[INFO0064] - Embedded Systems
Remote-Controlled Electronic Lock

April 2014

S. Deffet, T. Galiotto, Q. Garnier, J. Nix
First-year Master's Degree in Engineering

Contents

List of Figures

Part I

Technical Part

Chapter 1

Introduction

Contents

1.1 Statement/Specifications

In the field of home automation, facilities for users and power consumption are of primary importance. Numerous approaches and solutions are feasible but the choices must be carefully made in order to ensure the final product efficiency.

This project takes place in the context of the "Ingénieur de projets" contest. The aim is to fully design a remote-controlled electric lock and its command system. The solution that will be implemented will be as much general as to be applicable in a broad range of home-automation application.

The goal is to open a door without material keys or electronic cards, but with a mobile phone in the most secure and the fastest manner.

By full design, one must understand the practical realisation of the electronic circuit, the low-level control part (e.g. local command of the electronic lock) as well as the top-level control part (e.g. communication with a smartphone application).

1.2 Summary

The first part of this report consists in a preliminary study of the different types of electric locks that we may made use of, as well as the different constraints that are imposed on us in the scope of this contest.

The lock that is used is an electric impulse-driven strike. The system is commanded by a microcontroller. The embedded system interacts with the user's smartphone via its Bluetooth module. The communication between the Bluetooth module and the microcontroller is a UART one.

After, the strike-command program is studied. First the expectations are clearly formalized. Then, cryptography choices are exposed. Finally, we talk about the communication rules that we have implemented.

Subsequently, we study the possibility to offer additional user features. It is decided to implement an adaptive lighting.

After, the electronic system is designed. First, power issues and involved choices are discussed. Then, sensors and other components are chosen according to the constraints.

Once the electronic board is achieved and the electronic components are tested, we implement the command-strike program. Android application issues are also briefly discussed.

Chapter 2

Electric Lock

Contents

Figures

2.1 Definition and Types

An electric lock is a locking device which operates thanks to an electric current.

They are many types of electric locks and those devices can be classified in many ways. Depending on the fact that control systems associated with locks are part of the device or not, we can classify them with respect to the authentication method they use (e.g. RFID) or on the other hand, with respect to their mechanism. Since the authentication system is part of our job, we will only consider the raw class of locks without an authentication system in place.

To solve our problem we will only consider electric strikes. They are generally low voltage devices that use a solenoid to control a movable keeper. Indeed, they are particularly suitable for home appliance and do not prevent the use of a mechanical lockset that opens with a key.

2.2 Overall Architecture

2.2.1 Electric Strikes Categories

There are mainly two kinds of electric strikes. The first kind consists of strikes that keeps unlocked as long as power is supplied. Once the current stop flowing, this kind of strikes locks the door again. The second category comprises strikes that are unlocked thanks to a current pulse and that keeps unlocked (thanks to a mechanical system) until the door is opened by the user.The power required to unlock the strike can also dictate the choice, they come in a variety of voltages between 6 and 24 volts.

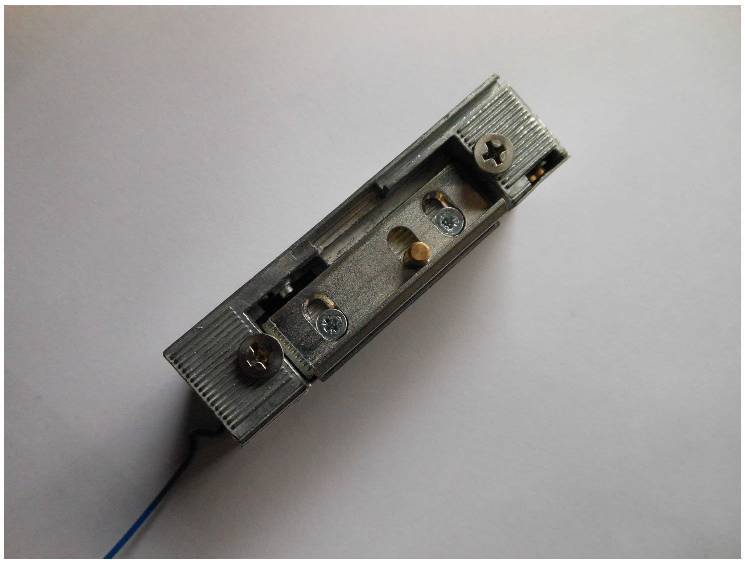


Figure 2.1: Chosen electric strike.

2.3 Choices

Since the device must be battery-powered, our choice naturally moved towards the pulse-driven strikes. The strike we bought can be seen in figure (2.1).

Chapter 3

Access Control

Contents

3.1 Problem Definition

Before we determine the control mechanisms that will be used, we are going to define in detail the expected features.

Main problem:
Open a door with a smartphone.

Constraints:

- The communication must be wireless.
- The user must not have to memorize a password.
- A communication interception as well as the smartphone steal must not give the hacker the capability of unlocking the door.
- The system must be battery-powered.

3.2 Choices

Now that we have exposed the issue, we will determine the methods for solving the problem.

Three approaches have been considered:

1. WI-FI Communication A WI-FI communication could be used to establish a secure channel between the mobile phone and the electric lock. The drawbacks are the cost of WI-FI modules and their power consumption.

2. 3G Communication The use of a 3G communication instead of a Wi-Fi communication suffers from many more drawbacks than WI-FI: delay inherent to this process, need of mobile subscription (that would also make the application unable to work on tablets), and, more generally, the cost.

3. Bluetooth Communication A Bluetooth communication does not suffer from the drawbacks that were mentioned for WI-FI and 3G communications. They are much less expensive and have a lower power consumption. In addition Bluetooth modules are generally easier to use than WI-Fi ones.

So, the approach that we have chosen is the Bluetooth communication. Note that this communication mean is unsecure by nature, so that it must be made secure in software.

3.3 Cryptography

3.3.1 AES Encryption

AES stands for Advanced Encryption Standard. It is a symmetric-key encryption algorithm widely used in practice for its strength but also for its easy-to-implement side.

AES encrypts 128-bit blocks with a key that may be of three different lengths: 128, 192 or 256 bits. For simplicity, we will only consider 128-bit keys The number of rounds needed to encrypt a 16-byte block with a 16-byte key is 10. As the decryption use the same techniques and is the invert of the encryption, only encryption will be explained.

The encryption flowchart in Fig. (3.1) can be divided in four basic subdivisions:

- The first phase is the KeyExpansion. Here, we derived round keys from the cipher key using Rijndael's key schedule (We wont enter into more detail for this as this is not the main purpose of the project/course). AES requires a separate 128-bit round key block for each round plus one more.
- Then, we enter into the initial round where Key Addition is performed (see below).
- Now, we are in the loop where four steps are required:

  S-Table Substitution Here, the bytes of the current matrix (e.g. state) are substitute with another thanks to a 8 bits lookup table. This lookup table names Rijndael S-box (S is for substitution) is particular because it is derived to have non-linearity properties.

  Encode Row Shift The Encode Row Shift step operates on the rows of the current matrix ; it cyclically shifts the bytes in each row by a certain offset. For AES, the first row is left unchanged. Each byte of the second row is shifted one to the left. Similarly, the third and fourth rows are shifted by offsets of two and three respectively. For blocks of sizes 128 bits and 192 bits, the shifting pattern is the same. Row n is shifted left circular

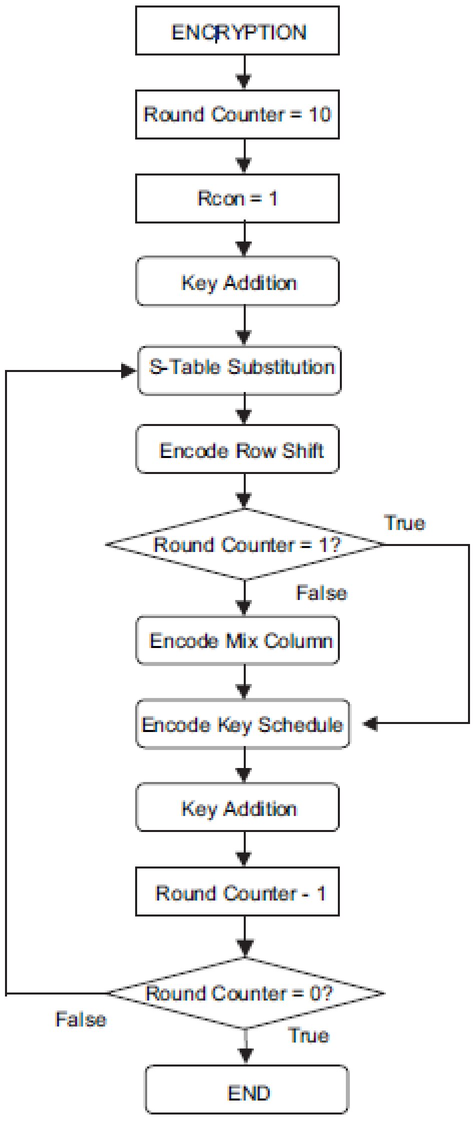


Figure 3.1: AES encryption flowchart

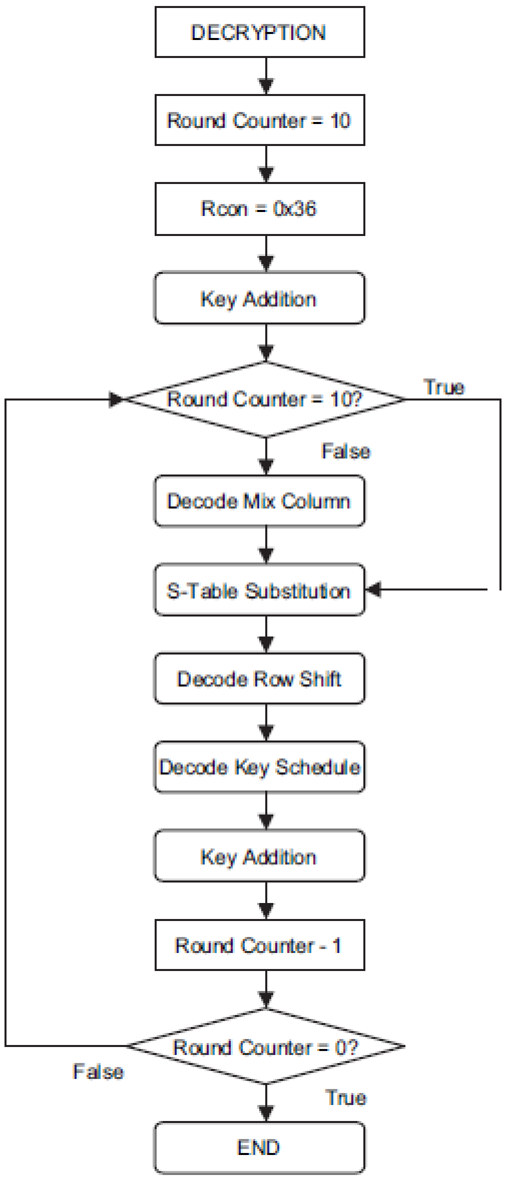


Figure 3.2: AES decryption flowchart.

by n-1 bytes. In this way, each column of the output state of this step is composed of bytes from each column of the input state. The importance of this step is to avoid the columns being linearly independent, in which case, AES degenerates into four independent block ciphers.

Encode Mix Column During this step, the four bytes of each column of the current matrix are combined using an invertible linear transformation. The Mix Column function takes four bytes as input and outputs four bytes, where all four output bytes are affected by each input. Together with Row Shift, Mix Column provides diffusion in the cipher. Moreover, each column of the current matrix will be multiplied with a fixed polynomial which is derived from a fixed matrix.

key Addition In the key Addition step, the round key is combined with the current matrix. For each round, a key is derived from the main key using Rijndael's key schedule; each derived key is the same size as the state. The derived key is added by XORING each byte of the state with the corresponding byte of the derived key.

- Finally, the last phase (and also round) where we only three of the four steps are used:S-Table Substitution, Encode Row Shift,key Addition. Encode Mix Column is not used because, this way it permits to do the decryption in a simpler way (e.g. in a inverted way)

3.3.2 Authentication Control

The main problem we have been dealing with was that we had to ensure that only authorized people will be able to open the door. At this point, there is absolutely no need to encrypt the data. Actually, a key (chosen by the user) can be shared between the user and the lock so that only people who have that key will be able to open the door. We call such a key a "private key".

Note that at first sight, it may be important to give the user the possibility to change the key value.

3.3.3 Protection Against Interceptions

In addition to authentication control, we had to ensure that someone, close to the door and hidden, could not open the door by simply intercepting messages between the smartphone and the lock, and sending them back to open the door. This problem may be solved by taking care that for the same request (e.g. "Open the door"), the data sent will never be the same. Of course, this requires data encryption.

To ensure that the AES encryption of a given request will lead to different data each time it is made, the key used to encode the data must change in time. Such an encryption key is called a "random key".

Practically, when the user wants to send a request to the lock, it first connects to the device. Then, the device generates a random key. This random key is encrypted with the private key and then the encrypted data are sent to the user. The user decrypts these data and so he gets the random key. Only the users that have the private key will be able to decode the random key so that a man-in-the-middle will not be able to decrypt it. The user sends the request (e.g.

"Open the door") encrypted with the random key. Each time a request is made, this process happens, so that a same message is never encrypted to lead to the same data.

Note that the random key does absolutely not have to be a perfectly random key, it must just be different each time it must be sent. Hence, a pseudo-random key will be perfectly suitable. Note also, as it has been previously mentioned, that encryption keys must be 16-byte long.

3.4 Communication Rules

Recall that messages are of three types: messages that don't need to be encrypted (e.g. the connection request), messages that are encrypted with the private key (e.g. the random key) and messages that are encrypted with the random key (e.g. an open request). Taking into account these cryptography issues, we have established simple communication rules that are listed in appendix (C).

First of all, only critical operations such as an open request or a key-modification request will be encrypted with a random key. Secondly, a random key cannot be used twice and must be encrypted with the private key. Finally, non-critical messages such as a random-key request or an error message will not be encrypted.

Note that to facilitate implementation, it has been decided that each message to be sent will be 16-byte long. This allows to use fixed-size buffers and we also get rid of the padding need, for AES encryption.

Chapter 4

Other User-intended Features

Contents

4.1 Problem Definition

The embedded system may be used to offer a wide range of user-intended features that are of interest because of its location (next to the door) and its communication facility with smartphones. One can think, for instance, about access monitoring, embedded weather station or smart lighting.

We have chosen to add an adaptive lighting system to our application.

Main problem: Illuminate the door.

Constraints:

- The door must be illuminated when a presence is detected.
- The luminosity must be sufficient and adapt to ambient light. In particular, it must not illuminate during daytime and must light at night.
- The system must be battery-powered.
- The user must have the ability to turn off the lighting (permanently).

4.2 Choices

The approach that we will follow is the use of a light sensor to measure the luminosity level. This level will be used to control in software the light intensity to produce.

Now we must choose the kind of lighting we will use. The battery requirement brings us to choose LEDs. Two approaches have been considered. The first one was the use of power LEDs and the second one was the use of high-brightness LEDs. We decided to use high-brightness

LEDs. In particular we have chosen to use the C503D-WAN-CCBDB232 LEDs from CREE®. These LEDs are specified to be particularly suitable for torches and thus matches our power-consumption requirement.

Finally we have chosen to supply the LEDs lighting with a dedicated battery. It particularly makes sens when considering that the lighting features is an option and also when considering power constraints on the embedded system.

4.3 Adaptive Lighting

The algorithm will be very simple. The lighting will be proportional to the ambient darkness. In particular it will be maximal at night and non-existent during daytime. In strong shadow, it will have an intermediate level.

Chapter 5

Electronic Design

Contents

5.1 Problem Definition

Before discussing the practical realisation, it is worth recalling the different expected features.

The electronic board must be able to command a electric strike. The strike to command can be driven with DC current as well as with AC current. If DC current is used, the voltage to apply must be comprised between 6V and 12 V. The nominal current is 240 mA. The maximum current that the strike may undergo is 400 mA.

The communication between the board and the smartphone is a Bluetooth one.

The board as well as the strike must be powered with batteries. Consequently, the batteries level must be monitored and the user must be warned in case of low-battery.

The electronic design must also be robust with respect to board misuses by the user such as polarity inversion of the batteries and brutal interruption of the power supply.

5.2 Power

5.2.1 Batteries

We could discuss the choice of the batteries after having chosen the other components or we can choose the batteries and then design the rest of the circuit under this constraint. We decided to first choose the kind and number of batteries that will be used, mainly because the batteries will have to be replaced by the user, so we can not choose a possibly outlandish kind of battery that will suit us once the circuit will have been designed.

We limited the range of cells in which to choose to three of the most commonly used ones: 9V Alkaline, 1.5V AA Alkaline and 1.5V C Alkaline. To make our choices we looked at the Energizer®product datasheets of these products (www.energizer.com). 9V cells have been immediately eliminated from the list because of their poor capacities (in mAh) with respect to 1.5V cells. Finally, it has been decided that four AA 1.5V cells will be used.

5.2.2 DC/DC Converter

Since, the maximum power supply voltage is 6V, it must be raised to at least 6V (maximum 12V) in order to operate the strike. This is achieved by the use of a boost converter. This must carry a current of around 240 mA.

We choose the MEJ2S0509SC DC/DC converter whose characteristics are listed below:

- Output Power: 2 W
- Input Voltage Range: 4.5 V to 5.5 V
- Input Voltage-nominal: 5 V
- Output Voltage-Channel 1: 9 V
- Output Current-Channel 1: 222 mA

5.2.3 Design

MOSFETs

MOSFETs are used to switch on and off various circuit components.

In our circuit, classical N-channel MOSFETs are suitable for all the components that must be commanded. We will use BS170 MOSFETs from Fairchild Semiconductor ®. The main specifications are listed below:

- Transistor Polarity: N-Channel
- Drain-Source Breakdown Voltage: 60 V
- Gate-Source Breakdown Voltage: +/- 20 V
- Continuous Drain Current: 0.5 A
- Rds On: 1.2 Ohms
- Forward Transconductance - Min: 0.32 S

- Power Dissipation: 0.83 W

This choice of MOSFET is particularly suitable for our application since it can be seen in the datasheet that the maximum allowable current with a gate-to-source voltage of around 4V (typically the output voltage of a pin of the microcontroller) and a drain-to-source lower than 6V (worst case in the scope of our application) is around 350 mA.

Freewheeling Diode

As the electric strike is an inductive load, it may generate a voltage peak when it is shut down. Such a voltage peak may be dangerous for the MOSFET.

By its nature, a MOSFET already has a diode between its source and its drain. This diode might be sufficient. Nevertheless, for safety reasons, we have adjoined to the MOSFET commanding the strike an external diode that is much faster.

We chose the UG06B from Vishay ®mainly for its low reverse current ($5\mu A$) and its short recovery time ($25ns$).

5.3 Logic

The logic part must be capable of doing a certain amount of tasks:

- Perform A/D conversions to determine the battery level and the ambient light;
- Perform AES encryption as well as a PID regulation;
- Communicate via UART with the Bluetooth module ;
- Manage an emergency signal when the power supply is removed;
- Generate logical output value to drive the MOSFETs and some LEDs.
- Generate a PWM signal;
- Consume very little power while not active.

5.3.1 Microcontroller

To achieve all these tasks, we choose a PIC18 microcontroller from Microchip ®, the PIC18F4620.

The features of this controller meets all our expectations. For instance,

- Data EEPROM (bytes): 1024
- Digital Communication Peripherals: 1-UART, 1-A/E/USART, 1-SPI, 1-I2C1-MSSP(SPI/I2C)
- Capture/Compare/PWM Peripherals: 1 CCP, 1 ECCP
- Timers: 1 x 8-bit, 3 x 16-bit
- ADC: 13 ch, 10-bit
- Interrupts among which pin-triggered interrupts.
- Sleep mode with current down to 100 nA typical

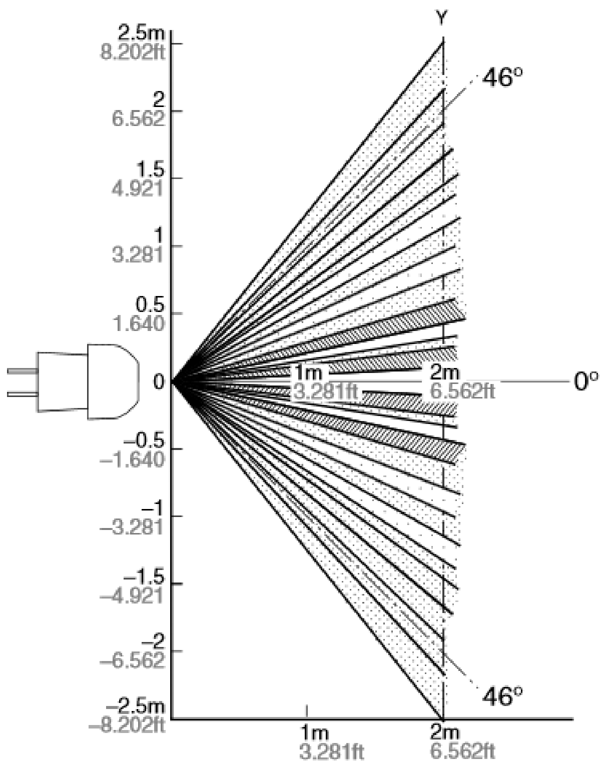


Figure 5.1: IR sensor detection area.

5.3.2 Sensors

Infrared sensor

The aim of this sensor is to detect a presence in order to save energy (e.g. by powering off the Bluetooth module). Therefore, it must have very little power consumption.

The sensor we use is the AMN32112 from Panasonic. It is a passive sensor and consumes around $50\mu A$ (3-6 V). It has a digital output and a 2-meter detection range which is perfectly suitable in the scope of this application, as can be seen in Fig. (5.1).

Photodiode

We decided to use a photodiode. A (reverse-biased) photodiode gives a current that is function of the light intensity. Hence, a operational amplifier is necessary to convert this current into a voltage that will be admissible for the microcontroller.

We decided to use the TSL257SM-LF from ams®that has a built-in operational amplifier.

5.3.3 Power Supply and Voltage Regulator

A standard LM7805 cannot be used. Indeed, it dissipates too much power even if the output current is relatively low. Hence, we decided to use a low-dropout (LDO) regulator. An LDO regulator is a voltage regulator that works with low input/output voltage differentials. This enables minimizing power dissipation [3].

We choose the MCP1700-4002E/TO from Microchip whose main characteristics are listed below:

- Input Voltage MAX: 6.5 V
- Output Voltage: 4 V
- Dropout Voltage - Max: 350 mV
- Output Current: 250 mA
- Load Regulation: 1 %

This means that the battery level can fall from 6V down to 4.35V (e.g. down to 1.0875V per cell) before being unusable.

To measure the cells level we will use a simple voltage divider as an entry of one of the microcontroller A/D converters. Since, the maximum recommended impedance for analog sources is 2.5 k (cf. PIC18F4620 datasheet), it will consume relatively much power and hence should be powered off when the microcontroller is in sleep mode. A MOSFET is thus needed. Actually, the voltage divider and the Bluetooth module may share the same MOSFET as they both need to be powered on and off at the same time.

5.3.4 Bluetooth Module

The Bluetooth module that we will use is the TEL0026 from DFRobot. It provides TTL level UART interface which is supported by our microcontroller. Its other main characteristics are:

- Voltage: 3.5 to 8 V
- Current: 50 mA
- Transmission power: +4 dBm max
- On-chip antenna

5.3.5 Schematics

The schematics corresponding to our circuit can be seen in Fig. (5.2).

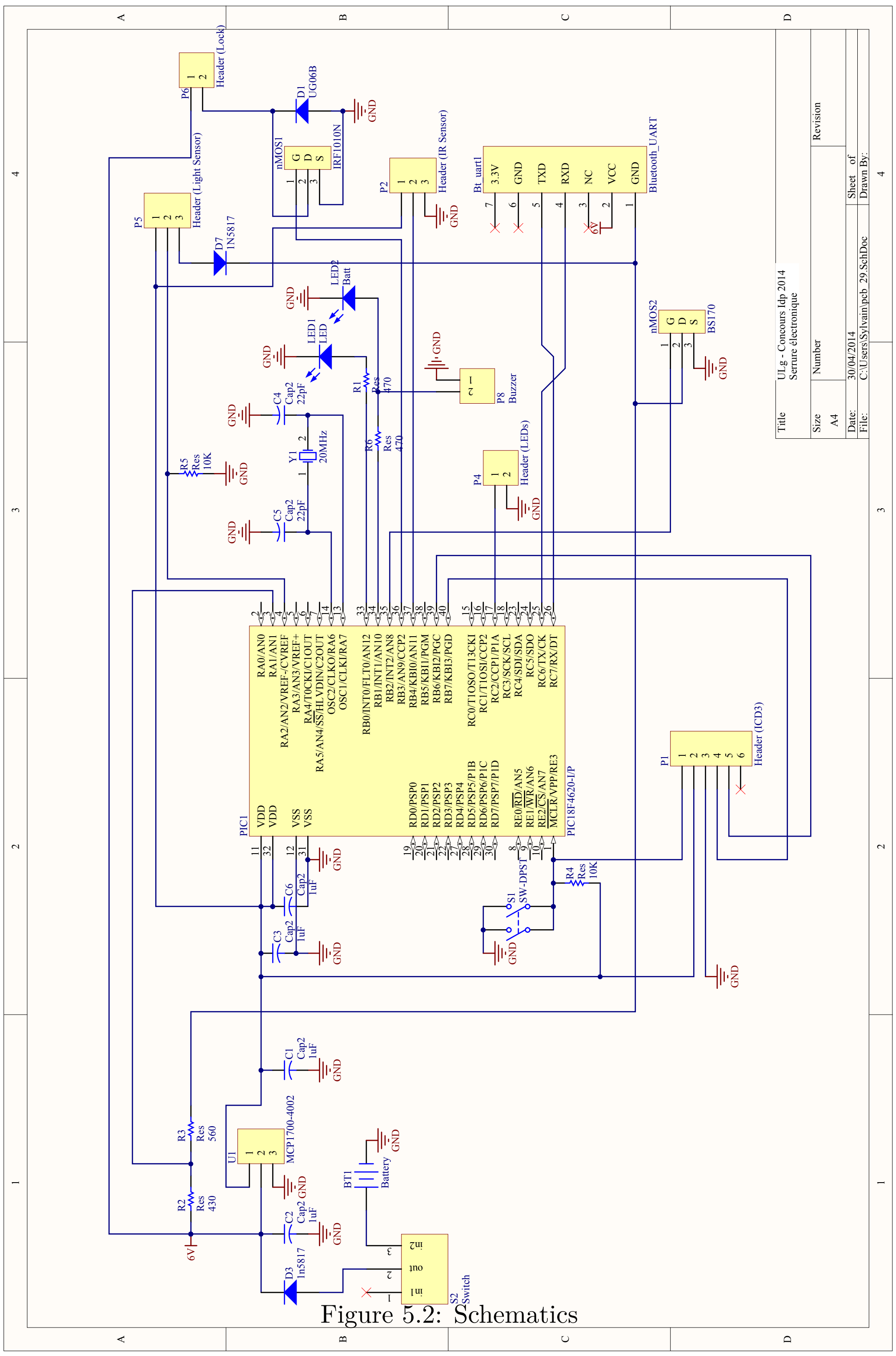


Figure 5.2: Schematics

Chapter 6

Software Aspects

Contents

6.1 Microcontroller

6.1.1 Problem Definition

Now that the board was designed, we must write the code that will be run in the microcontroller.

It must meet several expectations:

- Command the electric strike;
- Encrypt and decrypt data with the AES algorithm;
- Generate pseudo-random numbers;
- Test the level of the battery;
- Enter sleep mode after a delay when the IR sensor does not detect any presence anymore;
- Wake up when the IR sensor detects a presence;

- Communicate with the Bluetooth module via USART;
- Modulate a LED lighting with respect of the luminosity given by an analog sensor;

The program has been directly implemented in assembly language as the execution time is of prior importance.

6.1.2 Strike-command Program Implementation

Constraints

When writing this program, we faced several constraints.

In order that the private key remains in memory when no power is supplied, it must be written in EEPROM (Electrically-Erasable Programmable Read-Only Memory).

Another constraint is that the level of the battery must be monitored when a presence is detected (it is not worth the trouble to monitor this level if there is nobody to be aware of it). Nevertheless, it may not be so wise to test the battery only once, when the microcontroller wakes up, as it may remain awake for a long time. In addition, the output level of the battery may vary with the load applied so that more than one conversion may be necessary to accurately judge the battery level. Consequently, the battery testing will take place in the main loop of the program.

Since a request, such as an open request, may occur at any given time once the microcontroller is awake, it would be a pity to block the program until a request is received. To solve this problem, a request sent to the microcontroller will trigger an interrupt that will handle the reception. Once the reception is completed, a flag will be raised. Note that we must be particularly vigilant in the main loop when processing some buffers that may be modified by an interruption.

An external interrupt (a positive logic level generated by the IR sensor) must awake the microcontroller. On the other hand the microcontroller must enter sleep mode after no presence has been detected for a while. This is done by setting a counter. Each time a presence is detected by the IR sensor, the counter is reset and starts being incremented. When the counter value reaches a certain threshold (corresponding to a certain amount of time with no presence detected), the microcontroller enters sleep mode.

AES Encryption

Microchip ®provides data encryption routines for the PIC18 including an AES encryption routine. Their implementation of AES uses a 16-byte block and a 16-byte key, which yields to 10 rounds of encryption.

Overview of Routines

AESEncrypt:

- Function: void AESEncrypt(BYTE* key, BYTE* block)
- PreCondition: key and block variables loaded with key and data to encrypt
- Input: key[16] encryption key, block[16] data to encrypt
- Output: none

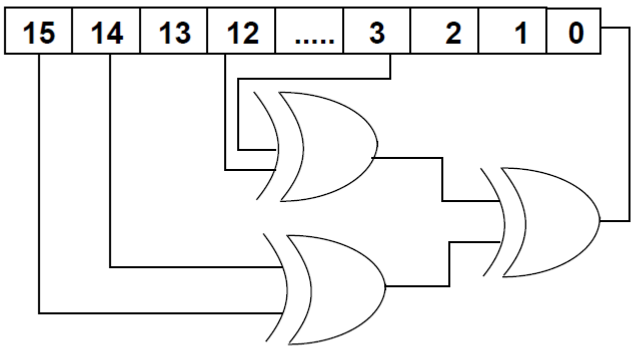


Figure 6.1: Pseudo-random number generator.

- Side Effects: block is encrypted and key has changed to the decrypt key. Called subroutines may use the table pointer
- Stack Requirements: 3 level deep
- Overview: block is encrypted and key has changed to the decrypt key

AESDecrypt:

- Function: void AESDecrypt(BYTE* key, BYTE* block)
- PreCondition:
- Input: key[16] key array, block[16] data array
- Output: none
- Side Effects: block has been decrypted and key should be at original encrypt key
- Stack Requirements: 3 level deep
- Overview: block has been decrypted and key should be at original encrypt key

Pseudo-Random Numbers Generation
Microchip ®provides math utility routines with a pseudo-random number generator for its second generation of high performance 8-bit microcontroller, the PIC17C42. This routine generate a 16-bit pseudo-random number by left shifting the content of the 16-bit register with the LSB set as shown in Fig. (6.1).

This routine was used to generates the random key. The macro performing a 16-bit pseudo-random number generation can be seen in Fig. (6.2).

6.1.3 Communication

The detailed protocol that we have established may be found in the Appendix.

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) is used to communicate with the Bluetooth module. The mode is asynchronous. The baudrate was arbitrarily fixed to 9600 bits/s.

```
rand16 macro BHi
    rlcf BHi, w
    xorwf BHi, w
    rlcf WREG, 1

    swapf BHi, 1
    swapf BHi+1, w
    rlncf WREG, 1
    xorwf BHi, w
    swapf BHi, 1
    andlw 0x01
    rlcf BHi+1, 1
    xorwf BHi+1, 1
    rlcf BHi, 1
```

Figure 6.2: Macro performing a 16-bit pseudo-random number generation.

As discussed in section (6.1.2), reception is handled via an interrupt. On the contrary, sending does not occur in an interrupt.

6.1.4 Power Management

Cells-Level Monitoring

A LED (as well as a buzzer) placed on the electronic board allows the user to monitor the state of the battery. When the battery level drops below a certain threshold, the LED starts blinking and the buzzer starts beeping (requiring the use of a timer). In addition, the battery level may be send to the user's smartphone when it establishes a connection with the embedded system.

The use of an A/D conversion has already been discussed in section (5.3.3).

Infrared Sensor and Sleep Mode

The infrared sensor has a logical output.

Suppose first that the microcontroller is in run mode, executing the main loop. The aim is to maintain the microcontroller in run mode as long as a presence is detected and also during a certain delay after any presence is not detected anymore (typically a few seconds). This is managed thanks to the use of a counter, as already said.The reason why the presence detection is not managed via an interrupt is that it may occur by far too often.

Suppose now that the microcontroller is in sleep mode. It must be woken up by the presence of logical high value produced by the IR sensor. An exit from sleep mode may be triggered by any of the available interrupt sources.Thus, we use a PORTB interrupt-on-change. This mechanism triggers an interrupt when an input changes on PORTB<7:4>.

6.1.5 Adaptive lighting

To modulate the light intensity, we generate a PWM signal. The signal duty cycle is a function of the ambient light intensity that is determined thanks to an A/D conversion. In particular, it is set to zero when the ambient light is high and it is set to one in darkness.

6.2 Android Application

6.2.1 Problem Definition

The application must be able to interact with the electric lock, via Bluetooth. It must have a user interface allowing the user to open the door as well as to set the private key and eventually change it.

6.2.2 Key Storage

The private key must be stored in a persistent manner (so that it is not lost when the application is closed).

Android provides several options to save persistent application data:

- Shared Preferences: store private primitive data in key-value pairs.
- Internal Storage: store private data on the device memory.
- External Storage: store public data on the shared external storage.
- SQLite Databases: store structured data in a private database.
- Network Connection: store data on the web with your own network server.

Internal storage was chosen. Files saved to the internal storage are set as private so that other applications cannot access them (nor can the user). If the user uninstalls our application, these files are removed.

6.2.3 Cryptography

As AES is implemented in java (e.g. for android), we used the appropriate library. To encrypt a message, we simply have the encryption 128 bits key and divided the message in a 128 bits block (in our case, we only use messages of 128 bits). As we use full block, we do not have to pad our blocks. Finally, we have the decrypt function to decrypt a cipher with a key (the key has to be the same as for encryption in our case).

6.2.4 Bluetooth Communication

Discovering Devices

The electric lock is supposed to have already been paired to the device (through the Set Up menu). In this way, we avoid performing a device discovery which is a very slow process. Hence, the first step of a Bluetooth communication consists of querying paired devices. The user is proposed the list of paired devices ans selects one of these to which to send the request (e.g. an open request).

Connecting to the Paired Device

Then the phone must connect to the device. At this point, we must chose whether we want to connect as a server or as a client. We want, of course, to connect as a client.

The connection is initiated and managed by the use of a Bluetooth socket.

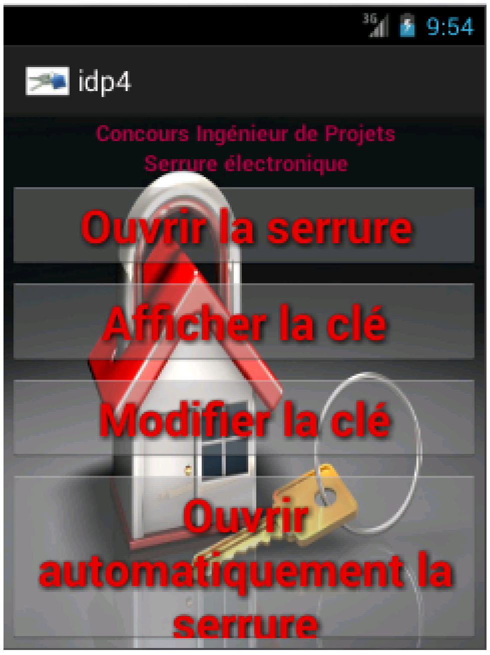


Figure 6.3: Android app: main menu.

Sending and Receiving Data
Now that the phone is connected to the embedded system we can send data or wait for incoming data.

The methods performing these operations are blocking methods. However, we don't want to block the user's phone until these actions have been performed. In particular, we want the user to have the possibility to cancel the sending of a request (this is done by closing the Bluetooth socket). Consequently, the sending and receiving methods will be called in a new thread. Nevertheless, this creates a new problem as the Bluetooth socket is used by both threads (in the main thread to give the user the possibility to cancel the communication and in the new thread to perform sending and receiving operations). Hopefully, the Bluetooth socket is a thread-safe object that may be used in many threads without generating conflicts.

6.2.5 User Interface

The main menu is shown in Fig. (6.3). The user interfaces may be seen in details in Appendix (B).

6.2.6 Service

A Service is used to perform a longer-running operation while not interacting with the user. The service performs its activity even if the application is closed (actually the application process keeps running but the application is not displayed on the screen and appears to the user as it was closed).

In this project we implemented a service that constantly try to open the door so that when the user approaches the door he doesn't have to take his phone and open the app to unlock his door. Of course, the service is launched at the demand of the user and may be stopped whenever he wants.

Note that services, like other application objects, run in the main thread of their hosting process so that Bluetooth operations must be performed in another thread as discussed in section (6.2.4).

6.3 Codes

The microcontroller program and the Android applications consist of several thousands of lines of code. Consequently they cannot be placed in the Appendix. They are joined to this report in the archive.

Chapter 7

Results and Encountered Problems

Contents

7.1 Strike Command

In practice, the MOSFET that we had chosen was not able to pass enough current with a gate-to-source voltage of 4V. We solved the problem using an IRF1010N instead of the initially-determined IRF1310N.

The boost converter appeared to be useless and so was not used.

The strike-command program works as expected.

7.2 Power Management

From our point of view, the IR sensor is a little bit too sensitive. It also has a longer range than the one specified in the datasheet.

However, it works well and the microcontroller enters sleep mode when no motions is to be detected and is woken up when somebody passes in front of the IR sensor.

7.3 Adaptive Light

The light sensor appeared to be perfectly adequate in the scope of our application, having maximal sensibility when strong darkness settles.

7.4 Product Demonstration

In order to demonstrate our product, we have built a small door equipped an electric strike connected to our embedded system. This door, during one of its building step, may be seen in Fig. (7.2).

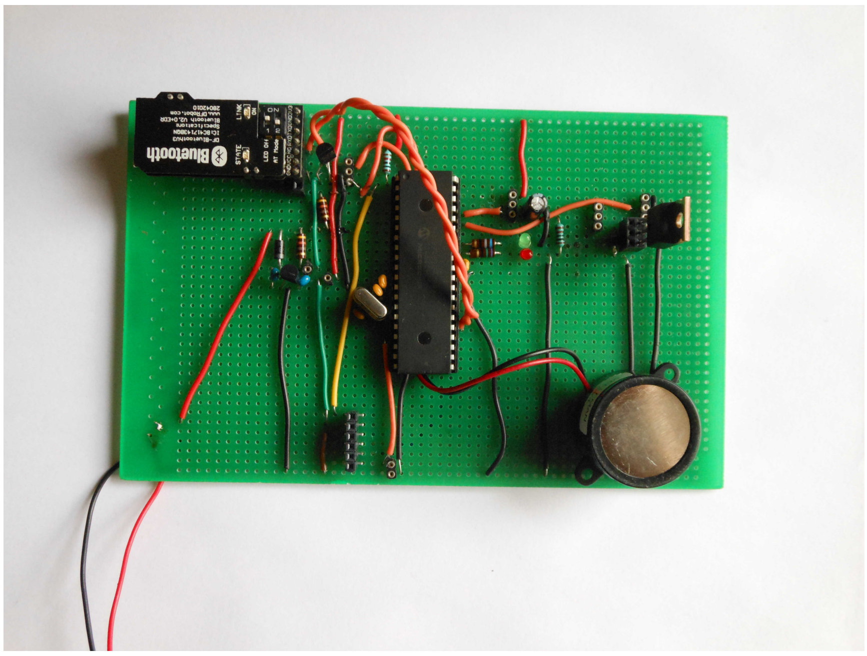


Figure 7.1: First prototype.

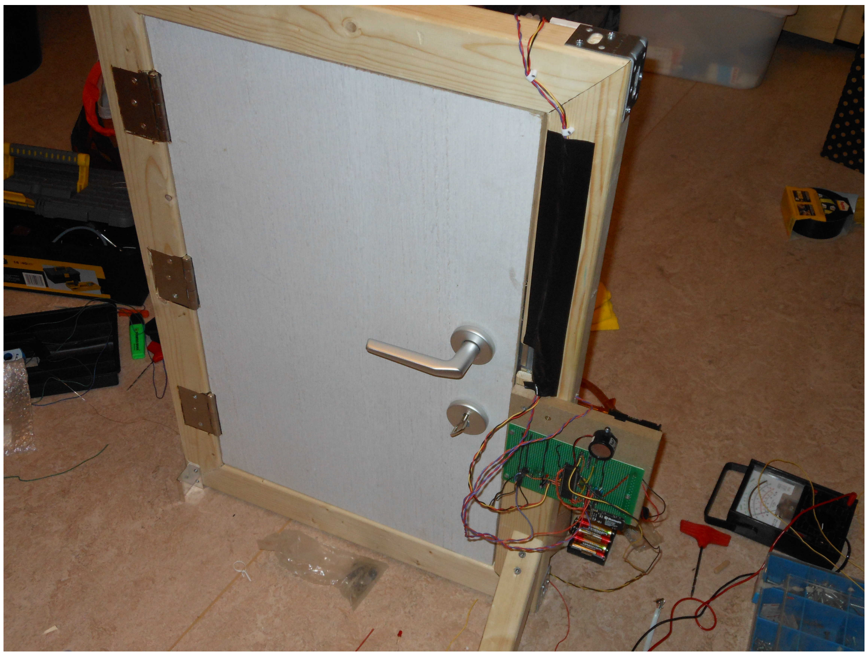


Figure 7.2: Small door for the product demonstration.

Chapter 8

Conclusions

Contents

8.1 *Ingénieur de Projets* Contest

This work takes place under the *Ingénieur de Projets* contest.

It is a multidisciplinary projects (mixing data processing and electronics) performed in a small group of four students from various sections.

Information about this project in particular may be found on the following website: http://ingenieurprojets.be/downloads/descriptions/2014-serrure.pdf.

8.2 Results

During this project, we fully conceived the electronic system of a remote-controlled lock.

The electric lock consists of an electric strike connected to an embedded system that is able to communicate with a smartphone via Bluetooth.

In order to ensure security we established communication rules involving the use of cryptography. This rules mainly lie on the use of a private key as well as a pseudo-random generated key with limited life-time. The AES algorithm was used to encrypt data.

Power consumption has been a major concern throughout this work. The use of an IR sensor to enter sleep mode was one of the solution that we implemented in order to minimize power consumption.

In addition, a extra user intended feature, consisting of an adaptive lighting, has been implemented.

The whole system was tested with satisfaction. Effort was also made to give the ability to easily add extra features in the future.

Part II

Administrative Part and Team Work

Chapter 9

Team Work

The work was portioned according to members sections. These are resumed in table (9.1). J. Nix was in charge of the cryptography part. T. Galiotto played an important role during the choices of the components and the circuit design. S. Deffet and Q. Garnier mainly implemented the microcontroller program. Finally, the Android application was achieved by S. Deffet et J. Nix.

Despite this task partitioning, the project really took place as a team work, as no task were really separated from the others.

Member	Section
Sylvain Deffet	First-year Master's Degree in Electrical Engineering
Thomas Galiotto	First-year Master's Degree in Electrical Engineering
Quentin Garnier	First-year Master's Degree in Electrical Engineering
Julien Nix	First-year Master's Degree in Computer Engineering

Table 9.1: Members and respective sections.

Chapter 10

Calendar

To achieve this project we had determined a detailed calendar at the beginning of the work. This calendar can be seen in table (10.1). All the deadlines were pretty much observed.

Action	Datum Limit	Distribution
Components Ordering Android app specifications	01/02/2014	/ Everybody
Android app Bluetooth part	15/02/2014	2-4 members
Embedded Bluetooth	01/03/2014	2-3 members
IR sensor use	15/03/2014	2-3 members except Julien
Cryptography Light sensor + LED lighting	01/04/2014	Julien 2-3 members except Julien
Power consumption reduction	07/04/2014	2-4 members
PCB design or final board	15/04/2014	/
Finish Android app	01/05/2014	2 members
Adaptation to a hotel cas	August 2014 (More detailed plan must be established in July)	Everybody
Improvements	November 2014	Everybody

Table 10.1: Calendar.

Chapter 11

Budget

The electronic components were mainly purchased from Farnell, Mouser and Gotronic. The detailed ordering may be see in Appendix (D).

The electric strike as well as the door were ordered in a carpentry shop.

Until now, around half of the budget was used.

Appendix A

Use cases

- Actors: user & electronic lock
- Aim: authentication
- overview: User
  - opens the app and connects itself with Bluetooth
  - enters a unique code given with the lock
  - is authenticate
  - closes the app
- Actors: user & electronic lock
- Aim: open the door
- overview: user
  - approaches near from the door
  - the infrared captor detects the user and the Bluetooth module is on
  - *optional see section* the user opens the application
  - opens the lock (also the door)
  - closes the app
- Actors: user & electronic lock
- Aim: prevent opening with a stolen smart phone
- overview: user
  - has lost his smart phone (he still can open the door with a mechanical key)
  - can use another smart phone, install the app
  - changes the unique code in another
  - uninstall the application if it is not his smart phone or simply close it
  - closes the app
- Actors: user & electronic lock

- Aim: replace the batteries
- overview: user
  - the system alerts the user that the batteries are low
  - opens the box containing the system
  - replaces old batteries with new one
  - closes the box

Appendix B

Android Application: User Interfaces

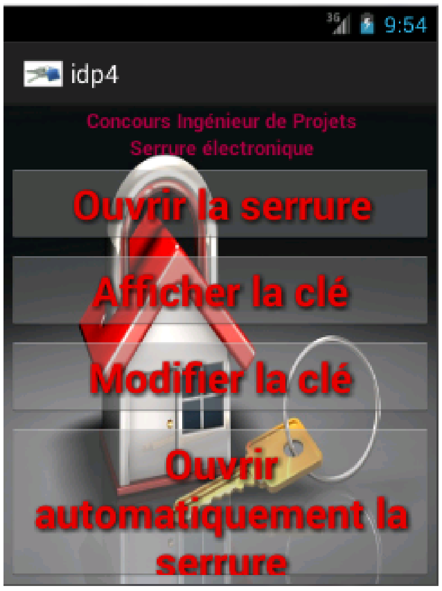
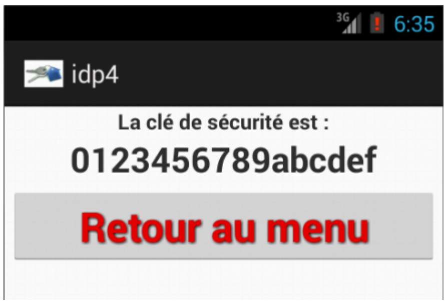


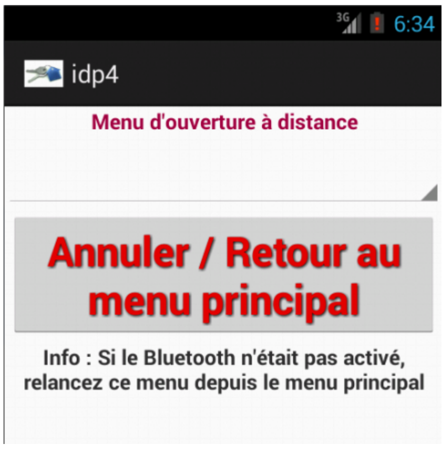
Figure B.1: Main menu.



La clé de sécurité est :

0123456789abcdef

Figure B.2: Display menu.


6:34
idp4
Menu d'ouverture à distance
Annuler / Retour au menu principal

Info : Si le Bluetooth n'était pas activé, relancez ce menu depuis le menu principal

Figure B.3: Open menu

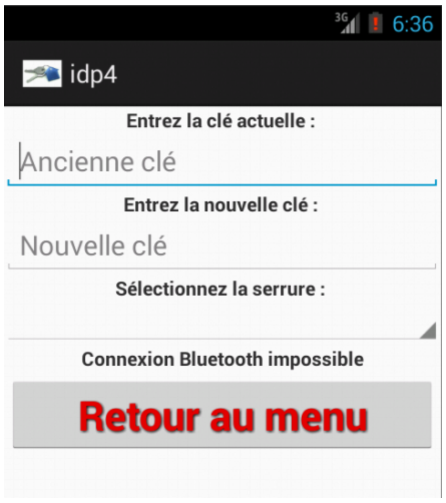


Figure B.4: Key modification menu.

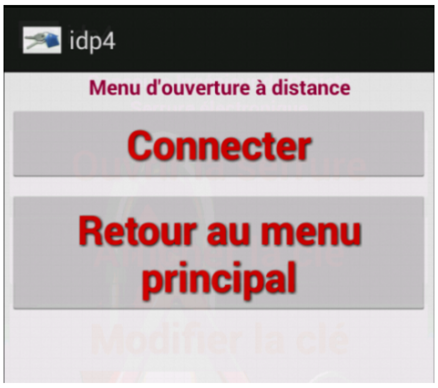


Figure B.5: Auto-connect menu.

Appendix C

Communication Rules

Here are the communication rules that were used. All the characters are ASCII.

C.1 Smartphone

Asking a random key

- Send: randomXXXXXXXXXX[1]
- Cryptography: clear

Open

- Send: openXXXXXXXXXXXX
- Cryptography: encrypted with the random key

Inform modification

- Send: modifXXXXXXXXXXX
- Cryptography: clear

Set new key

- Send: the new key
- Cryptography: encrypted with the random key

C.2 Electric Lock

Send a random key

- Send: the random key
- Cryptography: Encrypted with the random key

[1]X = Don't care

Send acknowledgment

- Send: okXXXXXXXXXXXXXX
- Cryptography: clear

Send error message

- Send: errorXXXXXXXXXXX
- Cryptography: clear

Send battery level

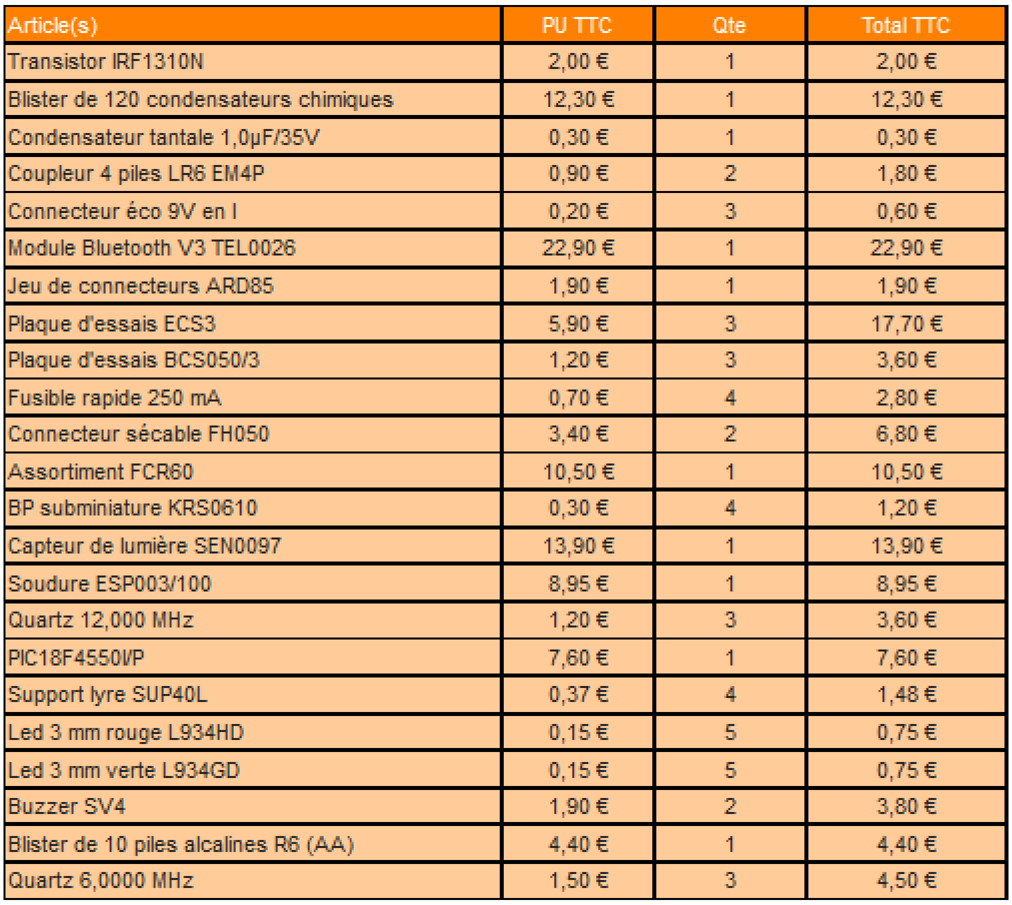
- Send: NbXXXXXXXXXXXXXXX
- Cryptography: clear

where Nb is a 8-bits signed number with first bit always zero. 127 = highest level, 0 = battery is dead.

Appendix D

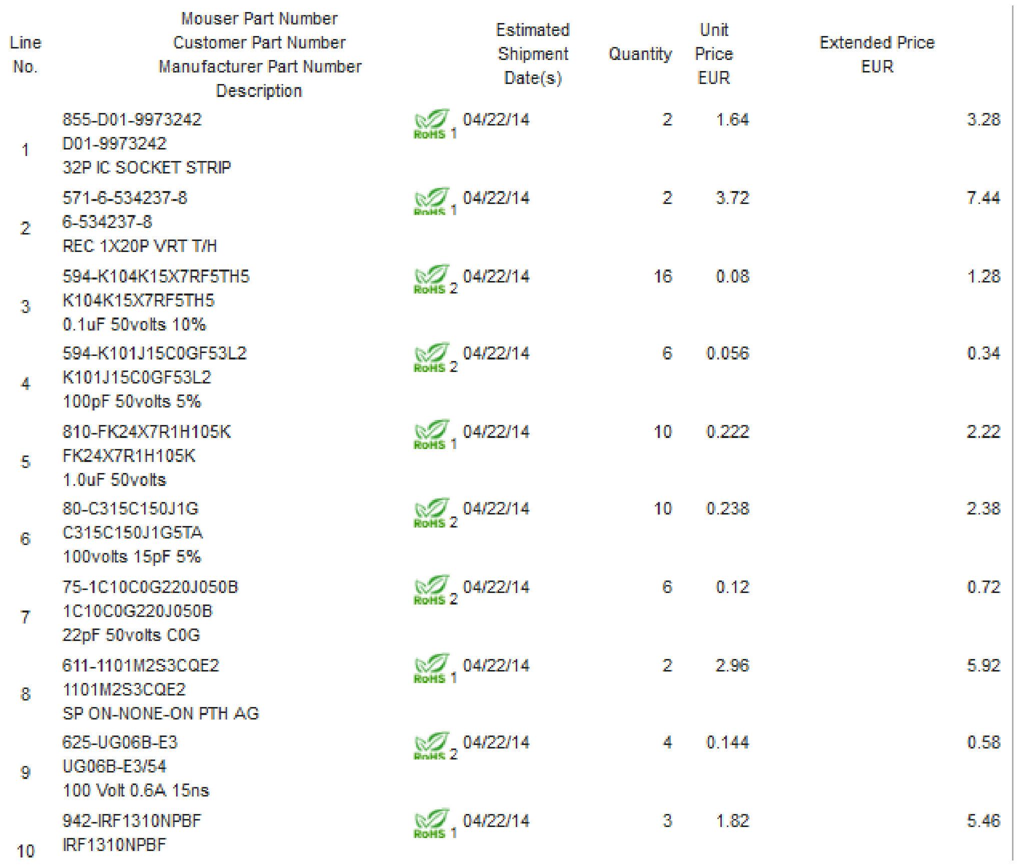
Detailed Ordering

D.1 Gotronic Ordering



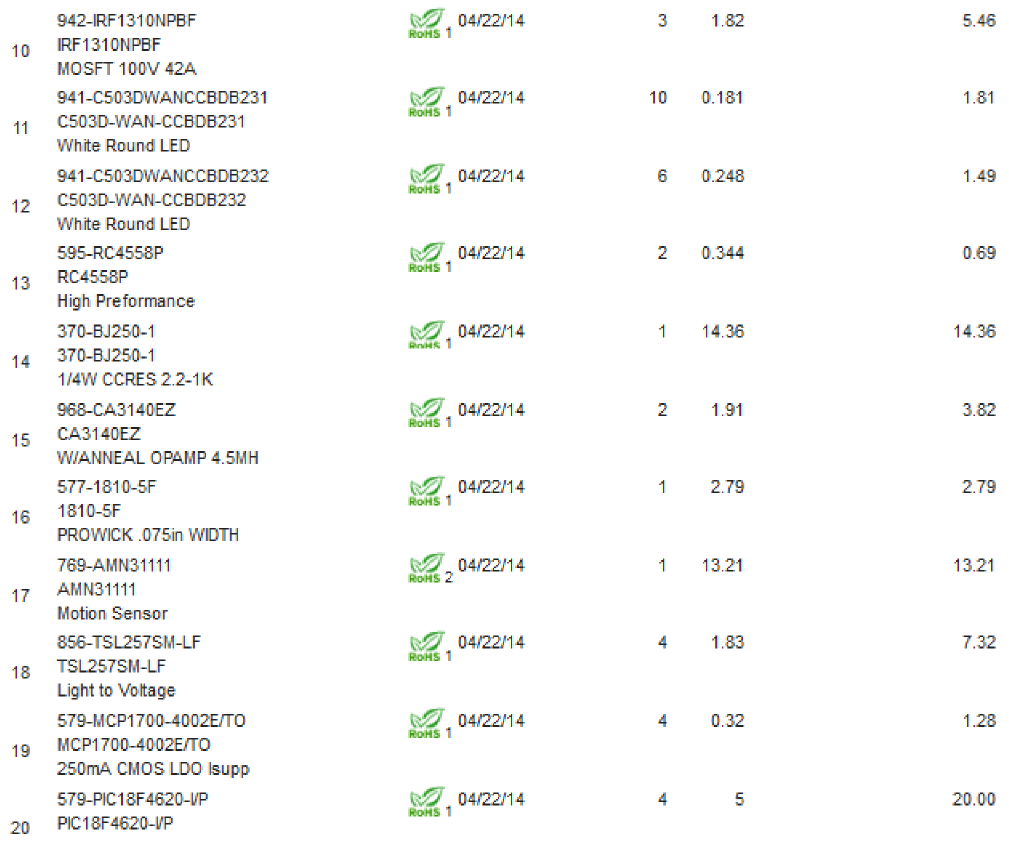
Article(s)	PU TTC	Qte	Total TTC
Transistor IRF1310N	2,00 €	1	2,00 €
Blister de 120 condensateurs chimiques	12,30 €	1	12,30 €
Condensateur tantale 1,0µF/35V	0,30 €	1	0,30 €
Coupleur 4 piles LR6 EM4P	0,90 €	2	1,80 €
Connecteur éco 9V en I	0,20 €	3	0,60 €
Module Bluetooth V3 TEL0026	22,90 €	1	22,90 €
Jeu de connecteurs ARD85	1,90 €	1	1,90 €
Plaque d'essais ECS3	5,90 €	3	17,70 €
Plaque d'essais BCS050/3	1,20 €	3	3,60 €
Fusible rapide 250 mA	0,70 €	4	2,80 €
Connecteur sécable FH050	3,40 €	2	6,80 €
Assortiment FCR60	10,50 €	1	10,50 €
BP subminiature KRS0610	0,30 €	4	1,20 €
Capteur de lumière SEN0097	13,90 €	1	13,90 €
Soudure ESP003/100	8,95 €	1	8,95 €
Quartz 12,000 MHz	1,20 €	3	3,60 €
PIC18F4550I/P	7,60 €	1	7,60 €
Support lyre SUP40L	0,37 €	4	1,48 €
Led 3 mm rouge L934HD	0,15 €	5	0,75 €
Led 3 mm verte L934GD	0,15 €	5	0,75 €
Buzzer SV4	1,90 €	2	3,80 €
Blister de 10 piles alcalines R6 (AA)	4,40 €	1	4,40 €
Quartz 6,0000 MHz	1,50 €	3	4,50 €

Figure D.1: Gotronic ordering.



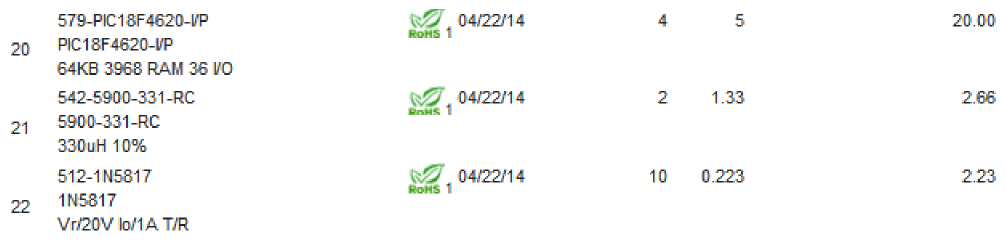
Line No.	Mouser Part Number Customer Part Number Manufacturer Part Number Description		Estimated Shipment Date(s)	Quantity	Unit Price EUR	Extended Price EUR
1	855-D01-9973242 D01-9973242 32P IC SOCKET STRIP	RoHS 1	04/22/14	2	1.64	3.28
2	571-6-534237-8 6-534237-8 REC 1X20P VRT T/H	RoHS 1	04/22/14	2	3.72	7.44
3	594-K104K15X7RF5TH5 K104K15X7RF5TH5 0.1uF 50volts 10%	RoHS 2	04/22/14	16	0.08	1.28
4	594-K101J15C0GF53L2 K101J15C0GF53L2 100pF 50volts 5%	RoHS 2	04/22/14	6	0.056	0.34
5	810-FK24X7R1H105K FK24X7R1H105K 1.0uF 50volts	RoHS 1	04/22/14	10	0.222	2.22
6	80-C315C150J1G C315C150J1G5TA 100volts 15pF 5%	RoHS 2	04/22/14	10	0.238	2.38
7	75-1C10C0G220J050B 1C10C0G220J050B 22pF 50volts C0G	RoHS 2	04/22/14	6	0.12	0.72
8	611-1101M2S3CQE2 1101M2S3CQE2 SP ON-NONE-ON PTH AG	RoHS 1	04/22/14	2	2.96	5.92
9	625-UG06B-E3 UG06B-E3/54 100 Volt 0.6A 15ns	RoHS 2	04/22/14	4	0.144	0.58
10	942-IRF1310NPBF IRF1310NPBF	RoHS 1	04/22/14	3	1.82	5.46

Figure D.2: Mouser ordering - part 1.



10	942-IRF1310NPBF IRF1310NPBF MOSFT 100V 42A	RoHS 1	04/22/14	3	1.82	5.46
11	941-C503DWANCCBDB231 C503D-WAN-CCBDB231 White Round LED	RoHS 1	04/22/14	10	0.181	1.81
12	941-C503DWANCCBDB232 C503D-WAN-CCBDB232 White Round LED	RoHS 1	04/22/14	6	0.248	1.49
13	595-RC4558P RC4558P High Preformance	RoHS 1	04/22/14	2	0.344	0.69
14	370-BJ250-1 370-BJ250-1 1/4W CCRES 2.2-1K	RoHS 1	04/22/14	1	14.36	14.36
15	968-CA3140EZ CA3140EZ W/ANNEAL OPAMP 4.5MH	RoHS 1	04/22/14	2	1.91	3.82
16	577-1810-5F 1810-5F PROWICK .075in WIDTH	RoHS 1	04/22/14	1	2.79	2.79
17	769-AMN31111 AMN31111 Motion Sensor	RoHS 2	04/22/14	1	13.21	13.21
18	856-TSL257SM-LF TSL257SM-LF Light to Voltage	RoHS 1	04/22/14	4	1.83	7.32
19	579-MCP1700-4002E/TO MCP1700-4002E/TO 250mA CMOS LDO Isupp	RoHS 1	04/22/14	4	0.32	1.28
20	579-PIC18F4620-I/P PIC18F4620-I/P	RoHS 1	04/22/14	4	5	20.00

Figure D.3: Mouser ordering - part 2.



20	579-PIC18F4620-I/P PIC18F4620-I/P 64KB 3968 RAM 36 I/O	RoHS 1	04/22/14	4	5	20.00
21	542-5900-331-RC 5900-331-RC 330uH 10%	RoHS 1	04/22/14	2	1.33	2.66
22	512-1N5817 1N5817 Vr/20V Io/1A T/R	RoHS 1	04/22/14	10	0.223	2.23

Figure D.4: Mouser ordering - part 3.

Bibliography

[1] David Flowers. *Data Encryption Routines for the PIC18*. Microchip Technology Inc. AN953.

[2] C. Paar and J. Pelzl. *Understanding Cryptography*. Springer Verlag.

[3] Paul Paglia. *LDO Thermal Considerations*. Microchip Technology Inc. AN761.

[4] Amar Palacherla. *Math Utility Routines*. Microchip Technology Inc. AN544.