, assigning the pin connections to their corresponding pins in the layout. The footprint is depicted in Figure 3.6.

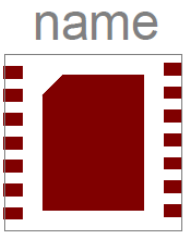


Figure 3.6: IC charger footprint

This charger chip had 14 pins. The input pin was routed to the supply net through the ferrite bead and a decoupling capacitor of 4.7 µF was needed. There were two output pins for the battery and two other output pins to connect the system load. The battery pins were wired to a 2 pin SMD JST connector where the battery was connected. The connector has one terminal connected to the IC charger that had to be connected to ground via a 10 µF capacitor. The other pin was directly connected to ground. The IC charger has two digital

inputs to select the input current or to suspend the IC (both via low level inputs). The input current is distributed for supplying the charging and supplying the load system. Thus, the current absorbed by the system had to be known in order to calculate the charging current. The $\overline{EN1}$ and $\overline{EN2}$ configurations and the corresponding current are illustrated in Table 3.8.

$\overline{EN1}$	$\overline{EN2}$	MODE
0	0	100 mA
0	1	500 mA
1	0	8000 x 2.1V / R_{SETI}
1	1	Suspend

Table 3.8: $\overline{EN1}$ and $\overline{EN2}$ control signals (Source: [46], page 9)

The USB standard values, 100 mA or 500 mA, can be chosen. If none of them were suitable for the application, the current could be set by a resistor connected to R_{SETI} pin. The value of this resistor can be calculated according to the desired current (system current + charging current) following the formula shown on the third line of Table 3.8. The USB port by default only can supply until 100 mA. If the application requires a higher value, it must be negotiated during the USB enumeration process. In the first design, a current of 500 mA was selected, so that $\overline{EN1}$ pin was connected to the ground and $\overline{EN2}$ to the supply trace. The R_{SETI} pin was left unconnected.

The chip has two open-drain output pins, $\overline{CHG}$ and $\overline{POK}$, that display the charging status and connection/disconnection according to the input source (the USB). These pins could be routed to a LED or to a microcontroller's input (through a pull-up resistor) in order to control the charge. In this case, the pin $\overline{CHG}$ was connected to a LED and its corresponding resistor to indicate the charge status. The $\overline{POK}$ pin was left unconnected, since it shows if a power source is connected and this function was accomplished by the LED connected to the USB power supply trace. The safety timer was not used in this design, thus, the pin for this function, $\overline{TMR}$, was wired to ground. In addition, the thermal protection was implemented. According to the datasheet, the pin for the temperature measurement had to be connected to a 10 kΩ NTC thermistor with a β value of 3500 K. The VL pin had to be connected to a 0.1 µF bypass capacitor. Finally, there were two output pins where the system load was connected. These pins also needed a decoupling capacitor of 4.7 µF. In the first schematic version, the decision was to place a mechanical switch between these pins and the rest of the circuit to, manually turn the power supply on and off.

The system output voltage was around 3.5 V. Nevertheless, the BLE module needed a supply voltage of 3.3 V. Therefore, the output from the charger had to be regulated to this value. This function was accomplished by a LDO regulator. The chosen component was the MCP1703 from Microchip Technology, specifically the 3302E/DB version, which has a

3.3 V output voltage. The input pin was connected to one of the switch terminals and needed a 1 µF decoupling capacitor. The output pin had to be decoupled with another 1 µF capacitor. Thus, the supply pins of the microprocessor, the BLE module and the sensors had to be connected to this 3,3V net.

The sip and puff sensor, MP3V7007GP, has 8 pins where only 3 are actually used (see Section 2.5.3 and [27]). One is the supply input, so it was connected to the 3.3 V supply trace. The second pin is the output voltage that was connected to the PF0 pin of the microcontroller, which corresponds to the first Analog-Digital Converter (ADC) pin. The sensor's ground pin was connected accordingly to the ground trace and the rest of the pins needed to remain unconnected. The MP3V7007GP sensor did not have an EAGLE model. Therefore, it was created in using the Eagle design tool. The footprint can be seen in Figure 3.7.

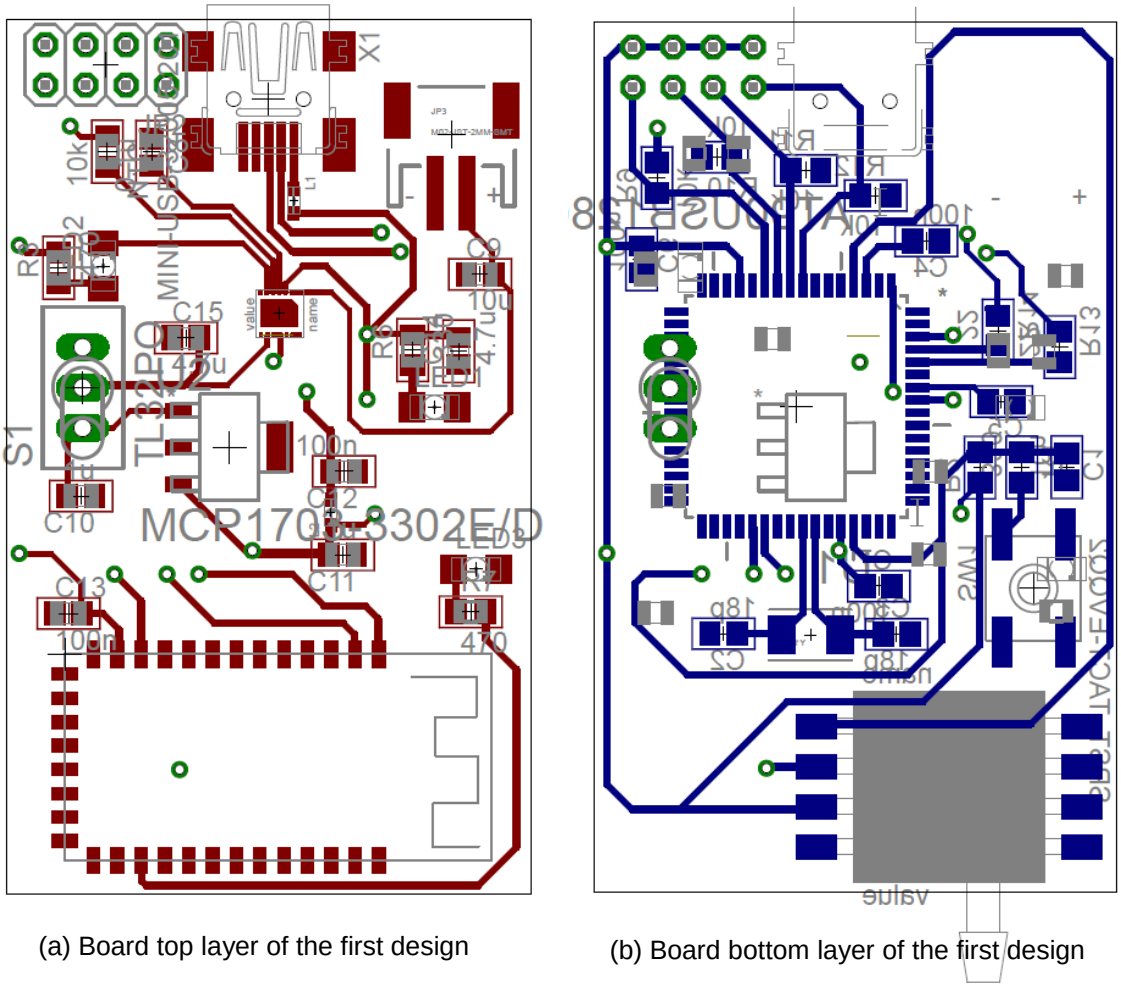
The pressure sensors, FSR149, were connected in a way that one pin connects with the 3.3 V supply net and the other one with the corresponding microcontroller pin (see Section 2.5.4 and [28]). For a right measurement value, the sensor had to be connected through a voltage divisor, so this pin was also routed to a 10 kΩ resistor, which was leaded to ground. The connection pins with the microcontroller were PF4, PF5, PF6 and PF7 pins, which correspond to ADC input pins.

Apart from the mentioned connections, the microcontroller needed some components for a proper operation. First of all it needed a power supply via a connection of the two VCC pins to the 3.3 V voltage supply trace. These pins also need a 100 nF decoupling capacitor. The microcontroller has another supply input, the analog voltage supply (AVCC). This pin needed to be connected to the voltage supply trace, but with a ferrite bead between the traces in order to reduce the possible high frequency noise. Moreover, this pin also required a 100 nF bypass capacitor. Also, the pins UCAP and AREF pins had to be connected to ground through a 100 nF decoupling capacitor. The microcontroller provides a reset signal, which lets the user restart the device. The reset is an "active low" signal, so a button connected to the ground via a series resistor and wired to a pull-up resistor and the reset pin was installed. The values of the resistors were 330 Ω for the series resistor y 4.7 kΩ for the pull-up resistor. The reset pin also needed 10 nF decoupling capacitor to reduce the electrical noise. In addition, this reset signal was connected to the BLE reset pin. The clock signal was provided by an external 16 MHz crystal which had to be connected to the XTAL1 and XTAL2 pins. Each crystal pin had to be wired to an 18 pF capacitor routed to ground. The UART pins, that allow the communication with the BLE device, had to be connected to their corresponding pins on the Bluetooth module. The UART transmission pin (TX), PD3, had to be wired with the BLE module's UART reception (RX) pin. In the same way, the UART reception pin (PD2), should be wired with the BLE module's UART transmission pin. Finally, the four ground pins had to be connected to the ground trace and the rest of the pins remained unconnected.

The BLE module has 34 pins. However most of them remained unconnected. The voltage supply pin had to be routed to the 3.3 V voltage supply trace with a 100 nF decoupling capacitor and the four ground pins connect to the ground net. The UART transmission pin had to be wired with the microcontroller's UART reception pin (the same reception UART pin of the module). The reset pin was wired to the reset circuit described above, so if the button is pressed, both microcontroller and the BLE module will receive a valid reset signal. Furthermore, the BLE module has a pin, PIO1, that can be connected to a LED,which will blink when searching for a pairing device (to establish a connection), and will shine as a pairing is achieved. (It does not shine when the BLE is not powered). Thus, a LED and a 470 kΩ series resistor were connected to this pin.

The whole components wired are shown in Figure 3.8.

After finishing the schematic, the components had to be placed in the board layout and the traces had to be drawn. The microcontroller was the component with most connections. Hence, it makes sense to places this part in the center of the PCB and arranges the other components around the microcontroller. The USB connector was placed in one of the narrow borders, just in the middle, since the enclosure had the hole for the USB cable at this point. The bypass capacitors had to be placed as close as possible to the corresponding pins. The height of the BLE module was similar to the board width. Thus, the best option was to place it in one of the borders. The sip and puff sensor and the pin header for the pressure sensor were also located in one of the boards' corners. While the components were placed in the PCB, the physical traces were drawn. Some traces cross each other and in some cases two connected components were on different layers. Consequently, these traces were conducted from the top side to the bottom side through vias. Ultimately, the whole board was covered by a ground plane on the bottom and another ground plane on the top layer. The ground planes provide an easier interconnection with all ground traces. Moreover, a ground plane reduces the electrical noise and the crosstalk noise between two adjacent traces. Few more vias were needed in order to connect the two ground planes. The result of the first PCB design run is displayed in Figure 3.7. The top layer corresponds to Figure 3.7(a) and the bottom layer to Figure 3.7 (b).



(a) Board top layer of the first design

(b) Board bottom layer of the first design

Figure 3.7: First board design

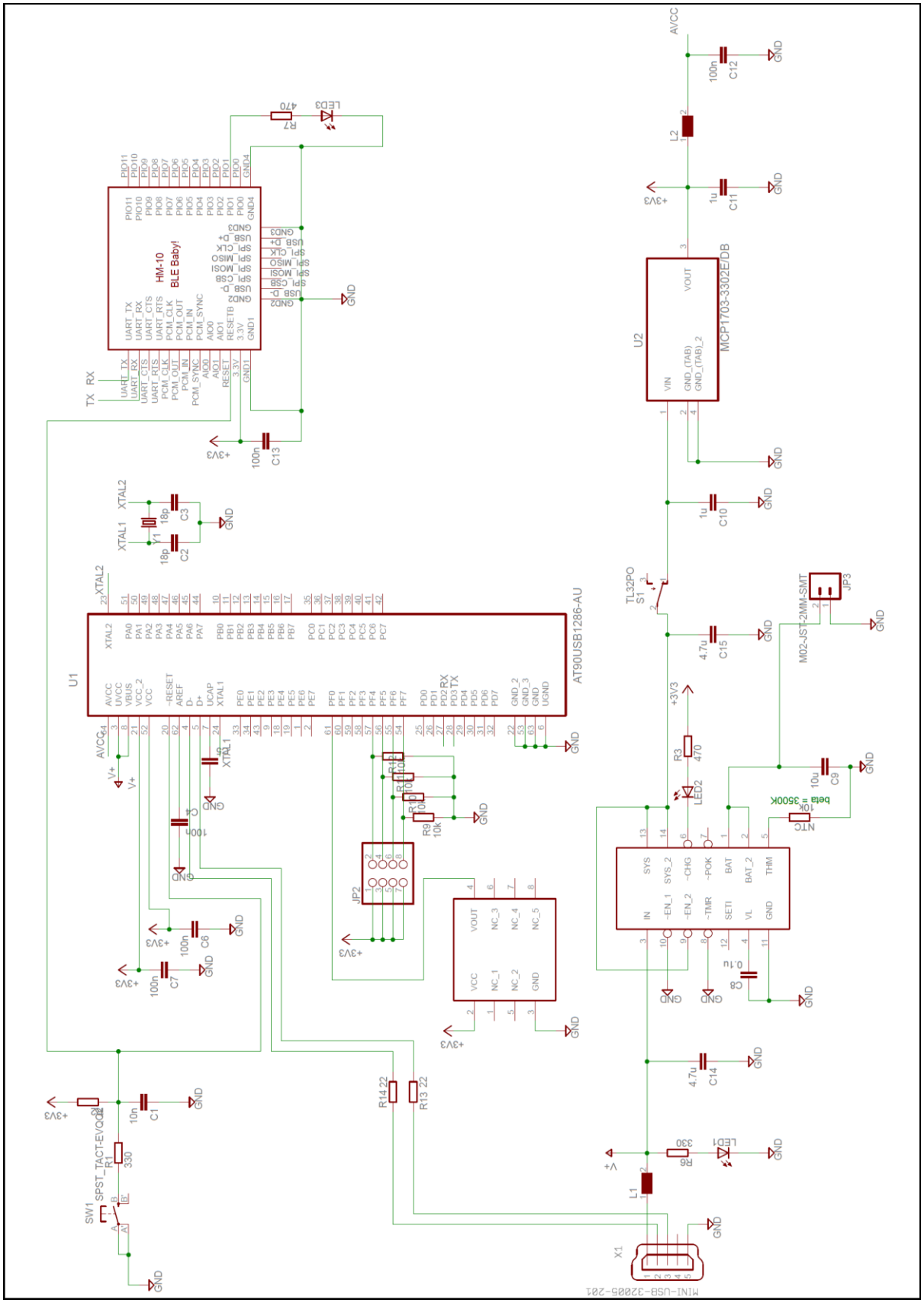


Figure 3.8: First schematic design

3.3.2 Second and Third Version

Before finishing the first design, some errors were detected. The errors found were:

- The antenna of the BLE module had to remain free of metallic elements, as for example traces, pads or planes. The metallic parts disturb the emission pattern of the antenna if they are too close, reducing the quality of the Bluetooth connection. Therefore, the pressure sensor had to be moved to the left and the ground plane had to be removed from the surroundings of the antenna.

- The pin PE2 could be used for the hardware bootloader activation. Accordingly, this pin had to connect to the ground in order to use the bootloader for flashing the microcontroller's firmware. First, this was done via a 10 kΩ resistor. However, this resistor was not necessary and the HWB pin could be wired to ground directly.

- The voltage supply traces had to be thicker, because they need to conduct high a current.

- The package of the ferrite bead was 0603. Nevertheless, there was no 1 µH inductor of this size available at the vendors consulted. Hence, it had to be changed to a 0805 size.

- The switch to turn on and off the system was through-hole and interfered with the microcontroller mounted on the bottom size. The solution was to change it to a SMD right angle switch.

- Placing the three LEDs together in a row would seem better than putting them on the board without any certain order.

- The capacitors and resistors package size chosen was 0805. However, using the 0603 packages would be possible and it would save space in the board.

- In spite of avoiding the sharp traces in the routing process, there were some of them that must be rearranged.

- All together some components could be better placed.

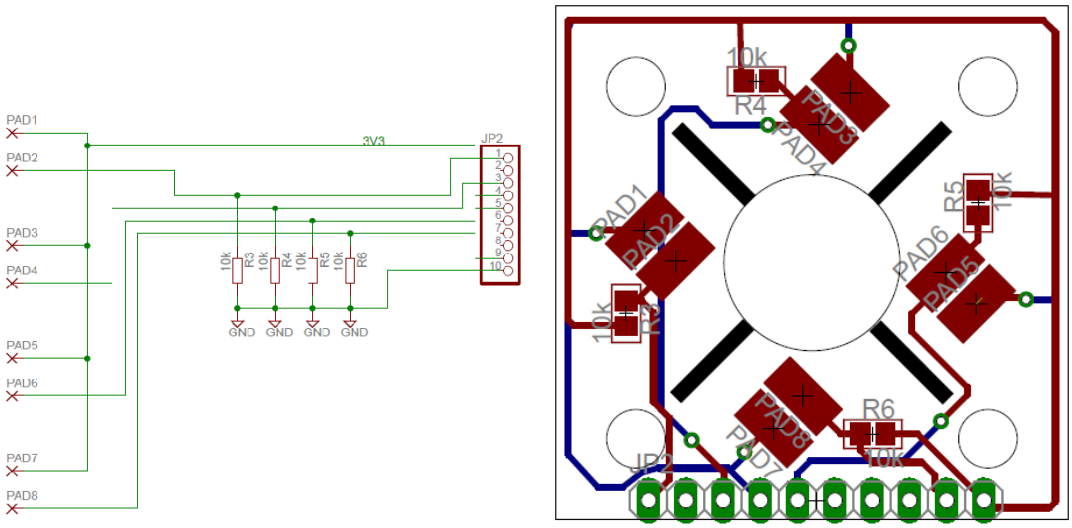
The first design needed a review. The problems encountered were corrected and the improvements were applied during the second and the third versions of the PCB desing. The components were placed more efficiently. However, the third version could be further improved, leading to the fourth (and final) version.

3.3.3 Fourth Version

After starting the fourth design iteration, some design considerations changed. In the first Lipmouse version, FSR sensors were mounted on a small plastic plate as can be seen in

Figure 2.19. The two terminals of each pressure sensor were routed to the PCB through a loose air wire. To improve this situation, the idea was to mount a small PCB (a daughter-board) behind the sensor plate. The metallic contact of the sensor would be soldered to this small PCB and it would wire to the main PCB through a flatband cable. The resistors for the voltage divider for the pressure sensors were also included on the new PCB. This solution was more compact, did not have loose wires and looked more professional that the solution accomplished in the first prototype. In addition, the supply and the ground traces had to be available on this small board. The daughter-board EAGLE design is shown in Figure 3.9.

Furthermore, a JTAG interface was added to the main PCB. The JTAG allows testing and debugging the electrical connections of a microcontroller to the PCB. For further information about JTAG interface consult [43].Moreover, the microcontroller can be flashed through the JTAG interface. The board's JTAG connector also has another function. As the JTAG is not necessary when the Lipmouse is working and the JTAG signals share the pins with the ADC inputs, the small PCB could be connected to the main PCB via the JTAG connector. Therefore, these pins are shared by the JTAG and the FSR pressure sensors.



(a) Schematic design (b) Board design

Figure 3.9: Daughter board design

Moreover, it was noticed that the charger IC, it could be difficult to solder it without special equipment. As it can be seen at Table 3.7 most of the single chip charger solutions have small sized packages and could not be used if easy reproduction of the design is a goal.

Consequently, the charger IC was changes to the MCP73831 from Microchip, which was available in a SOT23-5 package. This package can be soldered to the PCB even by hand.

The drawback was that this chip has a "sharing" load system. Therefore, a regulation circuit which makes the battery charge section and the system supply independent was designed. This circuit followed the recommendations exposed in [45] and is depicted in Figure 3.10.

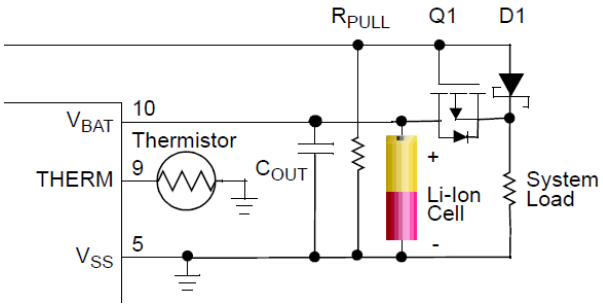


Figure 3.10: Load sharing circuit (Source: [45])

In Figure 3.10, the superior trace was routed to the supply voltage. When the USB is not connected, the R_{PULL} resistor pulls down the Q1 gate to zero. Thereby, the current flows from the battery to the system load. If the USB is plugged in, the transistor's gate is 5 V and the transistor switches off. Therefore, the current cannot flow from the battery to the system load. Instead, the current flows from the USB supply to the system load through the diode D1. Simultaneously, the battery is charged from pin V_{BAT}. The value of R_{PULL} had to be high, otherwise a lot of current would be wasted. A decent value was 100 kΩ.

According to the Section 3.5, the microcontroller's sleep modes were used to reduce power consumption in idle mode. For that purpose, a button was needed for the microcontrollers' wake-up and standby mode control. This was implemented via an external interrupt, so the button was wired to PB0 (INT0) pin. Furthermore, some landing pads where drawn and connected to other external interrupt pins (PE4, PE5 and PE6) in case of the necessity of more external buttons in a future prototype.

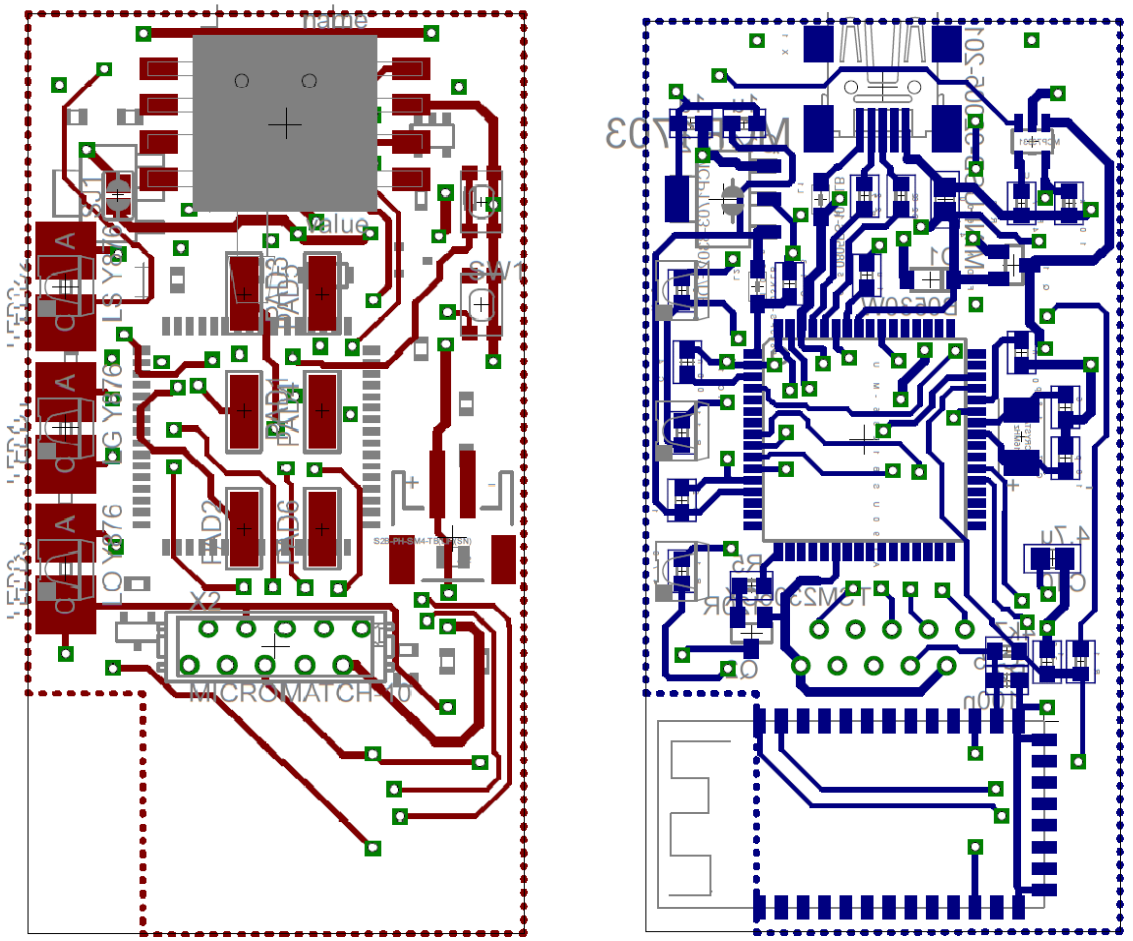
Finally, it was decided that it would be better to turn on and off the BLE module by software instead of a mechanical switch. Therefore, a MOSFET transistor was placed between the BLE ground pins and the ground trace. The transistor's drain was wired to the BLE module's ground pins and the transistor's source was wired to the PCB ground net. The gate was connected to an I/O pin of the microprocessor, PA3. Thereby, when this pin has a high level, the transistor is 'on' and the BLE ground is connected to the device's ground trace. In case of low pin level, the transistor is 'off', so the BLE ground floats, switching off the BLE module.

The final schematic design is depicted in Figure 3.12

The board layout was updated according to the following considerations: The USB connector had to be placed in the middle of one of the narrower edges, like in the previous

designs. The microprocessor had to be mounted in the center of the board, because many components were wired to it. The USB connections were routed to the microprocessor D+, D-, VBUS and GND. The best way to orientate the chip was facing these pins to the USB connector. The BLE module was placed in the opposite edge of the USB connector, in order to not disturb the rest of the components and traces. In this design, the module was allocated sideways too. The orientation chosen is shown in Figure 3.11 (b) to facilitate the routing with the microcontroller. The LDO regulator and the charger IC were situated next to the board edges in each side of the USB connector due to the availability of free space. The decoupling capacitors were allocated as close as possible to the corresponding components. The sip and puff sensor had to be connected at the furthest border in order to bend the plastic tube properly during the assembly of the PCB inside the enclosure. Therefore, it was situated on the top layer. The JST connector, the LEDs and the buttons were also placed at this layer, because there was not enough space at the bottom layer. The light of the LEDs has to cross the enclosure and the edge of the PCB should touch the enclosure. Hence, 90º LEDs were chosen and were placed close to the edge to avoid using light pipes to conduct the light outside of the enclosure. The buttons for the reset and sleep/wake-up, the device were also 90º-types for the same reason. They were mounted close enough to the case so that they can be pushed from the outside without any extension. The connectors for the JTAG and the daughter board were through-hole types, so both sides of the PCB had to be free of elements. The connector had to be placed on the top layer, because it had to be connected to the daughter board. The location chosen was between the microprocessor and the BT module, due to the free space for the holes. In addition, three solder pads were allocated on the top layer, connected with three microcontroller pins, allowing external interrupts to provide additional functions in a future (if needed). Finally, as explained in the first design, a ground plane was added to cover both sides of the board. The final board design is shown in Figure 3.11.

All components and the electrical connections were correct and the board design was fine. It constituted a valid version. Thereby, the board design was sent to manufacture and the components were ordered from the corresponding vendors. The bill of materials which have been ordered is shown in Appendix B.



(a) Board top layer of the fourth design (b) Board bottom layer of the fourth design

Figure 3.11: Fourth board design

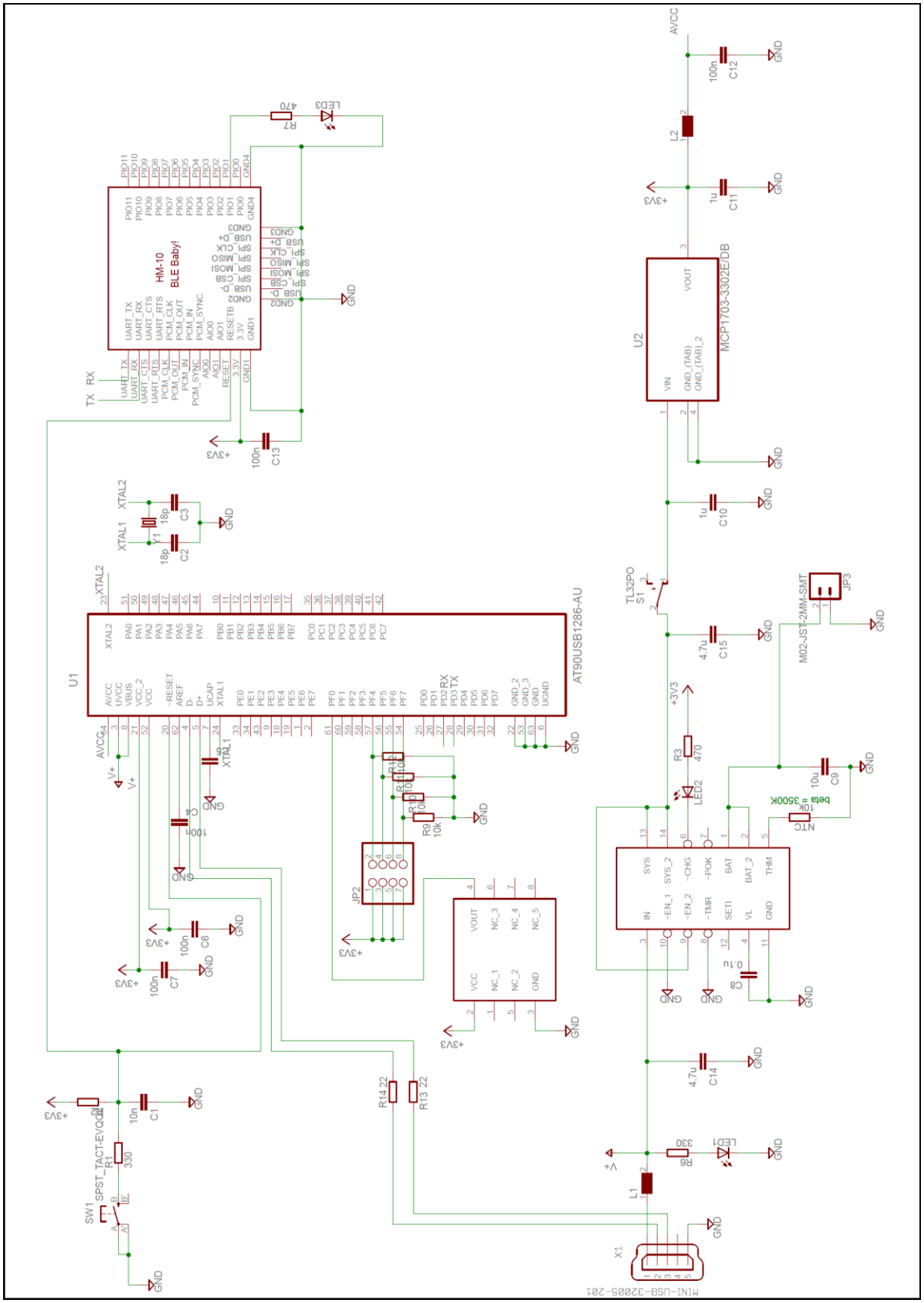


Figure 3.12: Fourth schematic design

3.4 The Plugin

Although the previous version of the Lipmouse used the Teensy++ 2.0 microcontroller, the plugin employed in the ARE model was the Arduino plugin. The reason was that both the Arduino Uno and the Teensy use an Atmel microcontroller, so they are compatible in many aspects. The Arduino plugin can configure the microcontroller digital pins as inputs or outputs and read and write the values of these pins. Moreover, there are six analog inputs which were used in the previous Lipmouse for reading the 4 sensors values through the ADC (so 2 ADC input were unused). In addition, this plugin offers a series of listeners for detecting changes in the pin values.

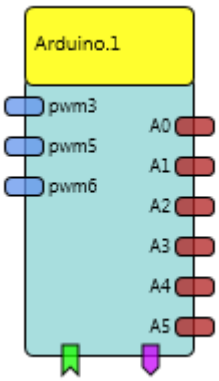


Figure 3.13: Arduino plugin

However, a new plugin was needed for the Lipmouse. Despite the fact that the Arduino plugin can be used for communicating with the Lipmouse's firmware, the Lipmouse is a different CIM which does preprocessing of the sensor values and thus needs its own plugin. If using the generic Arduino plugin for interfacing, the connections with the rest of the plugins in the ACS to create the Lipmouse model would not be very intuitive, as shown in Figure 3.13. The ADC outputs are A0, A1, A2, A3 and A4, however, the correspondence between sensor value and output is not explicit. A user who wants to use this plugin with Lipmouse hardware needs to know how it was implemented. The simpler and more intuitive the design of a plugin is, the easier will it be for a person to use it. Moreover, the Arduino plugin has functionalities that were not required and could confuse the user. Thus, one goal of this project was to develop a specific plugin for the Lipmouse.

First of all, the characteristics of the new plugin were defined: how many inputs, outputs and properties; how the data is displayed and its format; if it needs event listeners and/or triggers; among other features. In this case, the CIM emulates a mouse cursor. The requirements were:

- 3 outputs ports: one for the X value, other for the Y and other for clicking functions (pressure value).
- 1 property: for increasing or decreasing the frequency of the periodical updates.

A new CIM requires a new CIM ID, a new unique serial number and a new CIM feature list. The CIM ID is formed by four bytes. The two most significant bytes identify the type of CIM and the two less significant represent the version of the Lipmouse. The last CIM registered before the Lipmouse was the proximity sensor with a CIM ID of 0xA301. Therefore, to identify the Lipmouse, the unique CIM value 0xA4 was chosen and the version (of the

plugin), was set to 0x01. Hence, the complete CIM ID chosen was 0xA401. For the unique serial number, 0x12345678 was chosen. In case of two Lipmouse CIMs with the same address running in the same ARE, collisions could appear. In that case, one of the Lipmouse CIMs must use a different address (which can only be changed in the firmware source code), but this case is practically impossible.

The feature list is composed by three features. The first feature (command) returns the unique serial number to the ARE. This feature is compulsory in all CIMs since the ARE must know the unique serial number of each connected CIM device. The ARE sends one packet containing this command to each communication port at the beginning of the execution. If there is a CIM attached, it will respond to the ARE with a reply packet, containing its unique serial number. Thereby, the ARE registers the CIM device. The second feature refers to the time between two periodical updates of the sensor data. The frequency of the updates can be regulated depending on the users' requirements. This value is in the range of 0 to 65535 milliseconds. The 0 value means that the CIM should not send any periodic update. The last feature is defined for reading the values of the sensors. The Lipmouse emulates a mouse cursor, so these values correspond to the X- and Y positions and the pressure value (to implement the clicking characteristics or similar functions in the ARE model).

The difference between the right and left sensor value forms the X value and the top sensor minus the bottom sensor compounds the Y value. This operation had to be done in the ACS model when the Arduino prototype was used. However, in the new plugin, the subtraction is done in the Lipmouse's firmware. Thus, the X and Y values are already available to be used in the ACS model. The feature list is depicted in Table 3.9 .

CIM-ID	Feature-address	Access	Description	Data
0xA401 Lipmouse version 1 sensor	0x0000	r	Unique serial number	4 bytes
	0x0001	w	Set ADC value report period	2 bytes: period time 0 (off) to 65535 miliseconds
	0x0002	r	ADC value report	6 bytes: 3 channels of ADC values Byte 1: X low byte Byte 2: X high byte Byte 3: Y low byte Byte 4: Y high byte Byte 5: pressure low byte Byte 6: pressure high byte

Table 3.9: Lipmouse feature list

As it was shown in Section 2.2.4, the first step in the creation of a plugin is the creation of the subfolders and directories, the descriptor files and the source code skeleton. The plugin was made by using the Plugin Creation Wizard tool, in which the three output ports and their corresponding names, data types and descriptions were defined. Also the

periodic ADC update frequency property must be defined in the corresponding field of this tool. The name for the plugin was defined as "Lipmouse" and the category selected was "Sensor" inside the "Standard Input Devices" subcategory. After clicking in "Create plugin", the plugin was created containing this basic structure.

The descriptor files are *build.xml*, *descriptor.xml* and *manifest.mf*. Technically, these files did not have to be modified. In case that some parameters configured in the Plugin Creation Wizard tool need to be redefined, this can be done in the *descriptor.xml* file. (for example changing the name of the output ports, the data type or the description text box).

The source code of the plugin resides in the *LipmouseInstance.java* file. After the plugin creation, it contained a basic skeleton code. However, the functions were not complete and the code for the CIM protocol management and other auxiliary functions had to be included in this file. Some parts of the implementation of the CIM protocol were taken from Arduino's plugin code and adjusted to the Lipmouses' requirements, due to the similarities with the Arduino's plugin. All CIMs must implement the JAVA-interface class *CIMEventHandler*. This interface contains two methods that must be defined in each plugin for the implementation of the CIM protocol: *handlePacketReceived* and *handlePacketError* (see the page 36 from [18]). The ARE will send request packets to the CIM and the CIM will reply to them, or it can send the periodic updates without a request. When these packets arrive at the ARE, it processes them, and forwards dedicated packets to the registered listener functions residing in the plugin code. If an error is detected, it must be managed by the *handlePacketError* method. If the received packet is correct, it must be processed by the *handlePacketReceived* method. One or more actions must be performed, depending on the type of packet. If a reply to a "read feature" packet has been received, there are only two options: the reply contains a unique serial number packet or it contains an ADC report value packet. In the case of a received ADC report value packet, the data (the X, Y and pressure values) must be sent to their corresponding output ports of the plugin. When the CIM sends ADC values periodically in multiple event reply packets the X, Y and pressure values are sent to the output ports as well.

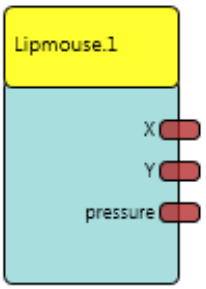


Figure 3.14: Lipmouse plugin

There is just one type of write feature packet contains the property value of the periodic ADC update frequency. Changing the ADC frequency property is handled by the *setRuntimePropertyValue* method of the plugin. This method sends the corresponding write feature packet to the CIM's firmware for changing the ADC frequency when the property has been changed during runtime.

Furthermore, the ADC data sent by the microcontroller must be converted to the format used by the other plugins connected to the Lipmouse plugin, in order for a proper

communication. The data of the plugin output ports must be integer, i.e. 4 bytes. However, the data sent by the microcontroller has two byte length (short integer). Therefore, two of the bytes must be padded with 0 to convert the short integer into an integer. If the short integer contains a negative value, the two bytes left must be padded with ones, i.e. 0xffff.
The process of sending packets to the CIM device firmware is supported by the *CIMPortManager* instance. This class is responsible for managing and controlling all communications with the CIMs. In addition, *CIMPortManager* is in charge of delivering the CIM packets from the Lipmouse to the plugin.

Finally, the *start*, *pause*, *resume* and *stop* method stubs provided during the plugin creation had to be completed. When one of these methods is called, the plugin must send the corresponding packet to the CIM to start / stop the periodic data transfers. This action must be implemented inside these methods. Furthermore, in the *start* method, the plugin code needs to get the connection to the COM port via the CIMPortManager and register it. In the case of the *stop* method, it closes the connnection. Moreover, the *start* method must send the write features command for setting the ADC update report frequency.

After the plugin programming, the last step was the registration of this plugin in the ARE. The activation was done by adding the line: "asterics.sensor.Lipmouse.jar" to the *loader.ini* file (.*AsTeRICS\bin\ARE\profile\loader.ini)*. Now every time the ARE is launched, it will start the Lipmouse OSGI bundle. The Lipmouse plugin is depicted in Figure 3.14. If a person wants to create his/her own ACS model and the Lipmouse plugin is not available in the ACS, the component collection (containing all available plugins of the ARE) has to be updated. There are two options: After activating the plugin in the ARE by modifying the *loader.ini* file and connecting ACS and ARE the tab "Download Component Collection" in the "System" allows all plugins loaded by the ARE to be displayed in the ACS. The other option is to use the "Plugin Activation Wizard" tool available in the "Misc." menu.

3.5 Firmware

3.5.1 Previous Firmware

As mentioned, the first version of the firmware was based on the Arduino firmware, due to the similarities between the Teensy and Arduino microcontrollers. In fact, the plugin used in the first prototype was the Arduino one. In the same way, some auxiliary files were reused in the firmware creation since the Lipmouse CIM also employs the ADC, the timers and the USB in a similar manner as the Arduino CIM's firmware does. In addition, the implementation of the CIM protocol was the same.

The flow structure of the Arduino firmware is summarized in Figure 3.15. The firmware starts configuring the parameters of the ADC, the timer and the USB interface. The clock frequency is set to 16 MHz. After the setup, the main loop, which is continuously executed, begins. The first step is checking if a new byte has been received. When an ARE packet is

received, the bytes are processed one by one using a protocol parse state machine. If the data is correct (the data fits to the specifications), the byte will be stored until the whole frame is received. If during the reception an error is encountered, the protocol will start again from the beginning to search a new frame. In other words, at first, the protocol parser is looking for the "@" (0x40) that marks off the beginning of a frame. When this byte is detected, the protocol knows that the next byte must be the "T" character (0x54). If the second byte was "T", means that a new frame is being received. The third byte should be the ARE ID low byte and the fourth one the ARE ID high byte. The next field expected will be the data length. First, the low byte will be received and -then the high byte. If the value is correct (inside the allowed range), the protocol parser continues to the next field. Thereby, the protocol parser is storing the successive fields of the packet until it is complete.

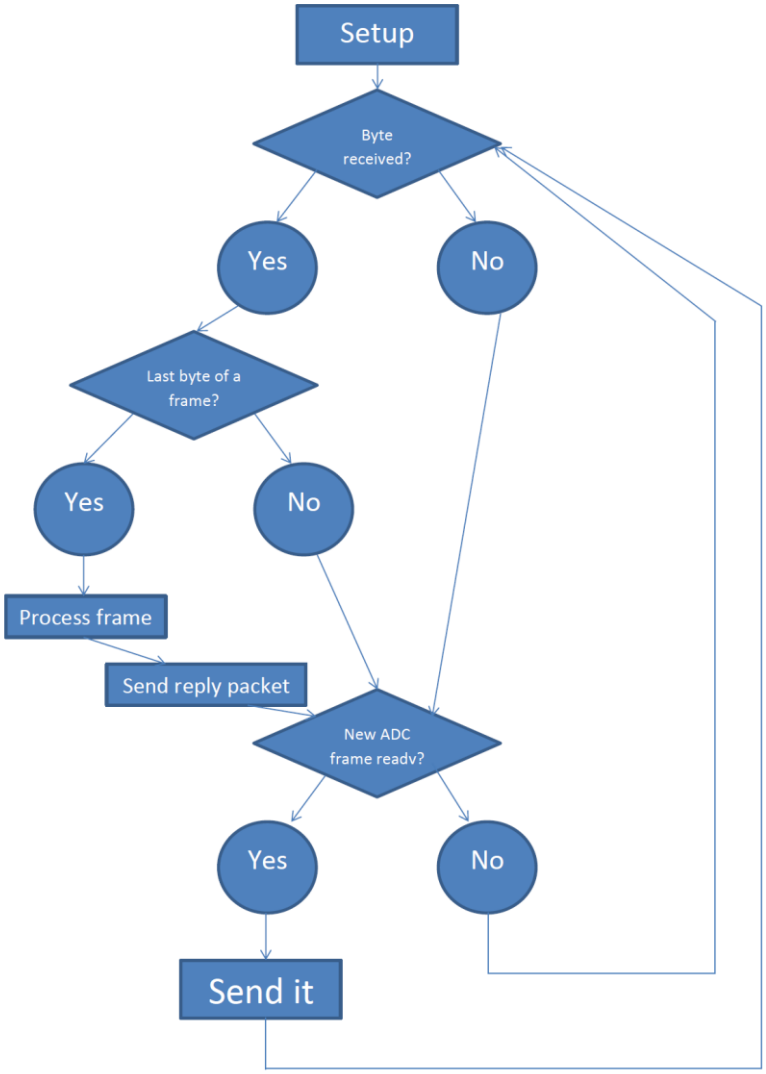


Figure 3.15: Firmware flow graph

When an entire packet was received, it must be processed. During this operation, the ARE reply packet is built parallel, in order to answer the request packet. The first processing task is to identify the type of packet (i.e. the request/reply code). There are 6 possibilities (Figure 2.7): the feature list request, the write feature request, the read feature request, the reset CIM request, the start CIM request and the stop CIM request. For the reset packet no action is necessary. However, if a start packet has been received, the timer must be enabled which triggers the periodic ACD event replies. If a stop packet has been received the timer must be disabled (in order to stop the event replies). After performing these actions, the required reply packet must be sent (to acknowledge the reception of the packet). In case of receiving a feature list request, the feature list is copied to the data field of the reply packet and it is sent to the ARE. For the read/write features, the corresponding CIM feature address must be read from the packet. The Arduino CIM features are:

Read feature

- Unique number
- Get pin values
- ADC report

Write feature

- Set pin direction
- Set pin state
- Set ADC period
- Set pin mask
- Set PWM

Nevertheless, only the unique number, ADC report and the set ADC period features are used for the Lipmouse.

The next step in the main iteration is to check if a CIM event packet must be sent or not. As the first version of the Lipmouse CIM inherited the Arduino source code, there were two types of CIM event packets. One was the pin change status checking. This type of event is not employed by the Lipmouse, so it was erased in the new version. The other was the ADC periodical report. This CIM event is used by the Lipmouse for sending periodically the sensor values to the ARE. The frequency of these events can be set by the user through the plugin property “update period”. By default, this value is 0, so it has to be changed in the ACS model; no report will be sent until this value is different from 0. The time between two events is measured by a timer unit of the microcontroller. When the counter reaches the desired value, the timer creates an interrupt and the ADC event reply frame is generated (by reading the relevant ADC input pins) and transmitted to the ARE.

3.5.2 Firmware Improvements

In this subsection only the improvements fulfilled will be discussed, since the rest of the code is inherited from the Arduino firmware. The CIM ID and the unique number were

substituted by the Lipmouse values. The feature addresses were modified accordingly. The unused features, like “get pin values”, “set pin direction”, “set pin state”, were removed.

The new Lipmouse prototype can be used with a USB cable or through a Bluetooth connection. So there are two operation modes: the USB mode and the BT mode. When a packet is to be sent, it must be delivered by the USB or by the UART interface which is directly connected to the BT module. Consequently, this part of the code was re-written. At the beginning of the code, during the configuration process, the selection of the operation mode was added. The detection consists on: switch on the microcontroller’s USB peripheral; if the USB port is connected to USB host, it will detect a new device attached and it will try to enumerate the device. Therefore, after switching on the USB peripheral and waiting a second (for completing the enumeration process), the availability of the USB connection can be checked. If the USB is operative, the USB mode must be selected. It is preferable to use the USB, rather than the BT, since it is more robust and reliable. Otherwise, the BT mode hast to be employed. Moreover, additional configurations had to be implemented for the BT mode. First, a new Lipmouse features the BT module with the default factory parameters, so it must be configured. This process is repeated every time the firmware starts, although one time would be enough. Alternatively, the configuration status could be saved in the EEPROM memory and checked every startup. The configuration consists of changing the baud rate speed from 9600 bps to 57600 bps, resetting the module and setting the role using AT commands (explained in the Section 3.1.2). The other configuration settings are: to disable the unused functions and peripherals in order to save power and enlarge the life cycle of the battery (it is also done with the USB mode, although in this mode the power consumption is not a problem). The disabled peripherals are the TWI (Two-wire interface), the SPI (Serial peripheral interface) interfaces and the Timer 3. The analog comparator, the brown-out detector and the watchdog functions are also disabled. If the USB is employed, the UART is disabled and if the BT is used, then the USB is disabled.

When the BT mode is being used and the USB is plugged in, the operation mode should change. However, the availability of the USB cannot be checked in each iteration. It would mean switching on the microprocessor’s USB peripheral; wait to the enumeration and to examine if the configuration was done correctly. The probability of connecting the USB per iteration is too low, so in a long term, this process would consume too much power. In order to save energy of the battery, it was preferred to select between USB or BT at the beginning of the firmware execution and if the user wants to change the selection, he/she would only have to stop the ARE, reset the firmware, wait a few seconds for the configuration and launch the ARE again.

One or more FSR sensors could deliver wrong force values (due to mechanical or electrical problems), leading to a drift in the X- or Y-values (and the mouse cursor respectively). The best way to counteract this effect is by adding an opposite offset value to

the current X and Y values. In other words, the device must be calibrated. As mentioned in the PCB design, Section 3.3, a button was wired to an external pin change interrupt of the microcontroller. The firmware was programmed to make a calibration each time the interrupt is triggered, i.e. each time the button is pressed or released. The calibration consists of reading all ADC sensor ports. These values correspond with the drift. Therefore, the values must be subtracted in each future value to compensate the drift. The user does not have to touch the mouth piece during the calibration in order not to alter the measurements. The calibration only removes the constants drifts. The variable drift due to the electronic noise cannot be suppressed with the calibration. Therefore, when the Lipmouse is not touched, there will be values near to zero that may produce a slight drift. These values must be removed or converted to zero by other plugins for example, “AdjustmentCurve” plugin.

The Limpouse does not have a turn on/off button. When the USB is plugged in, since it supplies the device, this is not a problem, but when it is unplugged, the Lipmouse is running until the battery is depleted. In that moment, there will not be enough energy to power the system, so it stops working. Hence, when the Lipmouse is not being used anymore, it must reduce its activity to the minimum, in order to consume the lowest energy possible. The two actions developed for this purpose were to switch off the BT module and to enter the microcontroller’s "deep sleep" mode. As shown on the PCB design, Section 3.3, a MOSFET was placed between the BT ground and device ground to switch on/off the BT module. The activation/deactivation is done via a microcontroller pin, i.e. it is done by software. When this pin is low, the transistor does not conduct the current (the BT ground is floating), so that the BT module is off. When the pin is high, the transistor allows the flux of current and the BT is power on.

The sleep modes halt the different clock systems and the modules associated with these clocks. Depending on which modules are turned off, there are 6 types of possible sleep modes: Idle Mode, ADC Noise Reduction Mode, Power-down Mode, Power-save Mode, Standby Mode and Extended Standby Mode. The ways of waking up the microprocessor are different in each sleep mode: external interrupts or pin change interrupts, TWI, Timer2, ADC, among others (Reference [37]). Figure 3.16 summarizes the active clock domains and the wake-up sources.

The CPU clock controls the general operations, calculations, the Status Register or the Stack Pointer, among other functions. The flash operations are regulated by the flash clock. The I/O clock governs the peripherals like SPI, USART, port pins, timers, and so on. It is also employed by the synchronous external interrupts. The Asynchronous clock is used by the microprocessor's asynchronous operations. The ADC clock controls the ADC module.

The USB clock is independent to the rest of the clocks and it is only running when the USB is enabled. The difference between Power-down and Power-save and between Standby and Extended Standby is that in the first ones the Timer2 is disabled and in the second ones, it is enabled, so it can be used as a wake up source.

When the Lipmouse is not being used anymore, no operation, calculation, register or peripheral is needed. Therefore, the sleep mode, which had to be employed, was the Power-down mode, since it is the one that switches off more modules and, consequently, it is the mode that consumes less power. The same external button was used to make the calibration and to sleep or wake up the microcontroller. The Lipmouse cannot be suspended if the user touches the button by accident. He/she must be aware of what it is doing. Thus, it was established to sleep/wake up the Lipmouse if the user is pressing the button during at least five seconds. If it is pressed shorter, the Lipmouse will not to perform any action (only the calibration).

	Active Clock Domains					Oscillators		Wake-up Sources								
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Osc Enabled	INT7:0 and Pin Change	TWI Address Match	Timer2	SPM/ EEPROM Ready	ADC	WDT Interrupt	Other I/O	USB Synchronous Interrupts	USB Asynchonous Interrupts (4)
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X	X	X
ADCNRM				X	X	X	X(2)	X(3)	X	X(2)	X	X	X		X	X
Power-down								X(3)	X				X			X
Power-save					X		X(2)	X(3)	X	X			X			X
Standby(1)						X		X(3)	X				X			X
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X			X

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT7:4, only level interrupt.
4. Asynchronous USB interrupts are VBUSTI, WAKEUPI, IDTI and HWUPI.

Figure 3.16: Active clock domains and wake up sources in the different sleep modes (Source: [37], page 53)

The implementation for entering the sleep mode was done as follows: when the user presses the button, a pin-change interrupt is triggered. In the Interrupt Service Routine (ISR), the timer is set and starts counting until five seconds are elapsed. At the end of the counting, the corresponding ISR for the timer is triggered. There, the timer is disabled (stops running). An auxiliary variable is used to store the result - if the 5 seconds have elapsed or not. When the button is released, this variable is checked. If the button is released before 5 seconds, the timer stays disabled and the normal execution must continue. Therefore, the Lipmouse will not enter in the suspend mode.

If the timer reaches 5 seconds, which means that the button is still pressed, the Lipmouse must be suspended. Before sleeping/waking up the microprocessor, the BT module is either turned off or on. In fact, this is done in the timer's ISR.

Additional considerations had to be taken into account. The microcontroller cannot be sent to sleep mode inside the timer ISR, since when the user will release the button, this action will wake up the microcontroller again. In the pin change ISR, as well as in the timer ISR case, the code for sending the microcontroller to sleep mode could not be inside the ISR , since the program execution would be halted inside the ISR. When the button is pressed again, the interrupt would not be triggered, because a previous one is still being processed. In conclusion, the sleep mode management must be done outside of the ISRs, i.e. in the main program. This is done through another auxiliary variable. Another issue to consider is the bounce effect of the button. When a mechanical button is pressed, the signal produced does not change from one level to another level immediately. Due to the components elasticity and the vibrations, there is an unstable interval during a short period of time. The bounce effect is depicted in Figure 3.17.

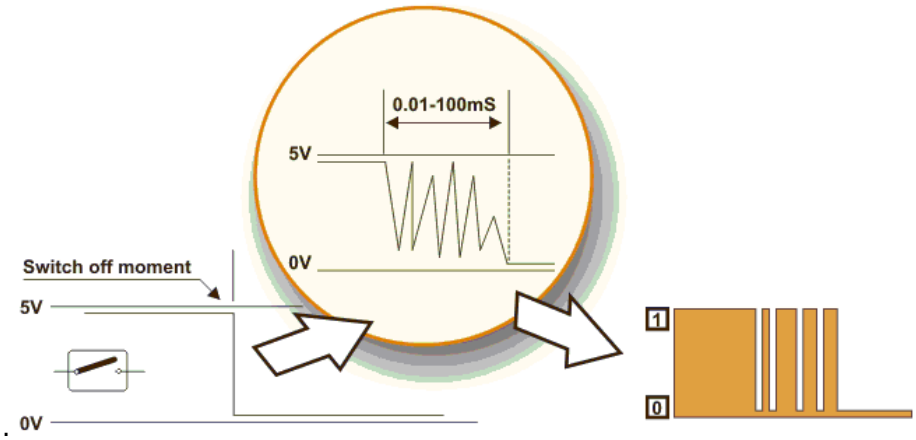


Figure 3.17: Bounce effect (Source: [47])

The bounces may trigger undesired pin-change interrupts. The solution implemented to avoid these unintentional interrupts, was to add a delay to the pin change ISR, to ignore subsequent pin changes. The time cannot be set too long, because if the button is released during this delay, the interrupt will be missed. The delay chosen for this application was 50 ms. However, a delay is not an optimal solution. The CPU is idle during the delay and cannot continue the normal execution of the firmware. There are hardware solutions, for instance a simple RC filter or there are specific ICs that filter the signal as MAX6816 from Maxim Integrated. These solutions need to modify the PCB design. Thus, it was preferable a software solution. A simple counter that is incremented (or decremented) until reach a value that is bounce-free. Nevertheless, the number of transition is not constant. If the value selected is too high, the count will not be reached. Consequently, the calibration will not be done or the sleep mode will not be set. Another option is when a

transition is detected, sample the button output. When a representative number of samples have the same level, the signal is stable and bounce-free. Thus, the calibration or the sleep mode action can be performed. This solution requires a firmware modification. It should be developed in a future work since the delay solution was kept in this thesis.

3.6 Fuses and Memory Lock Bits Configuration

After downloading the firmware, the microcontroller fuses must be configured according to application requirements. The fuses cannot be (un)programmed using FLIP program. This must be done with the AVR Studio using the AVR Dragon. The fuses which should be configured are:

- Once the microcontroller was flashed, the fuse HWBE had to be activated and the reset button had to be pushed for downloading other firmware. When the HWB fuse is programmed (HWBE = 0), it obliges the bootloader execution after a reset, instead of starting to execute the firmware. The bootloader allows flashing the microcontroller with new firmware. When the definitive version of the firmware was downloaded, the HWBE function had to be disabled, in order to reset the device without starting the bootloader (HWBE = 1).

- The fuse CKDIV8 is programmed (CKDIV8 = 0) by default. This fuse determines an initial value for the Clock Prescale Register. If it remains “programmed” the system clock will be prescaled by 8 at star up. Therefore, the fuse must be set as unprogrammed (CKDIV8 = 1).

- The microcontroller uses a 16 MHz ceramic crystal oscillator as a clock source. The clock source is selected by the fuses CKSEL3, CKSEL2, CKSEL1 and CKSEL0. The default values are: CKSEL3 = 1 (unprogrammed), CKSEL2 = 1 (unprogrammed), CKSEL1 = 1 (unprogrammed) and CKSEL0 = 0 (programmed). These values correspond to a ceramic resonator (slowly rising power) with a frequency range between 8.0 – 16 MHz. Therefore, these fuses should not be modified.

For further information about microcontroller fuses consult [37], page 367 and ff.

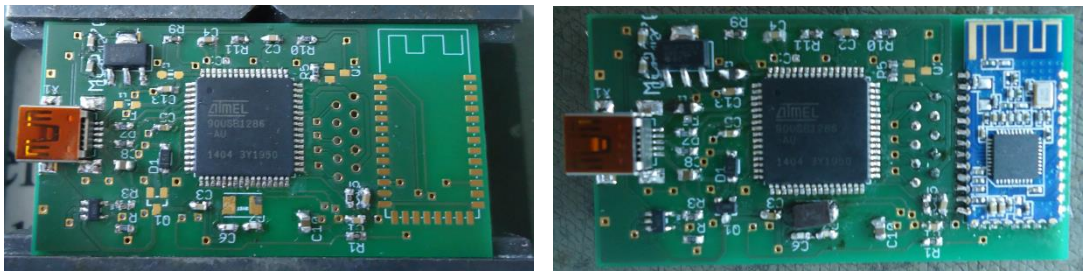
Besides fuses bits, there are other special bits that must be taken into account, the data memory lock bits (see [37], page 366). If the bootloader lock (memory protection in boot section) bits are disabled, the bootloader will be erased during the firmware download. Therefore, the FLIP tool cannot be used for subsequent firmware downloads. In this case the AVR Dragon connected via JTAG is the only way to change firmware. By default, the bootloader lock is disabled. Hence, if a user wants to use the FLIP for flashing the microcontroller, he/she must enable the bootloader lock using also the AVR Studio and the AVR Dragon.

4 Tests, Evaluations and Results

4.1 Board Assembling and Test

The PCB designs described in Section 3.3 were sent to a board manufacturer to build the daughter and the main boards. The components for the boards were also ordered. When all components arrived, they were soldered to the corresponding boards. Four copies of each board were requested in order to make four prototypes.

The first board was soldered by hand. The components were placed in their corresponding positions on the PCB. The metallic contacts had to touch the solder pad. Then both parts were heated with the soldering iron and when they were hot enough, the tin was melted with the head of the soldering iron connecting the component to the solder pad. The components were soldered one by one. First, the most difficult ones were soldered (the more elements surround the target component, the more difficult it is to solder). The soldering process is depicted in Figure 4.1.



(a) Main board during the soldering process

(b) Main board finished

Figure 4.1: The soldering process of the main board

The next step was to check if all electrical connections were right. If there was a short circuit and the board is powered, the board or USB supply could be damaged. This inspection was done with a multimeter and some faults were discovered. The first error was a via that was close to a solder pad with an excessive amount of tin. The amount of tin was enough to short circuit the solder pad and the via. The surplus material was removed from the via with solder wick. Another problem found was that the Schottky diode was incorrectly soldered. The direction of the diode was wrong. The diode was removed and soldered in the correct position. The LEDs did not have any mark to guess the correct direction and they were also soldered in the wrong direction. Hence, they had to be re-soldered in the suitable direction.

The other three boards left were soldered using a reflow oven. This process consisted of applying a solder paste to the solder pads of the board through a metallic mask. Once the solder paste had been applied to all solder pads, the components were placed in their corresponding positions on the solder paste.

After that, the board was put into the reflow oven. The oven heats the board until the solder paste is melted, using a pre-defined heat profile for about 10 minutes. Then the board is cooled, solidifying the tin between the solder pads and the metallic connectors of the components. All components cannot be soldered with this technique and some of them had to be soldered by hand, as for example the BT module. A better, robust, effective, faster and more professional result was achieved with this technique, rather than using the traditional hand soldering.

The components of the daughter board were also soldered. The resistor and the FSR sensors were soldered in their corresponding places. Afterwards, one of the small plastic plates was glued to the board and the head of the sensors was glued in front of the plate. Finally, the connector for plugging it to the main board was soldered. The result is displayed in Figure 4.2. For connection of the board to the AVR Dragon, a JTAG adapter cable was made. A male JST-connector compatible with the board connector was soldered to a 2x5 pin header which plugs it into the JTAG cable of the AVR Dragon. Figure 4.2 shows a picture of this cable.

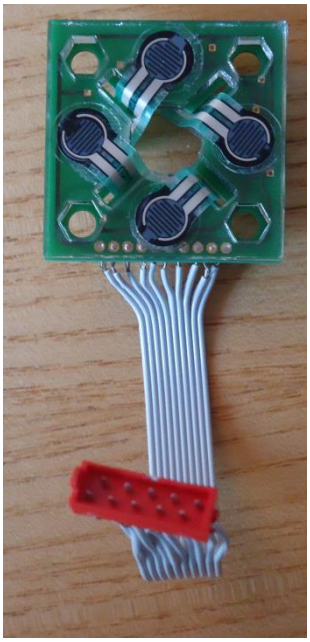


Figure 4.2: The daughter board finished

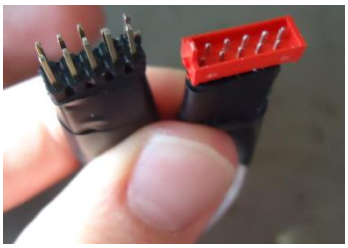


Figure 4.3: JTAG transition cable

When all electrical connections were correct, the USB cable could be plugged into the Lipmouse CIM to test the software. The first time, drivers for the USB were installed and the USB was enumerated correctly. There was no previous firmware flashed on the microprocessor, so the bootloader was executed and the microprocessor could be flashed with the Atmel FLIP tool.

4.2 Software Test

As will be explained in Section 4.6, there was a problem in the board that could not be solved satisfactorily. This malfunction causes a reset in the firmware when the BT module is switched on via the MOSFET transistor. After the firmware execution the BT module is turned on originating an infinite loop resetting continuously the firmware.

Therefore, the board was tested without the MOSFET transistor. The solder pads of the source and the drain were soldered through an air wire. Thus, the BT module ground was connected directly to the device's ground and the BT module was always ‘on’. Then, the board was working without any problem. The USB and the BT mode both worked as expected. The calibrations were done properly and when there was an unintentional drift, the user button was pressed and the measures became zero, showing that the calibration worked. The Lipmouse suspension was also checked. When the button was pressed for 5 seconds or longer, after releasing it, the microprocessor was sent to sleep. No sensor value was sent anymore, the activity of the Lipmouse stopped. If the button was pushed again for 5 seconds or longer, after releasing the button, the Lipmouse continued with its normal operation. If the button was pressed for less than 5 seconds, the Lipmouse kept on sleeping or running, respectively.

The sensitivity of the sensors was also tested in the ARE. According to the sensor pressed, the mouse cursor moved following the desired direction. Therefore, with this set-up, the Lipmouse plugin software and the board firmware worked as intended.

As the firmware works when bypassing the MOSFET transistor terminals with an air wire, the following step was to simulate the switch operation in a manual way. The air wire was modified with a pin header connector in order to connect and disconnect the ground by hand. The board was tested again. When the suspension button was pressed during five seconds, the microcontroller went to sleep and the air wire was disconnected manually. Thus, the BT module was shut down and the activity at the ARE was stopped. Before waking-up the microcontroller, the air wire was connected again. Then, the button was pressed for five seconds to wake up the controller. However, the Lipmouse plugin did not react. This occurs because the firmware switches off the Bluetooth module during the “sleep mode”. After establishing again the normal operation, the module is switched on so a reset is triggered due to the problem mentioned above. The solution was to stop the ARE, to reconnect the air wire, to wait for the establishment of the BT connection, to launch the ARE again and to press the user button in order to wake up the microcontroller. The result was that the ARE worked again since the switch-on function was done manually instead of by software and the reset only happened once.

Finally, the USB mode never showed a problem again. The USB mode worked correctly as expected. To change between the USB mode and the BT mode, the USB cable must be

plugged (for USB) or unplugged (for BT) and the board must be reset through the reset button. Then the firmware starts again and detects if the USB is available. Otherwise, BT mode is selected. In order to change the mode, the following actions should be performed: the ARE must be stopped, the USB cable should be dis/connected and the ARE should be restarted. In the case of the BT, the ARE cannot be launched again until the Bluetooth connection is established.

4.3 Current Measurement

The current consumed by the Lipmouse was measured in the different types of scenarios. A multimeter with a resolution of milliamperes was used for the empirical measurement of the current and the battery cable was manipulated, in order to place the multimeter in series with the battery cable. Thus, the current flowing in/out of the battery could be measured. The measures were taken only in the BT mode since the power consumption is not a critical issue when connected to USB.

The theoretical current consumption is:

- The microcontroller (active, 8 MHz and V_{CC} = 3 V): 5 mA typical, 10 mA maximum.
- The microcontroller (power-down mode, watchdog and brown-out detector disabled, V_{CC} = 3 V): 2 µA typical.
- The Bluetooth module (active): 8.5 mA.
- The LDO: 2 µA typical, 5 µA maximum.
- The charger IC (charging): 2500 µA typical, 3750 µA maximum.
- The charger IC (charge complete): 260 µA typical, 350 µA maximum.
- The charger IC (standby): 180 µA typical, 300 µA maximum.
- The charger IC (shutdown): 28 µA typical, 50 µA maximum.
- The LED and the resistor of the BT module status (assuming that PIO1 has an output voltage of 3.3 V; not specified in the data sheet): 7.02 mA.
- The transistors: A transistor, when conducts, part of the current is dissipated as heat. It means that a small part of the current is absorbed by the transistor showing a resistive behaviour. It is described by the R_{DS} (ON) parameter. The graphs of the data sheet do not offer enough resolution since the current is very small. In the case of the FD360P, R_{DS} (ON) = 100 mΩ. Thus, V_{DS} = 0.3 mV (calculated approximately with transistor equations), implies that the current absorbed by the

transistor is 3 mA. An analogous result for the TSM2306 transistor shows a consumption of 2 mA. These values are approximations.

- The ferrite beads and the capacitors do not dissipate electric power. The resistive effect is low, so it can be neglected.

- The power dissipated by the voltage divider of the pressure sensors was also neglected. The FSR resistance is >10 MΩ (no pressed) or several kΩ (pressed), plus the 10 kΩof the voltage divisor resistor. In the worst case, it is below 0.3 mA.

The current measures were done with the air wire version board due to the problems with the BT power switching which have been described in the previous section. The measurements were done without the MOSFET transistor. The Lipmouse in the active mode should consume theoretically: 5 to 10 mA + 8.5 mA + 0.3 mA + 7 mA + 3 mA ≈ 23.8 to 28.8 mA. If the microcontroller was in the power-down mode and the BT was powered, the consumption would be: 2 µA + 8.5 mA + 0.3 mA + 7 mA + 3 mA ≈ 18.8 mA. If the BT system was switched off (BT module + LED), it would be: 2 µA + 0.3 mA + 3 mA ≈ 3.3 mA. The issue is that the 7 mA of the BT module status LED and the 3 mA consumed by the load sharing system transistor may not correspond with the reality.

According to the empirical results, when the Lipmouse is running in normal operation, it consumes almost 25 mA, Figure 4.4 (a). When the microprocessor is sent to sleep and the BT is still powered on, the current measured in this case was approximately 15 mA, Figure 4.4 (b). If the BT system is shut down, the consumption goes down to 3.75 mA, Figure 4.4 (c). These results agreed with the theoretical values: 25 mA is inside the range 23.8 to 28.8 mA, 15 mA ≈ 18.8 mA and 3.75 mA ≈ 3.3 mA.

Furthermore, the current during the battery charge was measured. The resistor for setting the current in the fast charge period was selected in order to have a current of 200 mA during this phase. The precondition current, i.e. trickle charge period before the fast charge phase, is between 7.5 - 12.5 % of the value of the fast charge current. This leads to a current value between 15 mA - 25 mA. The measures were: 23 mA during the precondition period and 199 mA during the fast charge phase matching with the theoretical values.

The duration of the battery can be determined approximately through the power consumption. The capacity of the battery is 1000 mAh. This value decreases slightly with the number of dis/charging cycles. The Lipmouse consumes 25 mA. Therefore, the duration of the battery will be 40 hours. The Lipmouse has enough autonomy to be used for a couple of days. Moreover, a current of 200 mA means that the battery will be completely charged in approximately 5 hours. In conclusion, the Lipmouse will be always ready to use since it can be used during the day and charged during the night.

(a) Current consumed by the Lipmouse running in normal operation

(b) Current consumed by the Lipmouse in the sleep mode and the BT switched on

(c) Current consumed by the Lipmouse in the sleep mode and the BT switched off

Figure 4.4: Current measurement

4.4 The Bluetooth Coverage

Another fact of interest is the range where the device can be used. The different types of obstacles cause: diffractions, reflections (multipath effect), scattering, absorptions or guide wave effects. These effects and other applications using the same frequency-bands inside the Lipmouse coverage, degrade the RF signal. Moreover, the media is not static and is continuously changing (shadowing and fading). There are not two identical scenarios. Therefore, the coverage values are always estimations.

The typical scenario of the Lipmouse usage is inside a room where the user handles the computer from a few meters distance according her/his necessities. Since the user needs to see the computer screen, in most cases the Lipmouse and the dongle receiver will be in line-of-sight, i.e. without obstacles between them.

There are more or less complex models for RF coverage and many of them were deducted from collections of empirical measurements. These models offer good approximations

under certain conditions as for instance Okumura-Hata or Walfisch-Ikegami (chapter 4 of COST Action 231, [48]). On the one hand, the simplest models do not offer accurate values, only a general idea of the maximum distance between the transmitter and the receiver. On the other hand, the most complex models have a lot of variables to take in account and a deep analysis of the whole parameter space is required. These models are difficult to compute. However, they offer better results.

The path loss computation using the simplest model of indoor propagation models from COST 231, the "one-slope model" ([48]), is:

$$L = L_0 + 10 \cdot n \cdot \log(d)$$

where
L_0 = the path loss at 1 meter of distance,
n = power decay index,
d = distance between transmitter and receiver in meters.

The power emitted by the transmitter is 0 dBm. The sensitivity of the receiver is not explicitly given in the data sheet of the BT devices. Nevertheless, the BT standard says that the minimum sensitivity of a receiver must be at least -70 dBm for a bit error rate of 0.1 %. Taking these parameters into account and assuming L_0 = 33.3 dB and n = 4.0, (the same values that the Table 4.7.2 of [48]), leads to:

$$L = L_0 + 10 \cdot n \cdot \log(d)$$
$$70\ dB = 33.3\ dB + 10 \cdot 4.0 \cdot \log(d)$$
$$d = 8.27\ m$$

8.27m is the maximum theoretical distance between the transmitter and the receiver. This result is not accurate, but gives an idea about how much the maximum distance between the Lipmouse and the PC with a direct line-of-sight.

Moreover, some empirical measurements of the Lipmouse coverage were done. For that, various samples were taken among different points of two different rooms. The rooms tested were a 4 m x 3 m bedroom and a 8 x 14 mm hall. The Figure 4.5 shows an image of the test realized in the hall. The result obtained was that the Lipmouse worked properly without any failures in the different points tested in both rooms. The range was also measured between rooms. With only one wall between the PC and the Lipmouse, it was still running. But with 2 or more walls, the application began to lose packets.

Figure 4.5: Test of the indoor coverage

The coverage was measured outdoors in order to measure the limit of the range. It was observed that with a separation distance of 40 meters the application begins to receive faulty packets. An example of the defective packets received is depicted in Figure 4.6. This produces a loss of sensitivity in the cursor movements. The stumbling movements make it impractical for the use of the Lipmouse when the ARE is continuously receiving faulty packets. However, the dongle and the BT module were still paired. To lose the BT connection, a distance of at least 60 meters was necessary (these measurements were done with direct line-of-sight.)

The test revealed that the BT connection works well and the Lipmouse could be used without any problem in a wide range of applications.

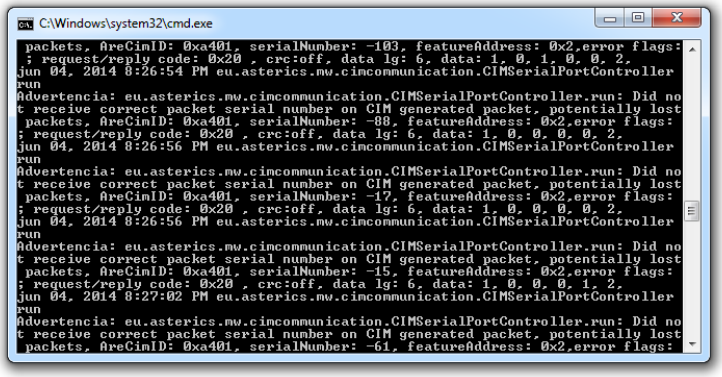


Figure 4.6: Faulty packets received

4.5 Applications

The objective of the Lipmouse is to provide a mouse emulator for people with motor problems in their upper limbs. A way of interacting with a computer will be provided to such

individuals. For that purpose, the required elements are the Lipmouse, a computer with the AsTeRICS software installed, and a suitable AsTeRICS model which interconnects the Lipmouse plugin with other suitable plugins to provide the desired functionality to the user. This section will focus on how a user or a developer must configure such a model.

For setting a simple model, the steps required are:

- Open the ACS editor.
- When the ACS is ready, click on the tab "Components".
- Select "Sensors" →"Standard Input Devices" → "Lipmouse" and drag the Lipmouse icon to a suitable position.
- Select "Processors" → "Basic Math" → "Threshold" and move it next to the Lipmouse.
- Select "Actuators" → "Input Device Emulation" → "Mouse" and move in front of the Lipmouse.
- Connect the output "x" of the Lipmouse with the input "mouseX" of the Mouse. An analogue step must be performed for the "y" output and the "mouseY" input.
- Connect the "pressure" output with the input "in" of the Threshold plugin. Connect the event trigger of the Threshold with the event listener of the Mouse. Select the event channel created, and for the "leftClick" event listener, choose "eventNegEdge".
- Select the Lipmouse and set the property "periodicADCupdate" to 30.
- Select the Threshold and set the properties as: "thresholdHigh" to 490, "thresholdLow" to 490 and "eventCondition" to "above->below". The rest of the properties do not need to be modified.
- Select the property "enableMouse" of the Mouse properties.
- Launch the ARE and connect it to the ACS through the tab "System" → "Connect to ARE" in the ACS editor.
- Click "Upload Model" and the "Start Model".
- Now the mouse cursor will be controlled via the Lipmouse and left mouse clicks can be generated via sip/puff actions.

The result obtained is depicted Figure 4.7.

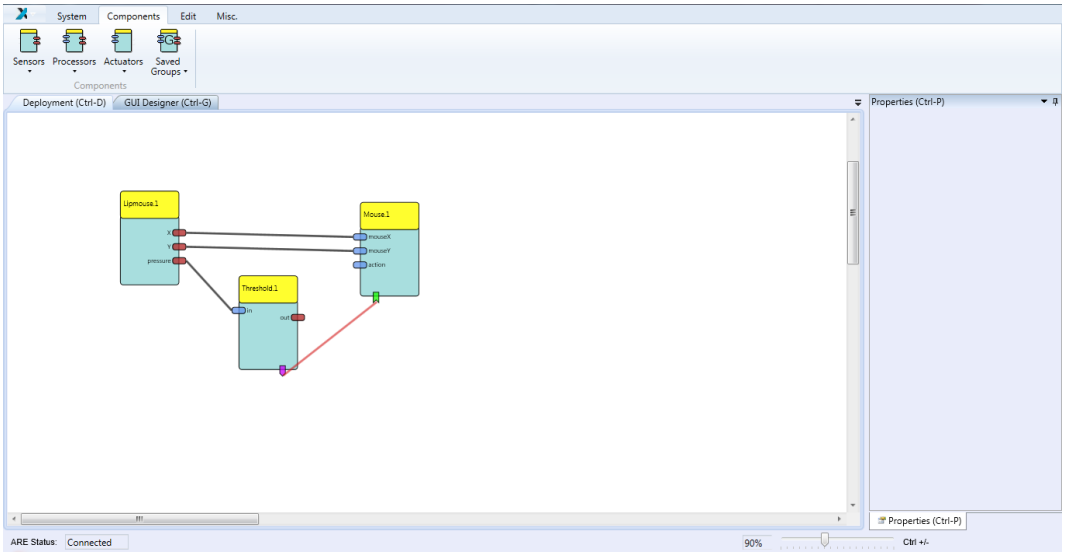


Figure 4.7: The example ACS model

Due to the electrical noise, this model does not work optimally, since there is a slight tremor in the cursor. As the electrical noise is fluctuating continuously, it could not be suppressed with the calibration. This problem can be solved with the AdjustmentCurve plugin ("Processors" → "Signal Shaping" → "AdjustmentCurve"). With this plugin, samples near to zero, caused by the electrical noise could be rejected. An example of this is the model showed in Figure 4.8.

The Figure 4.8 is a functional model and can be used without any tremor problem. At this point, the Lipmouse is ready for operation. The steps to follow are: first of all the ARE must be launched and the ACS model must be uploaded. Secondly, the Lipmouse hardware must be powered on or if it is already turned on, it must be reset. Then the model can be started. If a problem occurs during the execution or the user wants to wake up the device, the user must stop the ARE. After that the user must reset the Lipmouse and, in the case of the BT mode, he/she has to wait until the BT connection is established. Finally the ARE should be started. More sophisticated models can be developed including the use of other sensors and actuators. The combinations and the possibilities are wide.

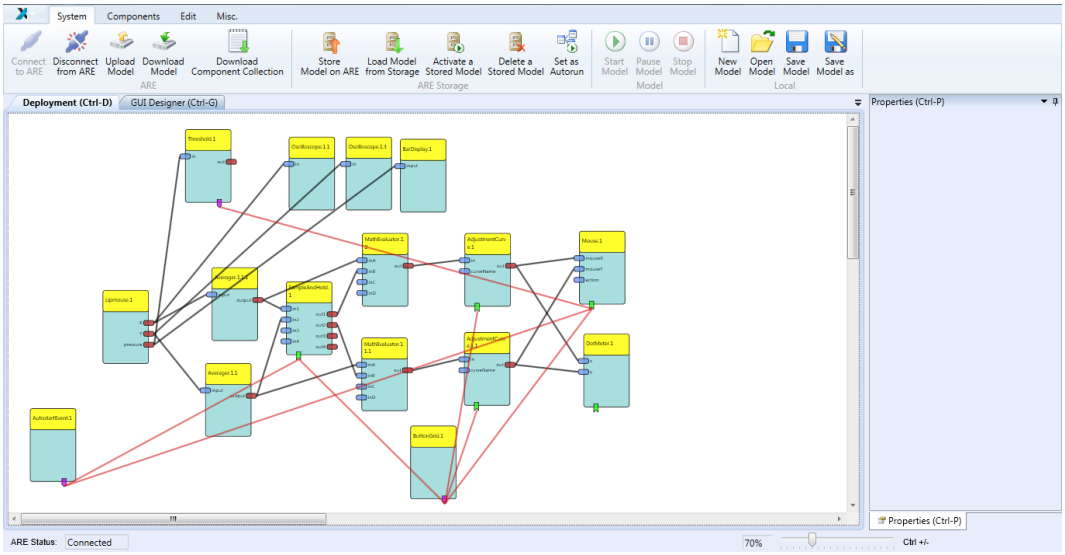


Figure 4.8: A more complex ACS model to use the Lipmouse

4.6 Remaining issues

During the software test, it was discovered that when the BT module is switched on through the MOSFET transistor, the board is apparently reset. Therefore, the firmware restarts and when the BT module is switched on again, the firmware is reset for a second time. This issue happens again and again, causing an infinite loop that does not allow to continue with the firmware execution. This problem could be caused by short breakdown of the supply voltage. When the BT module is turned on, the supply trace suffers a change in its impedance. Moreover, the bypass capacitor of the module must be charged. This may cause the power supply instability and thus the malfunction of the microcontroller.

One attempt to solve the problem was to change the resistor between the MOSFET's gate and the microcontroller pin by another resistor with a larger value. The resistor was substituted by a 1 kΩ (the value was multiplied by ten). Technically, this action will reduce the speed of the transistor response and the instability in the supply voltage trace should be lower. This attempt did not solve the problem, the resets continued to happen. As the problem was switching on the BT module, the code for performing this action was moved to the beginning of the program in order to do it as soon as possible. It worked in the hand soldered board. Nevertheless, when the same firmware was downloaded to one of the reflow boards, the reset problem was still there. So the problem was solved in one case, but not in all assembled devices. At the time of the conclusion of this thesis, it could not be found an explanation to why works in the hand soldered board and no in the other boards. Furthermore, a satisfactory solution to this issue was not found.

5 Conclusion

The results achieved were positive. The PCB design and construction were successfully accomplished. The battery system offers enough autonomy. The firmware and the plugin run as expected. Hence, Lipmouse is a device ready to use.

The Lipmouse project is not finished yet. Although it is a functional device and can be used without any problem, the project is still going on. The problem with the power supply trace and switch on/off of the BT module must be reviewed. In addition, other types of improvements could be incorporated. For example the clicking system of the Mouse plugin cannot be directly connected with the pressure output of the Lipmouse. It must be done through a threshold component. Therefore, the threshold could be integrated inside the Lipmouse plugin. A remodelling of the enclosure and the incorporation of a membrane for the mouthpiece, in order to avoid the accumulation of saliva and the proliferation of bacteria, must be accomplished in a future work.

This project offers the possibility of a computer access to people with physical problems in their upper limbs. The Lipmouse can be combined with other systems, based on the AsTeRICS platform and personalized solutions for these people could be created. It will allow them to be more independent and to enhance their quality of life. In summary, the prospects of the Lipmouse project are very promising.

Bibliography

[1] ATIA, "What is Assistive Technology? How Is It Funded?" http://www.atia.org/i4a/pages/index.cfm?pageid=3859 [Available on 07.05.2014].

[2] ATP, "III. Definitions and legal requirements", 2008, http://www.atp.ne.gov/techassist/def-legal.html [Available on 07.05.2014].

[3] IDEA, "Definition: Assistive technology device", 2004, http://idea.ed.gov/explore/view/p/,root,statute,I,A,602,1, [Available on 07.05.2014].

[4] Infogrip, "Infogrip web page", http://www.infogrip.com/[Available on 08.05.2014].

[5] EnableMart, "Enablemart web page", http://www.enablemart.com/ [Available on 08.05.2014].

[6] AbleData, "AbleData web page", http://www.abledata.com/ [Available on 08.05.2014].

[7] Attainment Company, "Attainment Comapnay web page", http://www.attainmentcompany.com/assistive-technology [Available on 08.05.2014].

[8] Enabling Devices, "Enabling Devices web page", http://enablingdevices.com/catalog[Available on 08.05.2014].

[9] AbleNet, "AbleNet web page", http://www.ablenetinc.com/Assistive-Technology [Available on 08.05.2014].

[10] LIFEtool, "IntegraMouse", http://integramouse.com/index_en.html [Available on 08.05.2014].

[11] Compusult Limited, "Josue 2 and Josue 3", http://www.jouse.com/jouse3/home [Available on 08.05.2014].

[12] Adaptive Computer Control Technologies Inc, "Lipsync", the official web page seems not exist, taken from http://www.abledata.com/abledata.cfm?pageid=113583&top=0&productid=192238&trail=0 [Available on 08.05.2014].

[13] QuadJoy, "QuadJoy 3", http://quadjoy.com/ [Available on 08.05.2014].

[14] TetraLite Products, "TetraMouse Xs and TetraMouse XA", http://tetramouse.com/ [Available on 08.05.2014].

[15] R. Haderer, "Application, extension and evaluation of AsTeRICS", 2014, Master Thesis, University of Applied Science Technikum Wien.

[16] AsTeRICS, "AsTeRICS logo", 2010, http://www.asterics.eu [Available on 15.04.2014].

[17] AsTeRICS, "AsTeRICS User Manual version 2.2", 2013, http://www.asterics.eu/index.php?id=51 [Available on 12.05.2014].

[18] AsTeRICS, "AsTeRICS Developer Manual version 2.2", 2013, http://www.asterics.eu/index.php?id=51 [Available on 12.05.2014].

[19] WinAVR, "WinAVR official web page", http://winavr.sourceforge.net/ [Available on 27.05.2014].

[20] The Eclipse Foundation, "Eclipse project official web page", http://www.eclipse.org/ [Available on 27.05.2014].

[21] CadSoft Computer, "EAGLE offcial web page", http://www.cadsoftusa.com/ [Available on 28.05.2014].

[22] PJRC, “Teensy++ 2.0 vendor”, http://www.pjrc.com/teensy/ [Available on 28.05.2014].

[23] Atmel Corporation, "Key parameters for AT90USB1286", http://www.atmel.com/devices/at90usb1286.aspx?tab=parameters [Available on 28.05.2014].

[24] JNHuaMao Technology Company, “Bluetooth Low Energy devices vendor”, http://www.jnhuamao.cn/bluetooth.asp?ID=1 [Available on 28.05.2014].

[25] JNHuaMao Technology Company, "HM Bluetooth module datasheet", 2014.

[26] ITEAD Intelligent Systems Co.Ltd, "Serial Port BLE Module (Master/Slave) : HM-10", http://imall.iteadstudio.com/im130614001.html [Available on 17.04.2014].

[27] Freescale Semiconductor, "MP3V7007 data sheet", http://www.freescale.com/les/sensors/doc/data_sheet/MP3V7007.pdf [Available on 29.05.2014].

[28] Interlink Electronics, "FSR400 data sheet", http://media.digikey.com/pdf/Data%20Sheets/Interlink%20Electronics.PDF/FSR400_Series.pdf [Available on 29.05.2014].

[29] Everwin Tech Co., "Battery data sheet", https://www.sparkfun.com/datasheets/Batteries/UnionBattery-1000mAh.pdf [Available on 30.05.2014].

[30] Bluetooth SIG, Inc., "Technology Overview", https://developer.bluetooth.org/TechnologyOverview/Pages/Technology-Overview.aspx [Available on 16.04.2014].

[31] N. Gupta, "Inside Bluetooth Low Energy", Artech House, 2013.

[32] IEEE, "IEEE Standard 802.15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specications for Wireless Personal Area Networks (WPANs)", 2002, http://standards.ieee.org/findstds/standard/802.15.1-2002.html [Available on 22.04.2014].

[33] Bluetooth SIG, Inc., "Bluetooth Core Specication, version 4.1", December 2013, https://www.bluetooth.org/en-us/specification/adopted-specifications/ [Available on 16.04.2014].

[34] S. Haykin, “Communication Systems”,4th ed. John Wiley & Sons, 2000.

[35] JNHuaMao Technology Company, "HM BLE USB Dongle datasheet", 2013.

[36] Tobias Hammer, "HTerm version 0.8.1 beta", 2006, http://www.der-hammer.info/terminal/ [Available on 18.04.2014].

[37] Atmel Corporation, "AT90USB1286 manual", 2012, http://www.atmel.com/Images/doc7593.pdf [Available on 22.04.2014].

[38] D. Linden and T. B. Reddy, "Handbook of batteries", 3rd ed. McGraw-Hill, 2002.

[39] R. M. Dell and D. A. J. Rand, "Understanding Batteries", 1st ed. The Royal Society of Chemistry, 2001.

[40] SparkFun Electronics , "Polymer Lithium Ion Battery - 1000mAh, product number PRT-00339", https://www.sparkfun.com/products/339 [Available on 30.04.2014].

[41] S. Dearborn, "Charging Li-ion Batteries for Maximum Run Times", April 2005.

[42] G. Paparrizos, "Fundamentals of battery charging: Part 1", May 2011.

[43] IEEE, “IEEE Standard 1149.1: Standard for Test Access Port and Boundary-Scan Architecture”, 2013, http://standards.ieee.org/findstds/standard/1149.1-2013.html [Available on 30.04.2014].

[44] Maxim Integrated, "Application Note 913: Switch-Mode, Linear, and Pulse Charging Techniques for Li+ Battery in Mobile Phones and PDAs", 2002, http://www.maximintegrated.com/en/app-notes/index.mvp/id/913 [Available on 09.06.2014].

[45] B. Chu, "Application Note 1149, AN1149", Microchip Technology Inc., 2008, http://ww1.microchip.com/downloads/en/AppNotes/01149c.pdf [Available on 06.05.2014].

[46] Maxim Integrated, "MAX8606ETD+ data sheet", Maxim Integrated Products Inc., 2008, http://datasheets.maximintegrated.com/en/ds/MAX8606.pdf [Available on 04.05.2014].

[47] M. Verle, "PIC Microcontrollers - Programming in C", 1st edition mikroElektronika, 2009 http://www.mikroe.com/chapters/view/17/chapter-4-examples/ [Available on 03.06.2014].

[48] European Commission, Directorate General XIII " Telecommunications, Information Market, and Exploitation of Research ", "COST Action 231: Digital Mobile Radio Towards Future Generation Systems : Final Report", 1999, http://www.cost.eu/domains_actions/ict/Actions/231 [Available on 06.06.2014].

List of Figures

List of Tables

List of Abbreviations

ACS	AsTeRICS Configuration Suite
ADC	Analog-Digital Converter
ALU	Aritmetic Logic Unit
AsTeRICS	Assistive Technology Rapid Integration & Construction Set
AT	Assitive Technology
AT (command)	ATtention (Hayes command terminology)
BLE	Bluetooth Low Energy
CAD	Computer Aided Design
CIM	Communication Interface Module
CRC	Cyclic Redundancy Check
CPU	Central Processing Unit
EAGLE	Easily Applicable Graphical Layout Editor
GFSK	Gaussian Frequency Shift Keying
HID	Human Interface Device
IC	Integrated Circuit
ID	Identification
IDE	Integrated Development Environment
IEEE	Institute of Electrical and Electronics Engineers
I/O	Input and Output
IrDA	Infrared Data Association
ISM	Industrial, Scientific and Medical
I-V	Intensity (electrical current) vs Voltage
ISR	Interrupt Service Routine
JTAG	Joint Test Action Group
LED	Light Emitting Diode
LDO	Low Drop Out (regulator)
MAC	Media Access Control
NFC	Near Field Communication
NTC	Negative Temperature Coefficient
OSI	Open Systems Interconnection
PSK	Phase Shift Keying
PC	Personal Computer
PCB	Printed Circuit Board
RISC	Reduced Instruction Set Computer
RF	Radio Frequency
RFID	Radio Frequency Identification
SIG	Special Interest Group
SMD	Surface Mount Device
SPI	Serial Peripheral Interface
TDD	Time Division Duplex
TDMA	Time Division Multiple Access

TWI	Two Wire Interface
USB	Universal Serial Bus
UWB	Ultra Wide Band
WPAN	Wireless Personal Area Network
WWW	World Wide Web

A: Configuring the Bluetooth Dongle

As mentioned, the Bluetooth dongle needs to be configured before being used. It must be set as a master device. The first step is to install a terminal program for opening the BLE-dongle's virtual COM port and sending the commands to this COM port. There are many free terminal programs available. Here, the configuration will be explained using the HTerm program (Reference [36]). This software can be downloaded for free and runs on Windows and Linux platforms.

The first step is to plug the BLE USB dongle into the computer and to open the "Device Manager". After the enumeration, the device will be visible in the COM port section, see Figure A.1. In the example, the port number is COM4. The next task is to open the terminal program.

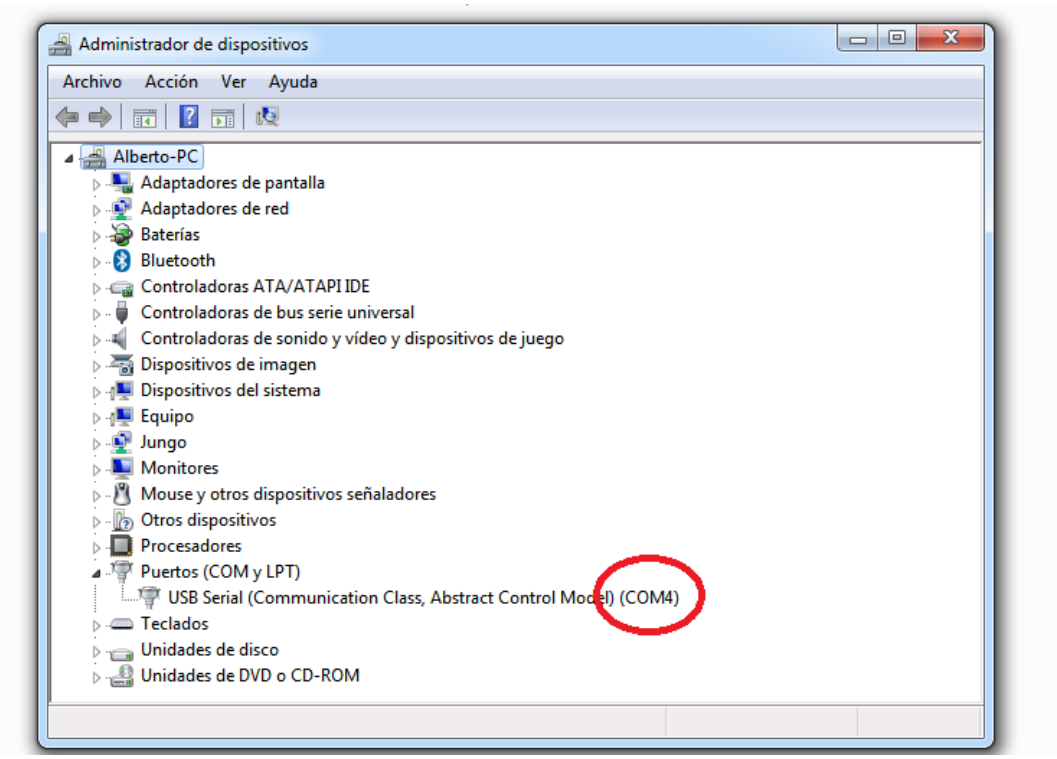


Figure A.1: The dongle COM port

In the case of HTerm, a window similar to Figure A.2 should appear. The first step is to select the corresponding COM port from the list, in this case COM4. If it has not appeared, click on the "R" button in order to refresh the COM ports. After selecting the suitable COM port, click on "Connect".

If there was no problem, the AT commands can be sent now. The first command to be sent must be "AT+RENEW", to delete a previous configuration in the dongle. This command restores the factory default settings. The dongle will respond "OK+RENEW", see Figure A.3. For the change to be effective, the dongle must be powered off and powered on again.

For that purpose, click the "Disconnect" button, unplug the dongle, plug it again and click the "Connect" button again. Now, the command for changing the role to master must be sent, "AT+ROLE1". If the dongle answers with "OK+Set:1", the dongle is configured and ready to be used, see Figure A.4. It will try to pair with the first slave device available.

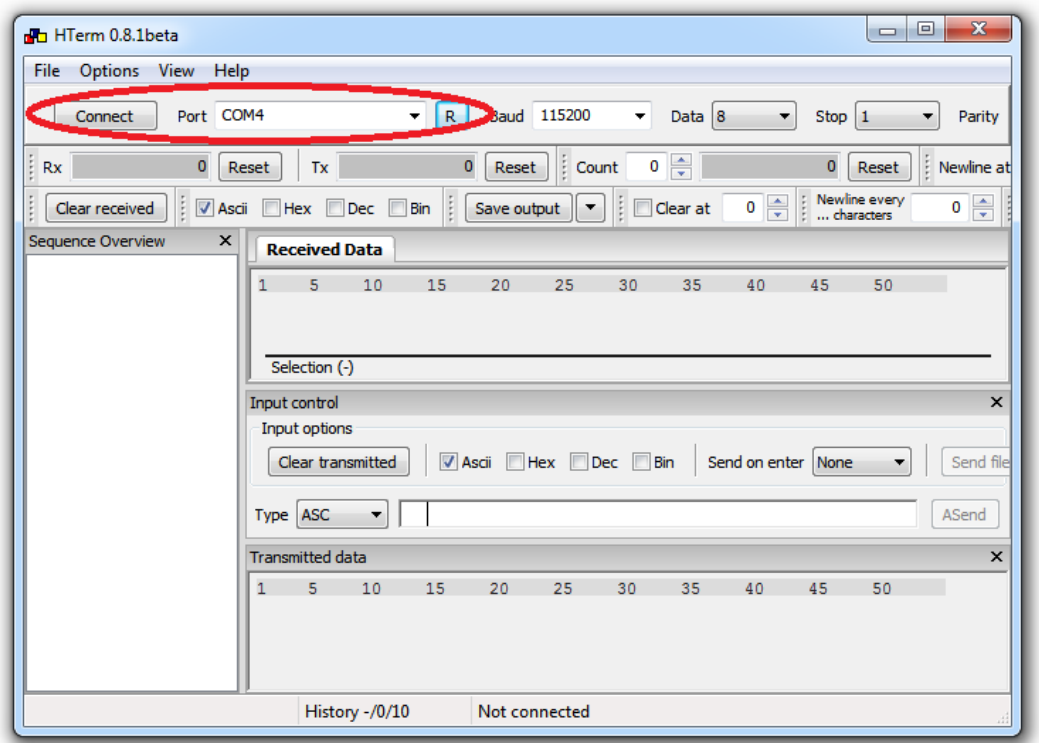


Figure A.2: HTerm

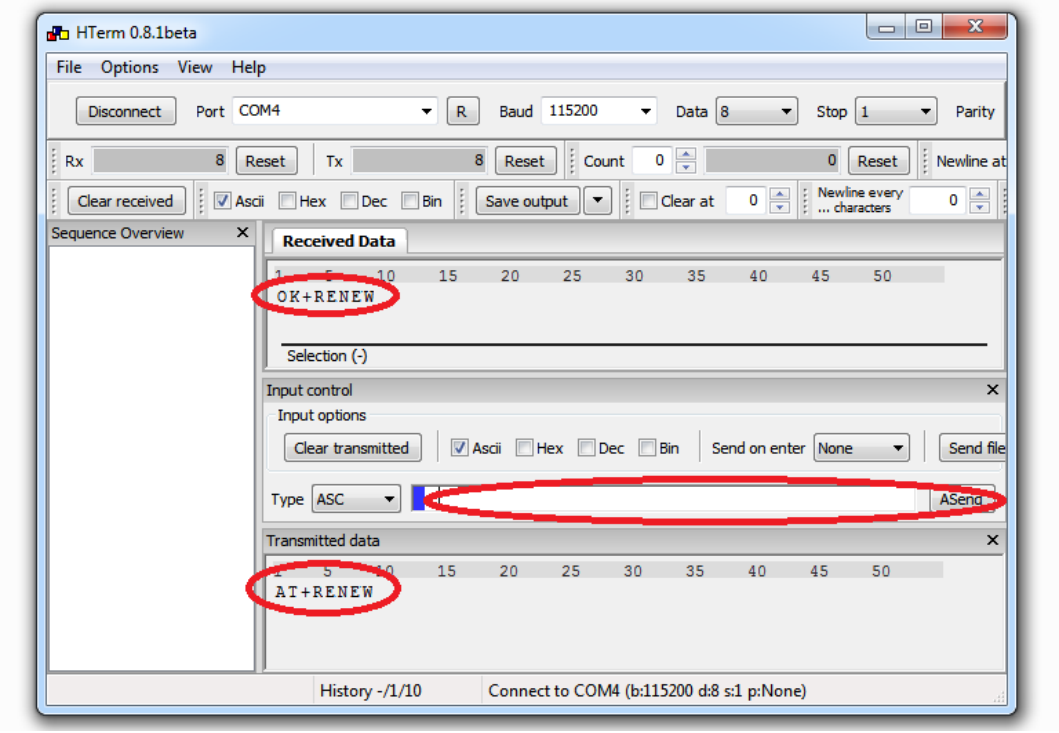


Figure A.3: Restoring the factory default settings of the dongle

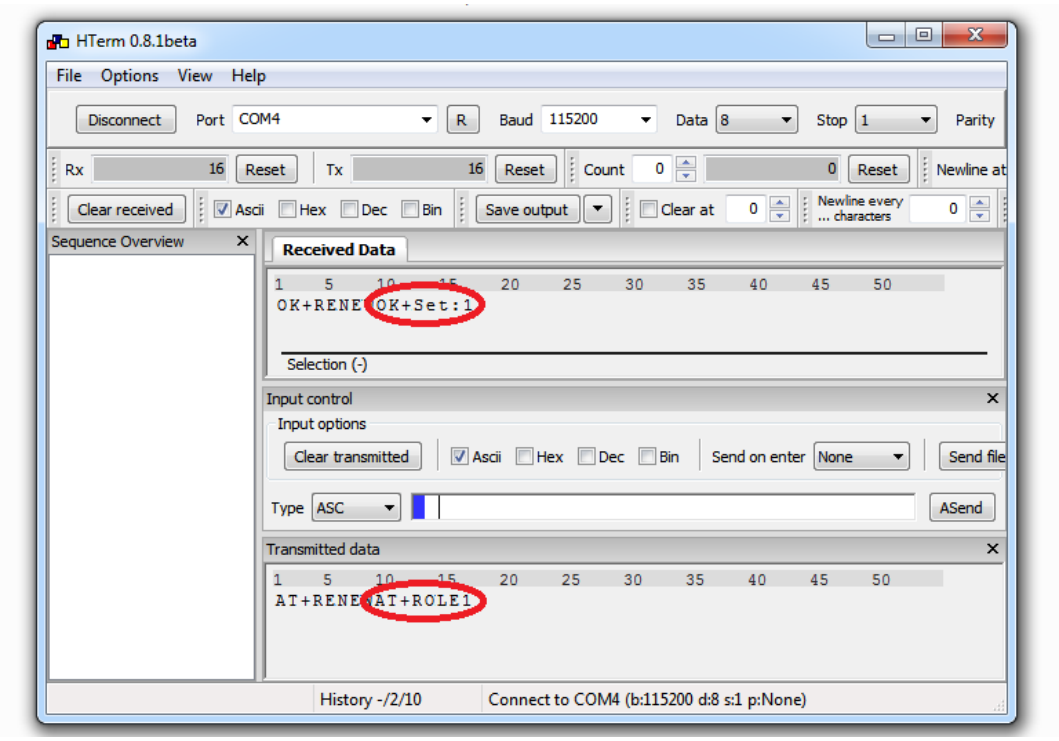


Figure A.4: Setting the dongle as a master device

Note: At the first time of use or after a renew command (AT+RENEW), the dongle will search for any available slave and will try to establish a connection with this device. If the connection is done successfully, the dongle will save its MAC address and, in a future connection, will only search for that device. It will not pair with another device. Therefore, if the dongle has to connect with a different device, it must delete the previous address. An easy way to delete this address is to send a renew command as it was explained in this appendix.

B: Bill of materials

Qty	Value	Device	Package	Parts	Description	Distributor	Dist. nr.
1		SJ	SJ	SJ1	SMD solder JUMPER	–	–
2	0805PS-103KLB	0805PS-103KLB	0805	L1, L2	coilcraft 0805PS-103KLB	Farnell	2286571
1	100k	R-EU_R0603	R0603	R4	RESISTOR, European symbol	Farnell	2331753
5	100n	C-EUC0603	C0603	C2, C3, C4, C13, C15	CAPACITOR, European symbol	Farnell	1759037
1	10n	C-EUC0603	C0603	C1	CAPACITOR, European symbol	Farnell	2320785
1	120R	R-EU_R0603	R0603	R5	RESISTOR, European symbol	Farnell	2331714
1	16MHz	CRYSTALTXC_7A	CRYSTALTXC_7A	CRYSTAL	Various standard crystals	Farnell	1667017
2	18p	C-EUC0603	C0603	C6, C7	CAPACITOR, European symbol	Farnell	1294483
3	1u	C-EUC0603	C0603	C5, C12, C14	CAPACITOR, European symbol	Farnell	2320814
2	22R	R-EU_R0603	R0603	R7, R8	RESISTOR, European symbol	Farnell	2331703
3	330R	R-EU_R0603	R0603	R1, R9, R11	RESISTOR, European symbol	Farnell	2331721
2	4.7u	C-EUC0805	C0805	C8, C10	CAPACITOR, European symbol	Farnell	1759420
1	470R	R-EU_R0603	R0603	R10	RESISTOR, European symbol	Farnell	2331723
2	4k7	R-EU_R0603	R0603	R2, R3	RESISTOR, European symbol	Farnell	2331736
1	AT90USB1286-MU	AT90USB1286-AU	TQFP64	IC1	Atmel AT90USB1286	Farnell	1748495
1	B0530W	B0530W	SOD123FL	D1	B0503W Schottky diode	Farnell	1863142
1	BLE-HM-10	BLE-HM-10	BLE-HM-10	U$1	Bluetooth 4.0 LE module	Iteadstudios	IM130614001
1	FDN360P	FDN360P	SUPERSOT3	Q1	Single P-Channel, PowerTrench (R) MOSFET	Farnell	9845336
1	LG-Y876	LEDSIDELED	OSRAM-SIDELED	LED1	LED green	RS components	708-0700
1	LO-Y876	LEDSIDELED	OSRAM-SIDELED	LED3	LED orange	RS components	654-6108
1	LS-Y876	LEDSIDELED	OSRAM-SIDELED	LED2	LED red	RS components	664-8138
1	MCP1703-3302E/DB	MCP1703-3302E/DB	SOT230P700X160-4N	MCP1703	IC REG LDO 3.3V 250MA	Farnell	1627177
1	MCP73831	MCP73831	SOT23-5	U2	Microchip's MCP73831	Farnell	1332158
1	MICROMATCH-10	MICROMATCH-10	MICROMATCH-10	X2	MicroMaTch 10 B 215464-0 / 1-215464-0	Farnell	3784654
1	MINI-USB-32005-201	MINI-USB-32005-201	32005-201	X1	MINI USB B Conector	Farnell	2112367
1	MP3V7007GP	MP3V7007GP	SMP-MP3V7007	MP3V7007	Pressure sensor	Farnell	-
2	SPST-TACT-KMR2	SPST-TACT-KMR2	KMR2	SW1,SW2	Microminiature SMT Side Actuated Button	Farnell	1437638
1	TSM2306CX	TSM2306CX	SOT23	Q2	30V N-Channel MOSFET	Farnell	1864577

Table B.1: The bill of materials ordered for the PCB