Software and hardware implementation for secure RF communications on low power devices

A Degree Thesis
Submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Auri Botines Puertas

In partial fulfilment
of the requirements for the degree in
Telecommunications Technologies and Services
ENGINEERING

Advisor: Sergi Bermejo

Barcelona, June 2021

Abstract

In recent years, there has been an explosion of internet connected and other type of remote-controlled devices. Companies and open source platforms have managed to make those technologies much more accessible an easier to use by makers and other individuals. However, there are multiple projects that lack security measures against potential wireless attacks which could be performed without the user's notice.

This work presents both hardware and software developments designed with security in mind, using state of the art components and algorithms that allow cheap low-power devices to exchange small messages securely. Despite those added measures, the end-user interface and high-level programming of the hardware remains simple.

The resulting examples show how perform more secure RF communications with these types of devices and how they avoid potential attacks.

Resum

Durant els últims anys hi ha hagut un gran increment de dispositius connectats a internet i d'altres tipus de dispositius amb control remot. Les empreses i plataformes de codi lliure han aconseguit fer aquestes tecnologies molt més accessibles per a "makers" i molts altres usuaris. De totes maneres, hi ha força projectes que no disposen o no mostren mesures de seguretat contra possibles atacs en el medi que podrien ser realitzats sense que el propi usuari se n'adoni.

En aquest treball es desenvolupa una part de maquinari i una de programari tenint en compte la seguretat en tot moment. S'utilitzen els últims components i algorismes que permeten a dispositius de poca potència poder intercanviar petits missatges amb seguretat. Tot i les mesures afegides, la interfície d'usuari final i la programació a alt nivell s'ha mantingut el més simple possible.

Els exemples finals mostren com efectuar comunicacions RF segures amb aquests tipus de dispositius i com s'eviten possibles atacs.

Resumen

A lo largo de los últimos años se ha producido un gran incremento de dispositivos conectados a internet y a otros dispositivos con control remoto. Las empresas y plataformas de código libre han hecho que estas tecnologías sean mucho más accesibles para “makers” y otros muchos usuarios. Sin embargo, existen bastantes proyectos que no disponen o no muestran medidas de seguridad contra posibles ataques en el medio que podrían realizarse sin que el propio usuario se diera cuenta.

En este trabajo se desarrolla una parte hardware y otra software teniendo en cuenta la seguridad en todo momento. Se han utilizado los últimos componentes y algoritmos que permiten a dispositivos de poca potencia poder intercambiar pequeños mensajes de forma segura. A pesar de las medidas añadidas, la interfaz de usuario final y la programación a alto nivel se han mantenido lo más simple posible.

En los ejemplos finales se demuestra como efectuar comunicaciones RF seguras con estos dispositivos y como se evitan posibles ataques.

Acknowledgements

I would like to thank both the project supervisor and my family for the patience and effort that have had during the realization of this work. I am also grateful for all individuals that did contribute or help this project in any way or form.

Revision history and approval record

Revision	Date	Purpose
0	01/03/2021	Document creation
1	18/04/2021	Document format and WP0 documentation
2	08/05/2021	WP0 documentation
3	05/06/2021	Updated HW documentation (WP1). Added SW development documentation (WP2). Added appendices. Added repositories.
4	09/06/2021	Revision modifications (05/06/2021 – Doc. rev.1)
5	17/06/2021	Final candidate
6	18/06/2021	Last revision

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Auri Botines Puertas	auri.botines@estudiantat.upc.edu
Sergi Bermejo	sergio.bermejo@upc.edu

Written by:		Reviewed and approved by:	
Date	18/06/2021	Date	18/06/2021
Name	Auri Botines Puertas	Name	Sergi Bermejo
Position	Project Author	Position	Project Supervisor

Table of contents

List of Figures

List of Tables

1. Introduction

1.1. Statement of purpose

This project aims to fill a gap found on a lot of remote devices that are being deployed in home automation, smart cities and many other areas where remote sensing and actuation have great advantages. There are numerous existing projects and companies providing software and hardware allowing any individual to build or buy those services and devices. Some of those projects are the most widespread, usually because they are the cheapest, easiest to integrate and openly documented. But a few of them have considered security measures like encryption, data integrity, authentication and anti-replay mechanisms, an important step to keep privacy and potential attacks away.

The main objective of this project is to develop and provide an easy to use and easy to integrate secure software and hardware solution for small low power devices remote communications. This solution should be programmable with high level libraries and simple function calls which at the same time must provide great secure mechanisms when used for RF communications. In addition, the board needs to be designed for the lowest power consumption possible and have battery management hardware built-in.

1.2. Requirements and specifications

This project requirements and specifications are designed for a certain type of devices that require high security but at the same time they are not really high performing and often need to consume as little as possible. Those devices are remotely controlled and may report or actuate over critical infrastructures. The physical device direct manipulation or attack (tampering) has not been considered in this work, the main focus has been the protection against attacks in the wireless environment. However, the board can support two different hardware secure elements that allows the user to implement tamper-proof security measures (preventing key access, even when an attacker has complete access to the device itself).

A final point that has also been considered is the easy integration with generic and available hardware and software solutions (like Arduino). Some implementations on those platforms lack these security measures but at the same time they are very easily accessible and therefore heavily used (with the consequent large number of potential insecure devices).

Project requirements

- Hardware:
  - Crypto integrated circuit (TRNG, SHA3)
  - Easy to integrate (microcontroller, PCB pinout, documentation, …)
  - Very low power (long battery life)
  - Small footprint
  - Integrated battery management
- Software:
  - Easy to program (C++, C, Arduino)
  - Symmetric-key cryptography (user has device access)
  - Fast software-implementable encryption & authentication algorithm
  - Git - updatable (bug & vulnerability fixes)
  - Post-quantum considerations

Project specifications

- Hardware (main components):
  - ARM Cortex-M microcontroller
  - Crypto coprocessor
  - SPI serial flash module
  - RF transceiver
  - Battery charge management controller
  - External antenna connector
- Software:
  - Arduino IDE integration
  - Encryption & authentication implementation
  - Custom secure communication implementation(s)
  - Github / Gitlab for code hosting and version control

1.3. Methods and procedures

Hardware design: The board developed and manufactured during this project is a whole new design. It uses a different microcontroller (low-power version) as the one which can be found in the most similar boards. It adds multiple components (cryptography, sensors, …). The PCB layout has been designed from the ground up.

Software design: The software is a mix of some previous projects and existing code implementations with new and modified parts from the author. References to original parts are specified in each section accordingly; see “3.2” and “2.5.1”.

1.4. Work plan

Below there is a description of the plan made before the project started. It was divided into four work packages each one having a list of specific tasks to be done (not necessarily in parallel). At the end there is a Gantt diagram that helps to visualize the timing of those tasks.

The time plan modifications made posteriori at the mid-term critical review and at different stages of the project are pointed out in red (forecasted) and green (actual date it ended). A more details are described in section "Deviations and incidences".

Work Packages

<table>
<tr><td colspan="2">SW & HW for secure RF communications on low power devices</td><td>WP ref: 0</td></tr>
<tr><td colspan="2">Major constituent: previous analysis & documentation</td><td>Sheet 1 of 2</td></tr>
<tr><td colspan="2" rowspan="2">Short description:

Previous analysis and documentation for various project-related tasks including: current and latest cryptography, post-quantum cryptography, current analogous SW & HW implementations, PCB RF design techniques, low power and battery management best practices.</td><td>Planned start date:
15/02/2021

Planned end date:
19/04/2021</td></tr>
<tr><td>Start event:
Project start

End event:
SW and HW implementations start</td></tr>
<tr><td>Internal task T1: pre-shared key algorithms and its uses.
Internal task T2: symmetric cryptography pros & cons.
Internal task T3: applicable latest security standards (RFC's).
Internal task T4: post-quantum cryptography implications.
Internal task T5: common and integrable components research.
Internal task T6: RF PCB design techniques.
Internal task T7: low power design.</td><td>Deliverables:

Summary of the research and analysis done for each area.</td><td>Dates:
07/05/2021</td></tr>
</table>

<table>
<tr><td colspan="2">SW & HW for secure RF communications on low power devices</td><td>WP ref: 1</td></tr>
<tr><td colspan="2">Major constituent: hardware prototypes (PCB's)</td><td>Sheet 1 of 2</td></tr>
<tr><td colspan="2" rowspan="2">Short description:

Hardware design of the board module (components, schematic, PCB, fabrication, assembly and tests).</td><td>Planned start date:
20/03/2021

Planned end date:
08/05/2021</td></tr>
<tr><td>End event:
Assembled PCB's</td></tr>
<tr><td>Internal task T1: general functional diagram and components.
Internal task T2: schematic design.
Internal task T3: PCB layout (placement and routing).
Internal task T4: fabrication output generation and manufacturing.
Internal task T5: PCB components.
Internal task T6: manual assembly of the boards.
Internal task T7: general tests.</td><td>Deliverables:

Finished working PCB prototypes</td><td>Dates:

20/05/2021</td></tr>
</table>

SW & HW for secure RF communications on low power devices	WP ref: 2	
Major constituent: **software implementations and programming**	Sheet 2 of 2	
Short description: Programming of board modules and its driver libraries. Design and implementation of easy to use secure communication algorithms.	Planned start date: 20/04/2021 Planned end date: 12/06/2021	
	Start event: End of HW development. End event: Working code into PCB's.	
Internal task T1: programming board modules (drivers/libraries) Internal task T2: secure algorithms implementations Internal task T3: testing algorithms with the boards	Deliverables: Crypto code	Dates: 05/06/2021

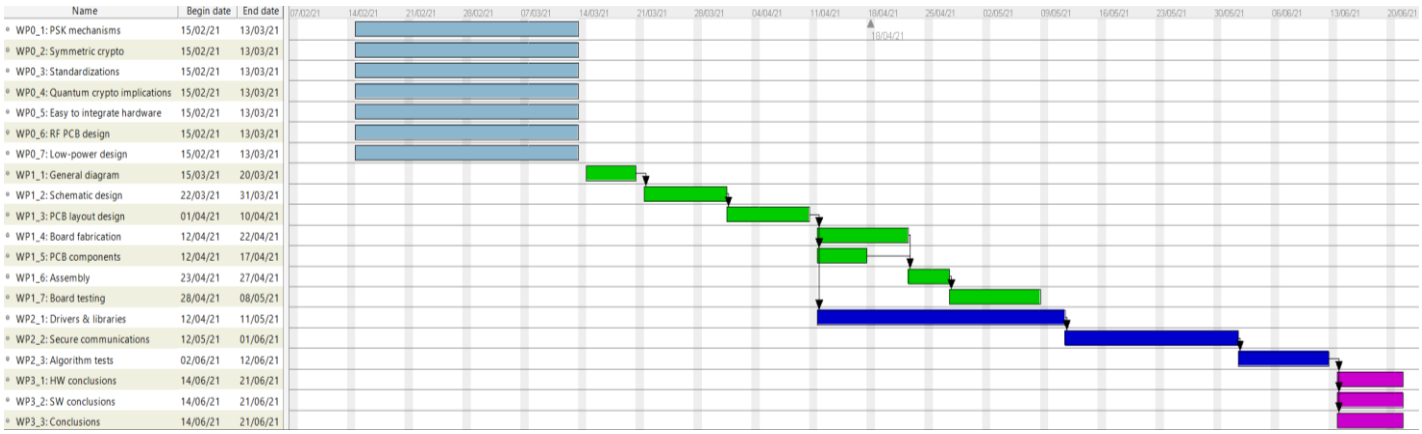
SW & HW for secure RF communications on low power devices	WP ref: 3	
Major constituent: **final conclusions**	Sheet 2 of 2	
Short description: Project final conclusions for its different areas. Final documentation and results.	Planned start date: 14/06/2021 Planned end date: 21/06/2021	
	Start event: All previous work packages finished.	
Internal task T1: hardware implementation conclusions Internal task T2: software implementation conclusions Internal task T3: general work conclusions	Deliverables: Finished project	Dates: 17/06/2021

Milestones

WP#	Task#	Short title	Milestone / deliverable	End date (week)
0	1	PSK mechanisms	(documentation)	19/04/2021 (17)
0	2	Symmetric crypto	(documentation)	19/04/2021 (17)
0	3	Standardizations	(documentation)	19/04/2021 (17)
0	4	Post-quantum crypto implications	(documentation)	19/04/2021 (17)
0	5	Easy to integrate hardware	(documentation)	19/04/2021 (17)
0	6	RF PCB design	(documentation)	19/04/2021 (17)
0	7	Low power design	(documentation)	19/04/2021 (17)
1	1	General diagram		20/03/2021 (12)
1	2	Schematic design		31/03/2021 (14)
1	3	PCB layout design		10/04/2021 (15)
1	4	Board fabrication		23/04/2021 (17)
1	5	PCB components		23/04/2021 (17)
1	6	Assembly		28/04/2021 (18)
1	7	Board testing	Finished working PCB's	20/05/2021 (21)
2	1	Drivers and libraries		12/05/2021 (20)
2	2	Secure communications		02/06/2021 (23)
2	3	Algorithm testing	Crypto code & libraries	05/06/2021 (23)
3	1	Conclusions (HW)		17/06/2021 (25)
3	2	Conclusions (SW)		17/06/2021 (25)
3	3	Conclusions	Finished project	17/06/2021 (25)

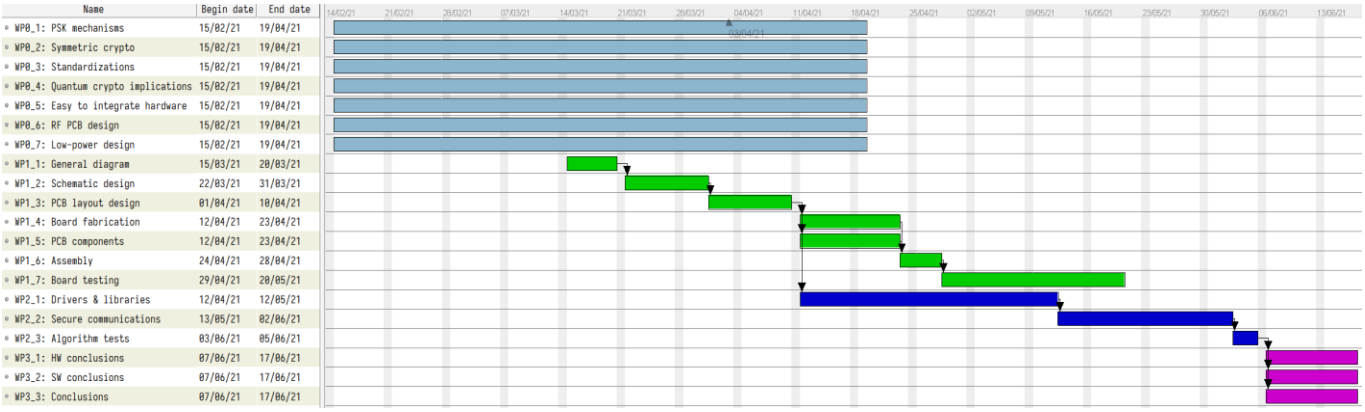
1.5. Time Plan (Gantt diagram)

First plan (estimated at mid februray 2021)



Name	Begin date	End date
WP0_1: PSK mechanisms	15/02/21	13/03/21
WP0_2: Symmetric crypto	15/02/21	13/03/21
WP0_3: Standardizations	15/02/21	13/03/21
WP0_4: Quantum crypto implications	15/02/21	13/03/21
WP0_5: Easy to integrate hardware	15/02/21	13/03/21
WP0_6: RF PCB design	15/02/21	13/03/21
WP0_7: Low-power design	15/02/21	13/03/21
WP1_1: General diagram	15/03/21	20/03/21
WP1_2: Schematic design	22/03/21	31/03/21
WP1_3: PCB layout design	01/04/21	10/04/21
WP1_4: Board fabrication	12/04/21	22/04/21
WP1_5: PCB components	12/04/21	17/04/21
WP1_6: Assembly	23/04/21	27/04/21
WP1_7: Board testing	28/04/21	08/05/21
WP2_1: Drivers & libraries	12/04/21	11/05/21
WP2_2: Secure communications	12/05/21	01/06/21
WP2_3: Algorithm tests	02/06/21	12/06/21
WP3_1: HW conclusions	14/06/21	21/06/21
WP3_2: SW conclusions	14/06/21	21/06/21
WP3_3: Conclusions	14/06/21	21/06/21

Actual times (not a plan)



Name	Begin date	End date
WP0_1: PSK mechanisms	15/02/21	19/04/21
WP0_2: Symmetric crypto	15/02/21	19/04/21
WP0_3: Standardizations	15/02/21	19/04/21
WP0_4: Quantum crypto implications	15/02/21	19/04/21
WP0_5: Easy to integrate hardware	15/02/21	19/04/21
WP0_6: RF PCB design	15/02/21	19/04/21
WP0_7: Low-power design	15/02/21	19/04/21
WP1_1: General diagram	15/03/21	28/03/21
WP1_2: Schematic design	22/03/21	31/03/21
WP1_3: PCB layout design	01/04/21	10/04/21
WP1_4: Board fabrication	12/04/21	23/04/21
WP1_5: PCB components	12/04/21	23/04/21
WP1_6: Assembly	24/04/21	28/04/21
WP1_7: Board testing	29/04/21	20/05/21
WP2_1: Drivers & libraries	12/04/21	12/05/21
WP2_2: Secure communications	13/05/21	02/06/21
WP2_3: Algorithm tests	03/06/21	05/06/21
WP3_1: HW conclusions	07/06/21	17/06/21
WP3_2: SW conclusions	07/06/21	17/06/21
WP3_3: Conclusions	07/06/21	17/06/21

1.6. Deviations and incidences

The first part of the project (research and documentation) took more time than the expected, partly because this project focused on the latest security standards and research being developed at the same time of this work.

The known global chip shortage crisis originated by multiple external factors caused some delays and modifications of the hardware part of this project, however, it did not affect the final result intended at the beginning.

Some parts of the project that couldn't be implemented can be found in the latest section "Conclusions and future development".

2. State of the art of the technology used or applied in this thesis

This chapter is intended to show and provide some useful references, short descriptions and best practices about simple wireless communications security and PCB design, but it is not a deep analysis or explanation of the best definitive methods to implement a secure RF communication or design a PCB. In the other hand, it tries to highlight state of the art methods, potential issues and pros and cons of the different technologies used in today's real world.

It focuses on a specific use case defined in "Requirements and specifications" and, therefore, the mechanisms chosen are thought to be the optimal for this case but not for every possible type of communication or device. The physical intrusion was not the main priority of this work, but strong symmetric ciphers and good true random number generators are essential to acquire good enough security levels. [1].

2.1. Pre-shared key algorithms and its uses

Pre-shared key algorithms are an essential part in symmetric cryptography although they are also used in some parts of asymmetric or public cryptography. They use a single secret key to encrypt/decrypt and perform other operations between the two or more entities which are communicating.

This key must be previously shared and known by all the members or entities that are communicating. The method in which this secret key is shared is not a part of this work but it is a critical part that should be considered when implementing and deploying devices or systems using those mechanisms, because anyone knowing the secret key will have access to all communications.

Public key cryptography methods are often used to generate a similar secret key from public and private keys, those methods are inherently slower, more power demanding and often require a more complex infrastructure than PSK systems. Despite that, they are required when having physical access to all the communicating devices is impossible or impractical (for example the vast majority of internet connected devices), usually those devices are very far away.

For remote, but not too far away communications with low bit rates (car key fobs, garage and household access doors, alarms, remote sensors & actuators, etc...), it makes sense to use pre-shared key systems. Once the owner/s having physical access to the devices (only one time) is already been taken in account and the devices themselves are secure, having private, secure and resistance to potential attacks is a must.

The main application of pre-shared key algorithms is encryption, integrity and authentication, but not always all mechanisms provide those. There are also other aspects to consider like constant time, nonce misuse resistance, message repudiation, message committing, security level, post-quantum cryptography implications, etc. This project aims to achieve security primarily by using strong cryptography and reduce potential wireless attacks [2].

Some widely used **PSK examples**:

- Rolling/hopping codes [1]
  They are often used in keyless entry systems, may be vulnerable to advanced replay attacks like the "Rolljam vulnerability". There are new keyless systems which use other advanced algorithms not related with rolling codes, they use an exchange of messages between parties, see [2] and [3].

- Wi-Fi protected access point passwords (for example, WPA-PSK)

- TLS-PSK
  Set of different transport layer security *cipher suites* used on the internet usually applied on processing power constrained and manually configured closed environments. See [4] and [5] (latest).

2.2. Pros and cons of symmetric over asymmetric cryptography

In the table 1, some of the most remarkable properties of both symmetric and asymmetric (or public) cryptography and encryption are presented. In the end, each one can be more suitable for some type of communications and worst for other ones, it usually depends on the specific use case.

2.3. Latest applicable security standards and considerations

There are several organizations providing standardizations and publishing detailed descriptions on several algorithms related with secure communications. This work aims to use the latest and more secure publicly available algorithms which can be implemented in the hardware modules developed with small modifications.

The latest standard being used on Internet (and IoT) devices is TLS1.3 [6] (2018) which uses some new cipher suites (like Chacha20-Poly1305) and removes old insecure ones. Those cipher suites define how key exchange, authentication, hashing and encryption algorithms are is used in the communications. There are other and more recent algorithms

not specifically designed for the Internet which may be faster and more suitable for small low power devices.

While this work was being done, the NIST announced (March 2021) the ten finalists of the "*Lightweight Cryptography*" project which presents algorithms specially designed for highly-constrained devices like the one designed in this project. Although the finalist is expected to be announced withing the next 12 months, the software developed for this work tries to use one of those secure-proven methods and hopes to keep the algorithm up to date or change it in case any vulnerability were to be detected. All those algorithms don't need specific hardware and can be implemented on practically any microcontroller, therefore, they can be easily maintained and updated over time without needing hardware changes.

	Symmetric / private key		**Asymmetric / public key**	
	Pros	***Cons***	***Pros***	***Cons***
Certificates	Not used.	Not used.	Unique key (root CA). Only one entity to be protected.	Unique key (root CA). Only one single entity can compromise all certificates it generated.
Secret keys	Same key for encryption and decryption.	Same unique key in different devices, more vulnerable points. Any device could compromise all the other ones sharing the same key without notice. Need to share the key in advance.	Public-private key groups (that usually change over time connection establishment). No need to pre share any secret key.	Different keys for encryption and decryption. Public key sharing method required (certificates).
Key sizes	Smaller key sizes for equivalent *security level*.			Bigger key sizes for equivalent security level.
Computation power required	Usually less.			Usually more.
Time required	Less.			More.
Non-repudiation		Not provided.	Can be provided (by a trusted third party).	
Post-quantum	Easily remediable with increased key sizes and randomness.			More complex algorithms.

Table 1: Pros and cons of symmetric and asymmetric cryptography

2.4. Post-quantum cryptography implications

In the past few years, the new concept of post-quantum cryptography has been emerged due to the new implications the future quantum computers may have over the cryptography used nowadays. Those computers may be able to solve very hard or impossible mathematical problems at the present time. Some of those mathematical problems are used especially on public-key cryptography (for example RSA prime factoring), as a consequence there is an active research on public-key encryption, and key-establishment algorithms resistant to quantum computers.

However, the larger part of symmetric-key encryption algorithms does not rely on that kind of mathematical problems hence the only implication of quantum computers may be a square root speed-up factor over a simple brute force attack (Grover's algorithm [7]). It is known that, with increased key sizes and key management protocols (multiple algorithms already exist), symmetric cryptography is resistant to quantum attacks. In fact, some popular algorithms (like SHA3 or AES with 256-bit key) are already considered secure.

As a consequence, for now, using up to date standards with large symmetric key sizes is enough to have a secure encryption. An authentication and integrity protocol based on those standards should then, be also secure.

2.5. Hardware design

The hardware part of the project consists of a PCB module which integrates a number of components carefully selected to provide it with secure remote-control functionalities, very low power capabilities and an easy integration with the Arduino IDE platform and some of its libraries. Some features of the PCB main components assembled are shown below.

	Component reference	Lowest consumption	Arduino integration
Microcontroller	ATSAML21G18B	< 1 µA	Modified ArduinoCore-samd (*Mattairtech* and *the author*)
Transceiver	RFM69HCW (RFM95W LoRa compatible)	< 1 µA	Modified *LowPowerLab*'s RFM69 library (compatible with RadioHead Packet Radio)
Charge controller	MCP73831T-2ACI/OT	0.1 µA (no charge)	-
Voltage regulator	TCR3UF30A,LM(CT	0.6 µA (10 µA load)	-
HW cryptographic authentication	DS28C16Q+U (or DS2477Q+T)	3.5 µA (400 µA) (or 0 µA)	The author
External memory	AT25FF041A-SSHN-T (compatible with other generic)	< 1 µA (or 0 A)	SPIFlash

Table 2: Main hardware components

2.5.1. Integration

Since one of the objectives of this project is to be easy to develop with, both the microcontroller and the transceiver module have been selected so that they can be somewhat easy to integrate with the Arduino IDE and its libraries and drivers.

The microcontroller (SAML21G18) is the low consumption version of another one (SAMD21G18) already used in some Arduino boards, however it is not so similar and the "ArduinoCore-samd", which contains the source code and configuration files of the Arduino Microchip's SAMD21 processor boards, has to be modified. The repositories below are the ones used to develop this work. The first one was created by the author for this project:

"AtArduinoCore-samL" (Atalonica)	**MIT**	*https://github.com/Atalonica/AtArduinoCore-samL*
"ArduinoCore-samd" (Arduino LLC)	**LGPL**	*https://github.com/arduino/ArduinoCore-samd*
"ArduinoCore-samd" (MattairTech LLC)	**LGPL**	*https://github.com/mattairtech/ArduinoCore-samd*

The transceiver module can be interfaced with *LowPowerLab's RFM69* or with *RadioHead Packet Radio* Arduino libraries. This project, however, adds a security layer on top of LowPowerLab's library (see 26). Finally, the board "TFG Zero" has a pinout spacing compatible with any standard protoboard and the pin functionalities and Arduino references are shown in Figure 1.

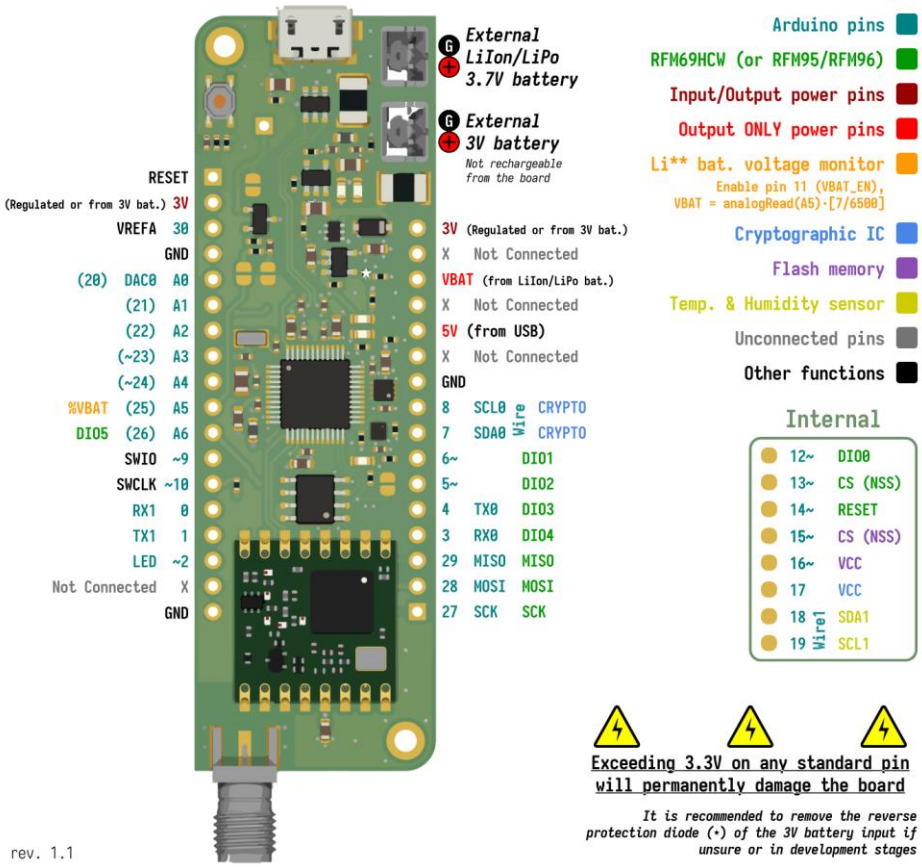


Figure 1: TFG Zero PCB pinout reference

2.5.2. PCB design techniques

To ensure good integrity and reduce loses of the signals travelling inside the printed circuit board (especially high frequency and RF signals), different methods for certain parts of the PCB have been used:

- **USB**

USB data lines are differential and must have a controlled impedance of 90 Ω. The PCB manufacturer provides a calculator that takes into consideration they PCB physical characteristics to provide us with a trace width for a specific impedance value of differential traces (KiCad built-in tools and others can also be used if the manufacturer doesn't have any specific application). The USB traces of the PCB developed in this project have a value of 0.2611 mm in width (for a 0.2032 mm spacing). For the USB signals to arrive at the same time both differential traces should have the same length (see marked left image in Figure 2). It is also important to keep an uninterrupted ground plane immediately under the controlled impedance traces and to keep same layer ground/power pours (or fills) away from those traces. This also applies to the next section (RF) where a 50 Ω matching impedance is used (instead of 90 Ω).

Figure 2: USB differential traces

- **RF**

Transceiver integrated circuits and modules specify the reference impedance value that should be used in the output/input pin (where the antenna should be connected). This impedance value must be the same for the external antenna, its connector and the PCB traces that connect the IC or module with the antenna. The transceiver used (RFM69HCW) can work with a 200 Ω or 50 Ω impedance, we've chosen the second since it is more standardized. There are different formulas and tools to calculate the matched impedance trace width for a specific impedance value. We've used the manufacturer's calculator, that, for our specific PCB parameters (dielectric, cooper) and type of line (microstrips, striplines) gave a value of 0.29337 mm.

Once we know the trace width of the RF signals, we need to ensure RF traces are shorter (in length) than the critical length to minimize RF effects. The formulas are again different depending on the line used (microstrips or striplines), in our case we used a microstrip trace:

$$Critical\ length = \frac{c}{f} \cdot \frac{1}{12\sqrt{\varepsilon_{r(effective)}}} = \frac{3 \cdot 10^8\ m/s}{868 \cdot 10^6\ Hz} \cdot \frac{1}{12\sqrt{3.3941}} = .01563\ m = 15.63\ mm \quad (2.1)$$

The above effective relative permittivity for a microstrip can be calculated as follows:

$$\varepsilon_{r(effective)} = \frac{\varepsilon_r+1}{2} + \frac{\varepsilon_r-1}{2\sqrt{1+12\left(\frac{H}{W}\right)}} = \frac{4.6}{2} + \frac{4.6^{-1}}{2\cdot\sqrt{1+12\left(\frac{0.2}{0.29337}\right)}} = 3.3941 \text{ if } \left(\frac{Width}{Height} > 1\right) \quad (2.2)$$

The relative permittivity of the dielectric should be provided by the PCB manufacturer, in our case the dielectric used is a “7628 prepreg” with an ε_r of 4.6.

As we can see below, by minimizing the distance between the antenna and the RF pin of the module, the trace length (including pads) is far below to the critical length previously calculated (7.4 mm < 15.6 mm).

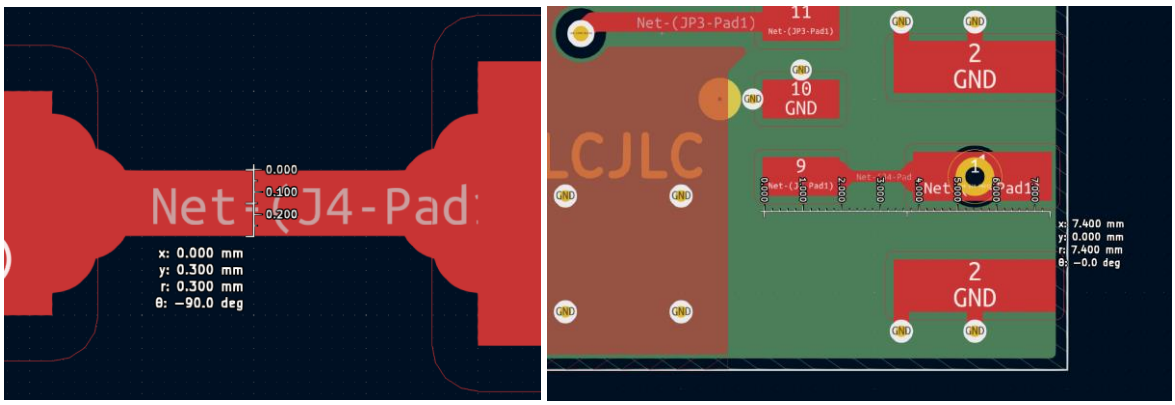


Figure 3: RF trace width and length

- **External crystal**

The external crystal oscillator purpose is to provide a very accurate constant frequency to the microcontroller. It must be placed very close to the MCU pins (and its respective capacitors). However, it can be an important source of noise that's why it is very important to not route anything else close or under it and to cut all planes under it. A solid ground plane under it can reduce electromagnetic emissions and noise but it also adds parasitic capacitance that can decrease frequency stability. For this design the ground plane directly under it has been cut out and the one at the bottom has been kept with a small gap around it connected to the main ground plane through a small trace. [8], [9], [10].

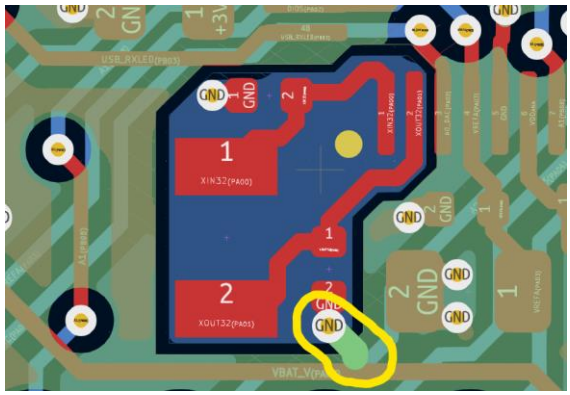


Figure 4: External crystal design

- **EMI / EMC**

There are different techniques that can be used to both prevent outside electromagnetic interferences from affecting the PCB circuits and minimizing the emissions of the board to outside devices. By having a four-layer board with an entire ground plane and ground pours under components and signals that ensure low current return paths is key to reduce those effects.

Low electromagnetic emission has been kept in mind during the design process but has not been the priority of this project, therefore the signal routing, power planes and board *stackup* could be modified to improve further the EM compatibility (EMC) of the board.

3. Methodology / project development

3.1. Hardware development

3.1.1. Diagram and schematic

Refer to appendices, first sections “Hardware diagram” and “Schematic” for both a high-level preview of the hardware components chosen and the detailed schematic used to build the PCB itself.

3.1.2. PCB layout

The printed circuit board designed consists on a four-layer stackup. The top and bottom layers are used for component routing the traces and the two inner layers are both reference planes carrying the ground and the 3V power signals. With four layers we can achieve an easier component placement, higher density and better electromagnetic, signal integrity and power distribution properties. A detailed view for all four layers layouts is shown in “PCB layout stackup” appendix.

3.2. Software development

Two objectives of this project are: providing high security but also having an easy implementation for any ordinary user to employ. The next two sections describe those two points: a secure lightweight algorithm and a brief description of an Arduino compatible library that implements it.

3.2.1. Xoodyak-based AEAD algorithm (software-only)

For an overall overview of the algorithm and its timing diagram, see appendix: “Secure algorithm summary diagram (SO)”. Note that all messages exchanged must be encrypted with any secure-proven ≥128-bit symmetric encryption algorithm, in our case we use AES-128 (built in the RFM69 transceiver module).

Parameter definitions

- **NREQ (12-byte buffer):** nonce request message, contains NREQNID, NREQRID and NREQH (in this order).
- **NREQNID (4-byte buffer):** nonce request identification name, must contain fixed values for all requests.
- **NREQRID (4-byte buffer):** nonce request random identifier, must contain random values for each request.
- **NREQH (4-byte buffer):** nonce request Xoodyak hash of NREQNID, NREQRID and KH.
- **N (16-byte buffer):** unique or true random generated array. Obtained either by a hardware certified TRNG or by an increment/decrement-only hardware counter.
- **KH (16-byte buffer):** pre-shared key used for Xoodyak hashing.

- **KX (16-byte buffer):** pre-shared key used for Xoodyak AEAD scheme (should be randomly generated).
- **KE (16-byte buffer):** pre-shared key used for top-layer encryption (should be randomly generated).
- **M (0 to 44-byte buffer)*:** contains the user message (will get encrypted and authenticated).
- **AD (1 to 4-byte buffer)*:** contains the user associated data (0-3 bytes) and 1 protocol-specific byte that indicates sizes (will get authenticated only).
- **C (0 to 44-bytes buffer):** contains the ciphertext of M (Xoodyak encryption of M). Same length as M.
- **T (16-byte buffer):** Xoodyak authentication tag used to validate AD and C integrity and authenticity.

* The combined length of **AD** and **M** must be 45 bytes maximum (i.e. if we transmit a 42-byte message, the user associated data length must be 2 bytes at most).

Nonce exchange algorithm

All the communications start with a nonce [11] exchange mechanism. This nonce is, later on, used as an initialization parameter on the encryption and authentication algorithm.
The first message is sent from whoever wants to send a secure message (encrypted, authenticated and with valid integrity) to a receiver. From now on, let's call them "A" (the sender) and "B" (the receiver). Those messages, in the order they are issued, are described below:

Nonce request (**NREQ**):
Sent from A to B asking for a nonce. Contains a user custom "nonce request" identifier **NREQNID** (same message for all nonce requests), four random bytes **NREQRID** (different for each nonce request) and a hash, **NREQH**, of those two parameters plus the pre-shared key **KH**. If **NREQ** fails the integrity check or B is waiting for an AEAD message when the **NREQ** is received, an error counter must be increased and the current minimum nonce generation time must be multiplied by this error counter.

Nonce response (**NRES**):
Sent from B to A when B receives **NREQ**. Contains the nonce (**N**) and a hash of: **N**, a transformation of (**NREQNID, NREQRID**) and **KH**. When **N** is either generated (by B) or received (by A) a short lifespan (<1s) nonce expiration timer must be started, at timeout no AEAD messages must be decrypted/encrypted.

AEAD algorithm

After a nonce is successfully received by A, this node sends the AEAD message. This message is generated using the Xoodyak AEAD algorithm [12] with the key **KH**, the nonce **N**, and the user message **M** and associated data **AD**. Refer to the parameter definitions above for its sizes (constrained by the maximum RFM69's payload size). The final payload which is send from A to B containing the unencrypted (but authenticated) **AD**, **C** (encrypted

M) and the authentication tag **T** is as follows (note that **AD** contains two parts: one is a protocol-specific information byte and the other is customizable for the user, up to 3 bytes):

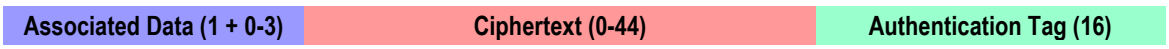


Figure 5: AEAD algorithm payload structure

If nonce timeout expires or any validation fails during encryption/decryption, buffers are emptied and the state reset. On the other hand, if a secure AEAD message passes all checks and is successfully decrypted, error counter and minimum nonce generation time are set to its default values.

3.2.2. "SecureRF" library

This is the library that implements the algorithm functionality described in the above section. It is compatible with the Arduino platform. Its built-in examples show how to securely transmit and receive critical communication lightweight commands using the RFM69 Arduino library (which is only actually required by the examples). The Xoodyak implementation used inside is a modified version of the one made by Rhys Weatherley which has coded all NIST competition finalists' algorithms.

Repositories:

"SecureRF" (Atalonica)		**MIT**	*https://github.com/Atalonica/SecureRF*
DEPENDENCY	**"RFM69" (LowPowerLab)**	**GPL3**	*https://github.com/LowPowerLab/RFM69*
"lwc-finalists" (rweather)		**MIT**	*https://github.com/rweather/lwc-finalists*

Public function prototypes and variables:

setKeys*(const unsigned char * **kx**, const unsigned char * **kh**)*

Sets hash and AEAD pre-shared keys, it must be called once before any nonce or message request/response.

Parameters

kx	Buffer to receive the input key.
kh	Buffer to receive the input key.

Returns

true	On success.
false	On error (keys are already set).

createNonceRequest*(const unsigned char * **nReqNameId**, const unsigned char * **nReqRandId**, unsigned char * **nReq**)*

Creates the nonce request message from name and random identifiers.

Parameters

nReqNameId	Buffer to receive the input unique nonce request identifier.
nReqRandId	Buffer to receive the input random nonce request identifier.
nReq	Buffer where the output message will be saved.

Returns

true	On success.
false	On error (keys not set or error generating hash).

onNonceRequest*(unsigned char * **nReq**, const unsigned char * **n**, unsigned char * **nRes**)*
Creates the nonce response message from the nonce request received and the provided nonce. **Parameters** **nReq** Buffer to receive the input nonce request. **n** Buffer to receive the new random nonce. **nRes** Buffer where the output message will be saved. **Returns** **true** On success. **false** On error (waiting AEAD, invalid nonce req., error generating hash or nonce not saved).
onNonceResponse*(unsigned char * **nRes**)*
Saves the received nonce if valid and prepares for receiving a new secure AEAD message. **Parameters** **nReq** Buffer to receive the input nonce response. **Returns** **true** On success. **false** On error (error generating hash, invalid response or nonce can't be saved).
createSecureMessage*(unsigned char * **message**, unsigned char **messageLength**, unsigned char * **ad**, unsigned char **adLength**)*
Creates the AEAD payload from user message and associated data. **Parameters** **message** Buffer to receive the user input message. **messageLength** Length of user message (in bytes). Maximum is 44 minus a.data length. **ad** Buffer to receive the user associated data. **adLength** Length of user a. data (in bytes). Maximum is 3. **Returns** **true** On success. **false** On error (keys not set, nonce expired, message length error, a. data length error or error in xoodyak AEAD encryption).
*static unsigned char **SECURE_PAYLOAD[62]***
Buffer containing the AEAD secure message payload ready to be send. It is updated when ***createSecureMessage****(…)* function returns true.
*static uint8_t **SECURE_PAYLOAD_LEN***
Length (in bytes) of ***SECURE_PAYLOAD***.
waitingSecureMessage*()*
Used to check if node is waiting for an AEAD message. Should be called before ***onSecureMessage****(…)*.
onSecureMessage*(unsigned char * **payload**)*
Reads, decrypts and validates an AEAD input payload message. **Parameters** **payload** Buffer to receive the input payload just received. **Returns** **true** On success. **false** On error (nonce expired, error in xoodyak AEAD decryption or validation).
*static unsigned char **PLAINTEXT[45]***
Buffer containing the decrypted and authenticated user message. It is updated when ***onSecureMessage(…)*** function returns true.
*static uint8_t **PLAINTEXT_LEN***
Length (in bytes) of ***PLAINTEXT***.
*static unsigned char **ASSOCIATED[5]***
Buffer containing the authenticated user a.data. Updated when ***onSecureMessage(…)*** function returns true.
*static uint8_t **ASSOCIATED_LEN***
Length (in bytes) of ***ASSOCIATED***.

Table 3: SecureRF library function prototypes and variables

4. Results

4.1.1. Integration

By using the developed Arduino-compatible core for the board and the library that handles the security algorithm (referenced in the previous section), the prototype board can be successfully programmed with the Arduino IDE (including the new 2.0 beta IDE) by using high-level functions that allow a fairly inexperienced user to handle it.

The examples show how to perform an easy message exchange between two pre-programmed remote nodes while using robust security mechanisms which give the communication the extra layer of protection intended at the beginning of this work.

4.1.2. Simulated wireless attacks

For the secure messages (AEAD), the security level of the messages is directly the one used by the Xoodyak payload times the AES-128 provides, the nonce exchange messages, however, are only AES-encrypted. Both methods have a security level of 128 bits.

Delayed replay attack:

An attacker may be interested in replaying (or sending a command that was sent previously) later on, whenever he wants. The algorithm created prevents that from happening.

First, the attacker continuously listens during some time and identifies how the messages are sent (if there is a nonce exchange, ACK's, etc). After that, it may record specific payloads, for example nonce requests and a specific AEAD messages he is interested to replay. Finally, the attacker executes and tries to impersonate another node by sending its recorded messages. If the attacker requests a nonce and it receives a previously sent value then he may try to replay the AEAD message that was sent with that nonce. The algorithm expands over large time periods the little probabilities of successfully executing this attack.

The diagram below (Figure 6) shows how the attacker could replay an AEAD message without ever having to decrypt any payload.

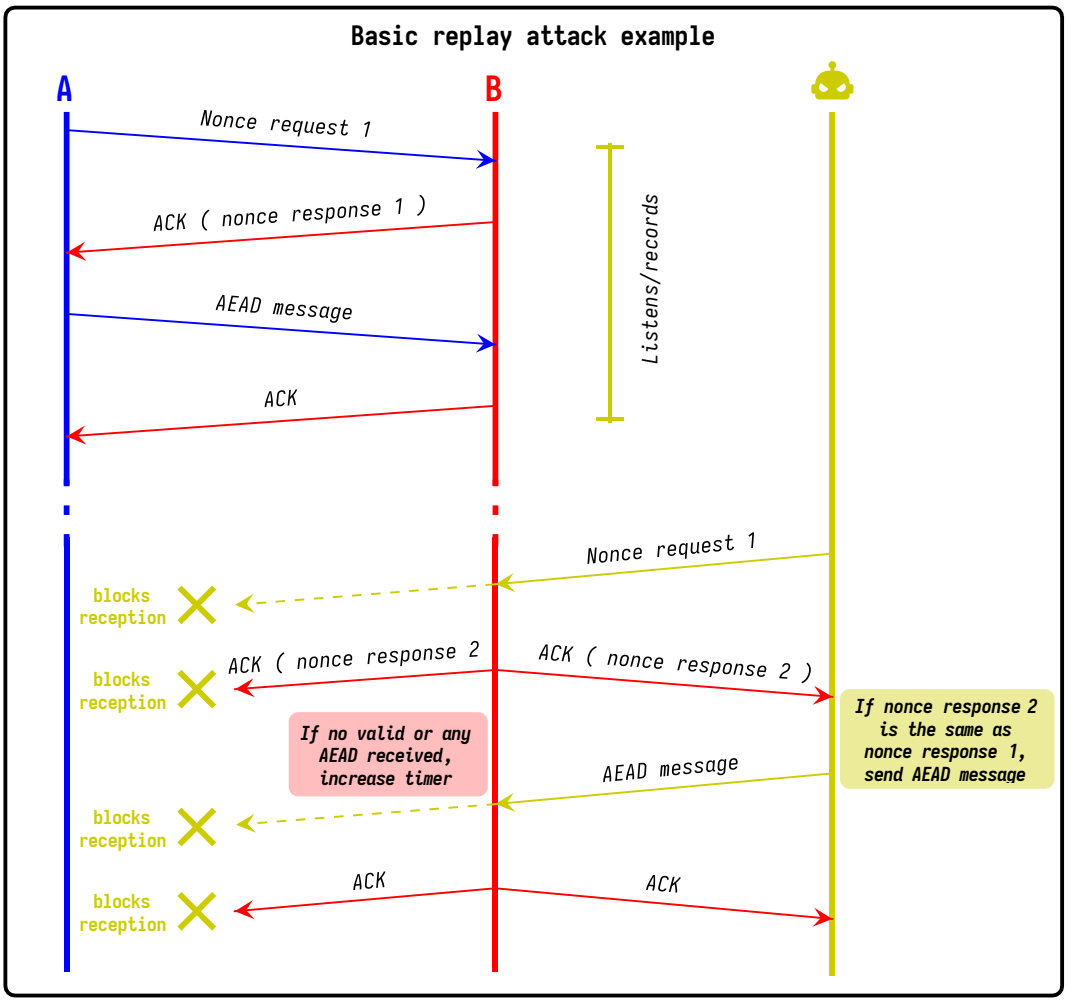


Figure 6: Basic replay attack example

The probability of a successful delayed replay attack (where the attacker does not need to decrypt any message), is directly related to how good and long are the nonces. If they are indeed truly unique (using a counter for example), the probability of successfully replaying a previous message is non-existent (since all future payloads will be different). However, if the nonces are randomly generated (with a true random generation engine), the probability is the same as correctly guessing a nonce. Since the nonces are 128 bits long (16 bytes), this probability can be obtained as follows:

$$P(\text{correct nonce guess}) = \frac{1}{2^{128}} \approx 2.939 \cdot 10^{-39} \quad (4.1)$$

This is, however, somewhat useless if an attacker could be attacking the system constantly (replaying nonce requests indefinitely). The algorithm, has a protection mechanism against that: for every nonce or AEAD message that fails, a timer is increased and the node stops responding to any requests while the timer has not expired.

A better way to describe the chance of a replay attack is then the probability of guess success per time interval. Since the probability of success within 'n' tries follows a binomial distribution $X \sim B(n,p)$, we have that

$$P(\text{k successes in in n tries}) = P(X = k) = \binom{n}{k} p^k (1-p)^{n-k}, \text{ with } p = \frac{1}{2^{128}}. \quad (4.2)$$

Note that this probability does not consider the 32 random bits of the nonce exchange since the attacker could always replay the same nonce request (or alternate between some of them). Since in our case 'k' is always equal to 1,

$$P(1 \text{ success in in n tries}) = P(X = 1) = np(1-p)^{n-1}. \quad (4.3)$$

Our timer increases exponentially for each failed try, then:

$$T' = 2T \cdot a_n \text{ with } a_n = n^2 + n + 1 \quad (4.4)$$

For a specific time interval trying to succeed, the number of tries would be approximately:

$$n(T_{attack}) = \frac{1}{2}\left(\sqrt{\frac{2T_{attack}}{T} - 3} - 1\right) \quad (4.5)$$

Finally, we can calculate the likelihood of accomplishing the attack during this period:

$$P(\text{success within } T_{\text{attack}}) = n(T_{attack}) \cdot \frac{1}{2^{128}}\left(1 - \frac{1}{2^{128}}\right)^{n(T_{attack})-1} \quad (4.6)$$

Several examples with T = 1s are shown in Table 4.

Attack running time	Number of attempts	Probability of success
1 day	~207	$\sim 6.08 \cdot 10^{-37}$
1 month	~1137	$\sim 3.34 \cdot 10^{-36}$
1 year	~3970	$\sim 1.17 \cdot 10^{-35}$

Table 4: Probability of attack success vs. attack running time

* Notes on delayed replay attack:

- The attacker must use specialized tools and techniques to be able to block the impersonated node (A) from receiving the messages sent by the intruder (the sniffing of the wireless traffic can be done much more easily).

- While the attack is being performed, the user may know something is going on since the nodes would not respond to his valid requests.

- The above example does not take into account in-between successful traffic which would reset the exponential timer. If the attacker was waiting for a correct transmission before trying his replay, the number of attempts per timer interval would increase depending on the frequency A and B exchange secure messages.

- The strength of the algorithm is the ability to prevent the attacker to replay a message whenever he wants.

Integrity:

Both the nonce exchange and the AEAD scheme have mechanisms to ensure the payloads transmitted are not tampered in any way. The nonce exchange messages contain hashes (with a PSK) that verify the rest of the payload, and the Xoodyak AEAD algorithm already checks for integrity and authentication of the communication in a similar way (with its validation tag). On top of that there is AES-128 encryption which makes it much more difficult to target specific parts of a payload (for example the associated data, which Xoodyak does not encrypt).

We can test the above stated by executing the basic “Sender” and “Receiver” examples but flipping any bit just before sending or receiving the payload. Then, the internal code of the library detects that and the high-level user gets an error (false) return value. Note that this simulated modification is done before AES encryption so we can test the library. If the bit was to be flipped after encryption, the error would be detected as well, since the payload would not pass any integrity or format checks.

See example in appendix “SecureRF Example (TAMPERED): “Sender” and “Receiver” (commented)”, where a payload bit was flipped (`secure.SECURE_PAYLOAD[0] ^= 0x10`) just before sending the message with (`radio.sendWithRetry(…)`).

5. Budget

Prototype costs:

- **Components**

  The entire component list for the prototypes, including its prices can be found in the appendices ("Bill of materials (BOM)" and "Components budget estimation").
- **Schematic design (included in labor costs)**
- **PCB design (included in labor costs)**
- **PCB prototypes manufacturing**

  25€/5 pieces: 4-layer, 75x25mm, HASL lead-free, silkscreen (both sides).

  70€/100 pieces (same specifications).
- **PCA prototypes assembly (included in labor costs)**

When manufacturing x5 PCB's the total cost for each one was **29.55€** (not including development and assembly costs). However, the costs (components and printed circuit boards) are obviously heavily reduced when manufacturing an increased number of boards, for example it would cost an estimated of **19.21€** if manufacturing 100 units (the cost goes further down for larger quantities).

Labor costs: This work has been carried out during the first semester of 2021 with an estimated dedication of 12 hours per week. Considering 22 working weeks and a junior engineer cost of 11 €/hour, the total working hours cost is about **2900 €** (to be added to the protype costs above).

Similar boards available on the market oscillate between 20€ and 25€ (some of them do not include security processors and sensors built-in). Even including one-time development and occasional labor costs (since almost the whole process can be automated), the manufacturing of large quantities (>1000) makes this prototype reasonable to be sold with sufficient margins (if there was an intentionality).

(*) All specific software used to develop this project both the programming and hardware (KiCad) parts is open source and free of any charge.

6. Conclusions and future development

The **hardware** prototypes passed all the design, manufacture, assembly and testing stages successfully and they have been able to incorporate all the proposed requirements. Although the work focuses on security provided by software, the boards can use the state of the art crypto processor built-in. The prototypes should be able to achieve very low power consumption, since all the components were selected accordingly (including the main microcontroller which cannot be found on any Arduino board on the market), the board runs on low voltage to further reduce power usage.

The **software** developed for both the integration with the Arduino platform and the implementation of a secure algorithm for the RF communications do work as expected. They are a fast way of substantially increasing the security of simple RF communications. Any user can acquire the necessary files publicly available on the internet and use or modify them to build custom projects with latest lightweight security.

High-level integration and open software have great impact on possible use cases of this work. By choosing and implementing compatible software and using already popular libraries, the added benefits of this work and its contributions may be able to reach more people and therefore increase the security of some existing vulnerable devices.

Future development and new features:

- High-level low-power sleep modes implementations (for SAML21)
- Use built-in crypto processor for nonce generation (decrement-only counter)
- Use built-in crypto processor to run standardized authentication algorithm (limited to this board or boards with DS28C16)
- AEAD message fragmentation
- NIST Lightweight Cryptography AEAD finalist implementation (when it comes out)

Bibliography

[1] D. U. (Atmel/Microchip), "The “Three-Legged Stool” of Cryptography," 2016. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8972-CryptoAuth-3-Legged-Stool-Article.pdf.

[2] K. M. (Atmel/Microchiop), "Attack Methods to Steal Digital Secrets," 2015. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-8949-CryptoAuth-Attack-Methods-Steal-Digital-Secrets-WhitePaper.pdf.

[3] "Rolling/hopping codes," [Online]. Available: https://en.wikipedia.org/wiki/Rolling_code.

[4] Atmel (Microchip), "Design and Security Considerations for Passive Immobilizer Systems," 2010. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Article_AC7_Design-and-Security-Considerations.pdf.

[5] Atmel (Microchip), "Designing Next-Generation Key Fobs," 2010. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/Article_AC7_Designing-Next-Generation-Key-Fobs.pdf.

[6] IETF, "RFC4279, Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)," 2005. [Online]. Available: https://tools.ietf.org/html/rfc4279.

[7] IETF, "RFC5487, Pre-Shared Key Cipher Suites for TLS with," 2009. [Online]. Available: https://tools.ietf.org/html/rfc5487.

[8] IETF, "RFC8446, The Transport Layer Security (TLS) Protocol Version 1.3," 2018. [Online]. Available: https://tools.ietf.org/html/rfc8446.

[9] L. K. Grover, "A fast quantum mechanical algorithm for database search," 1996. [Online]. Available: https://arxiv.org/pdf/quant-ph/9605043.pdf.

[10] STMicroelectronics, "STMicroelectronics Resources Application Note (AN5407)," 2020. [Online]. Available: https://www.st.com/resource/en/application_note/dm00660594-optimized-rf-board-layout-for-stm32wl-series-stmicroelectronics.pdf.

[11] Decawave (Qorvo), "Decawave Application notes (APH001)," 2018. [Online]. Available: https://www.decawave.com/wp-content/uploads/2018/10/APS010_DW1000-and-Wireless-Sensor-Networks_v1.1.pdf.

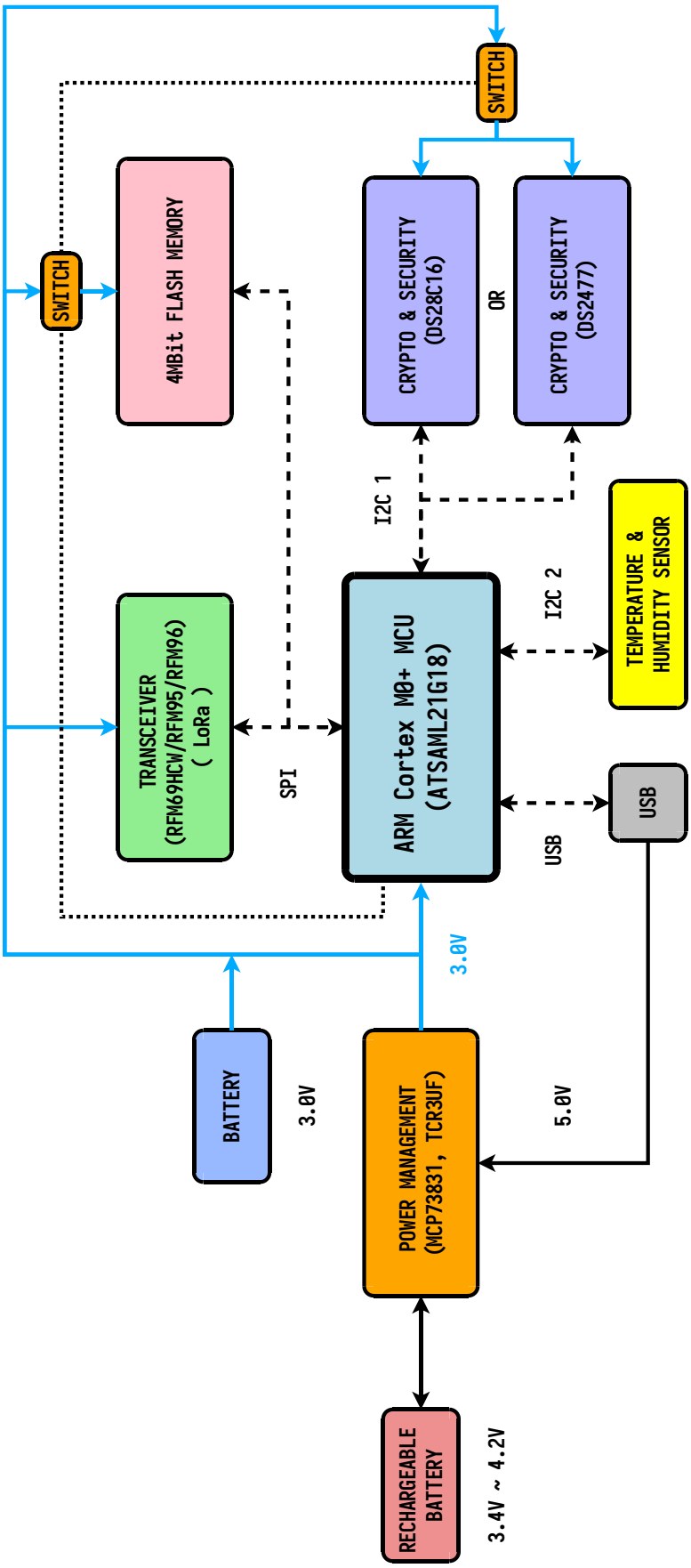
[12] ECS Inc., "CRYSTAL AND OSCILLATOR PRINTED CIRCUIT BOARD DESIGN CONSIDERATIONS," [Online]. Available: https://ecsxtal.com/crystal-and-oscillator-printed-circuit-board-design-considerations.

[13] NIST (COMPUTER SECURITY RESOURCE CENTER), [Online]. Available: https://csrc.nist.gov/glossary/term/nonce.

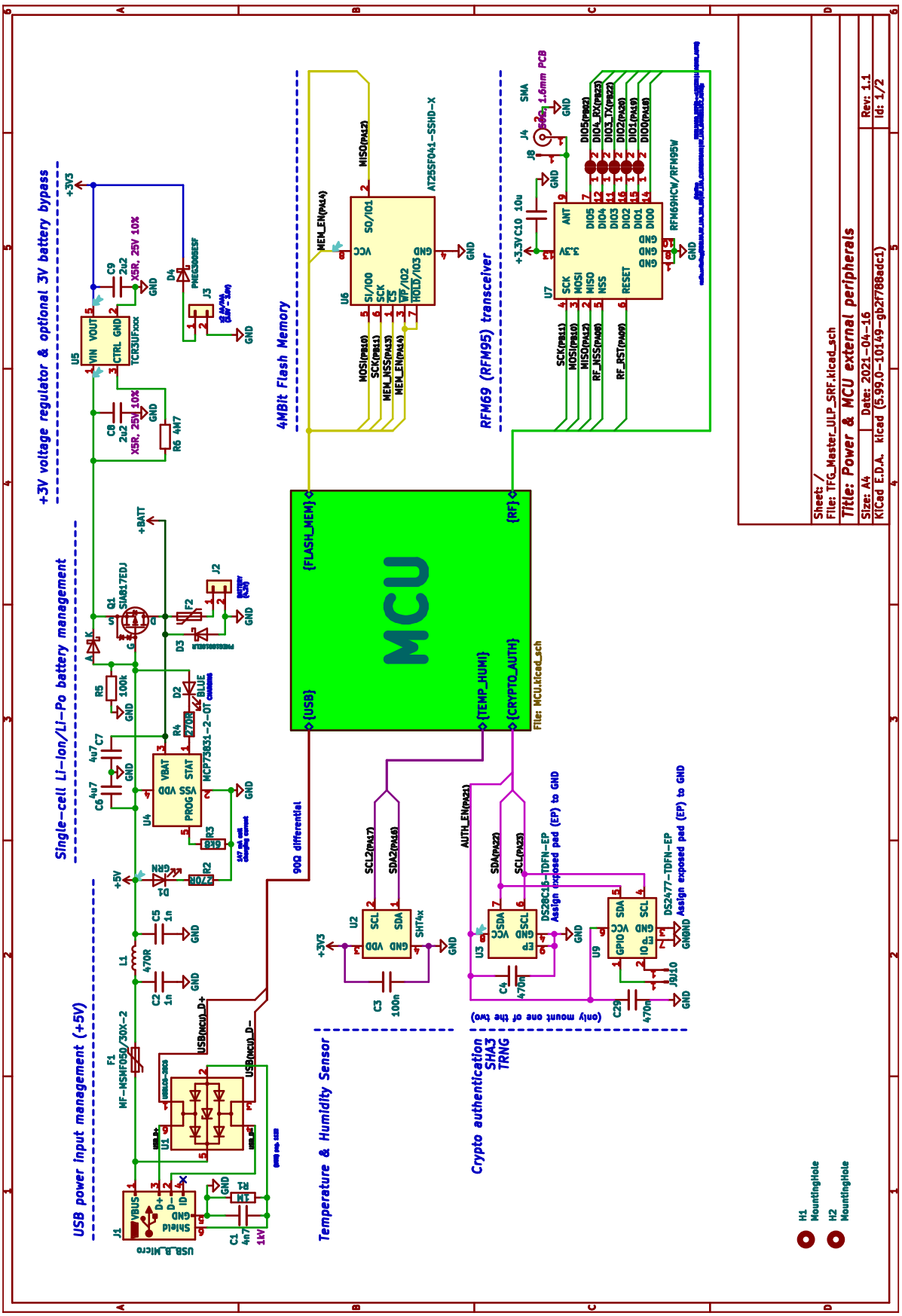
[14] X. a. (. paper), "Xoodyak, a lightweight cryptographic scheme," May 2021. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/xoodyak-spec-final.pdf.

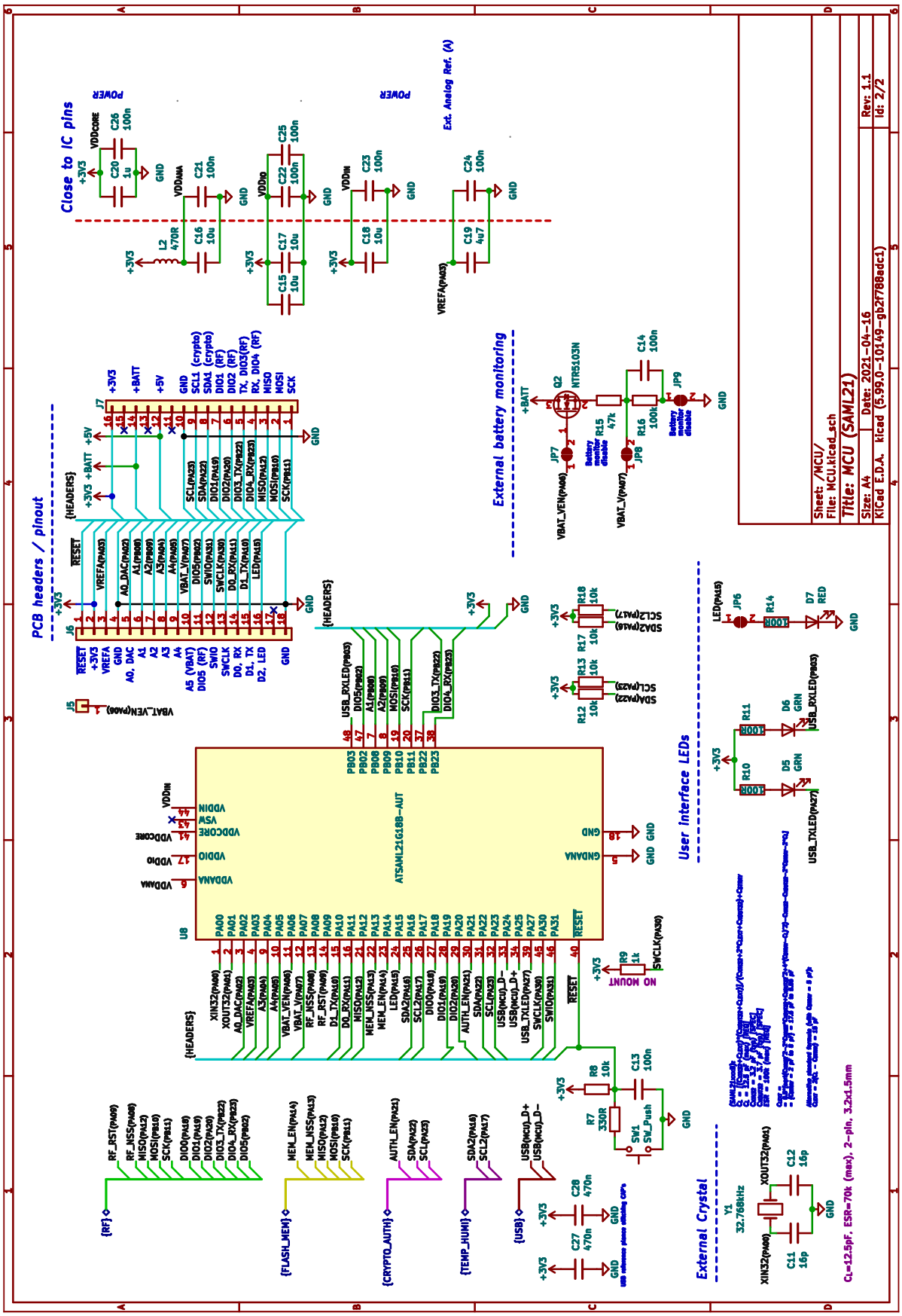
Appendices

Hardware diagram



Schematic


MCU
{FLASH_MEM}
{RF}
{USB}
{TEMP_HUMI}
{CRYPTO_AUTH}
File: MCU.kicad_sch
+3V voltage regulator & optional 3V battery bypass
Single-cell Li-Ion/Li-Po battery management
USB power input management (+5V)
4MBit Flash Memory
RFM69 (RFM95) transceiver
Temperature & Humidity Sensor
Crypto authentication
SHA3
TRNG
(only mount one of the two)
90Ω differential
Sheet: /
File: TFG_Master_ULP_SRF.kicad_sch
Title: Power & MCU external peripherals
Size: A4
Date: 2021-04-16
Rev: 1.1
KiCad E.D.A. kicad (5.99.0-10149-gb2f788adc1)
Id: 1/2


Close to IC pins
PCB headers / pinout
External battery monitoring
User interface LEDs
External Crystal
Ext. Analog Ref. (A)
Sheet: /MCU/
File: MCU.kicad_sch
Title: MCU (SAML21)
Size: A4
Date: 2021-04-16
Rev: 1.1
KiCad E.D.A. kicad (5.99.0-10149-gb2f788adc1)
Id: 2/2
ATSAML21G18B-AUT
C1=12.5pF, ESR=70k (max), 2-pin, 3.2x1.5mm

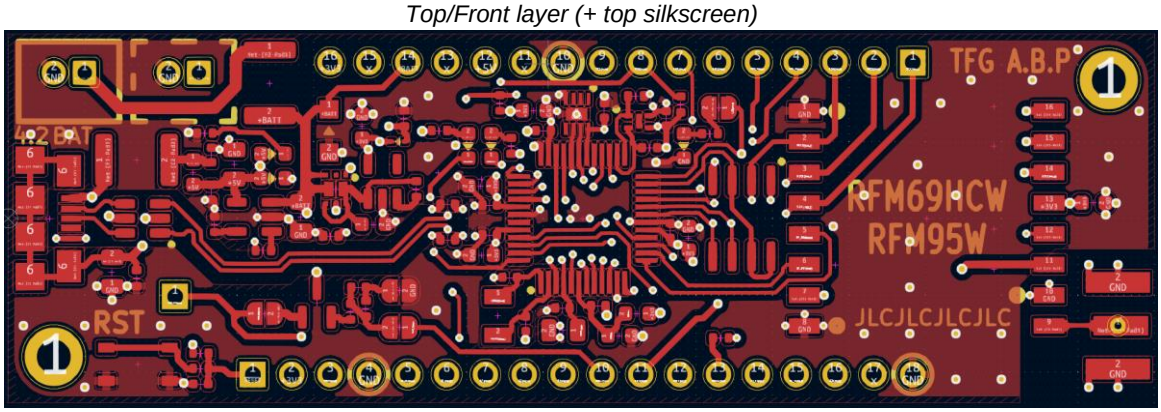
Bill of materials (BOM)

REFERENCE	VALUE	FEATURES	MANUFACTURER REF.	MANUFACTURER	QTY
C1	4n7	1kV	0805B472K102CT	Walsin	1
> C3, C13, C14, C21-C26	100n	X7R, 50V, 10%	GCM155R71H104KE02J	Murata	9
> C4, C29	470n	X7R, 10V, 10%	LMK105B7474KV-F	Taiyo Yuden	2
> C2, C5	1n	X7R, 50V, 5%	04025C102JAT2A	AVX	2
> C6, C7, C19	4u7	X5R, 25V, 10%	CL21A475KAQNNNE	Samsung	3
> C8, C9	2u2	X5R, 25V 10%	GRM188R61E225KA12J	Murata	2
> C10, C15-C18	10u	25V, 20%	GRM188R61E106MA73J	Murata	5
> C11, C12	16p	50V, 5%, C0G	GCM1555C1H160JA16D	Murata	2
C20	1u	25V, 10%, X5R	CL10A105KA8NNNC	Samsung	1
> C27, C28	470n	25V, 10%, X5R	C1005X5R1E474K050BB	TDK	2
> D1, D5, D6	GRN	2.2V, 20mA (0.5@10mA,2.05V)	APT1608SGC	Kingbright	3
D2	BLUE	3.2V, 20mA (0.5@8mA,2.9V)	150060BS75000	Wurth	1
D3	PMEG10010ELR	1A, 0.71V	PMEG10010ELRX	Nexperia	1
D4	PMEG3005ESF	500mA, 30V, 0.31Vf	PMEG3005ESFYL	Nexperia	1
D7	RED	2V, 20mA (0.5@10mA, 1.9V)	APT1608EC	Kingbright	1
> F1, F2	MF-MSMF050/30X-2	0.5A, 30V	MF-MSMF050/30X-2	Bourns	2
J1	USB_B_Micro	Flat SMD	10118192-0001LF	Amphenol	1
> J2, J3	Conn_01x02	2-pin, 2mm pitch	6-440054-2	TE Connectivity	2
J4	SMA	50Ω, 1.6mm PCB	CON-SMA-EDGE-S	RF Solutions	1
J6	Conn_01x18		22-28-4183	Molex	1
J7	Conn_01x16		22-28-4163	Molex	1
> H1, H2, J5, J8-J10, JP1-JP9	MISC_connectors				15
> L1, L2	470R	470R@100MHz, 1A	BLM18PG471SN1D	Murata	2
Q1	SiA817EDJ		SIA817EDJ-T1-GE3	Vishay	1
Q2	NTR5103N		NTR5103NT1G	ON Semiconductor	1
R1	1M	1%, 50V	CRGP0402F1M0	TE Connectivity	1
> R2, R4	270R	5%, 50V	ERJ-2GEJ271X	Panasonic	2
R3	6k8	1%, 50V	SFR01MZPF6801	Rohm	1
> R5, R16	100k	1%, 50V	RC0402FR-07100KL	Yageo	2
R6	4M7	5%, 50V	RC0402JR-074M7L	Yageo	1
R7	330R	5%, 50V	RC0402JR-7D330RL	Yageo	1
> R8, R12, R13, R17, R18	10k	5%, 50V	CRCW040210K0JNEDC	Vishay	5
R9	1k	1%, 50V	RC0402FR-071KL	Yageo	1
> R10, R11, R14	100R	1%, 50V	RC0402FR-07100RL	Yageo	3
R15	47k	1%, 50V	RC0402FR-0747KL	Yageo	1
SW1	SW_Push		PTS815SJM250SMTRLFS	C&K	1
U1	USBLC6-2SC6		USBLC6-2SC6	STMicroelectronics	1
U2	SHT4x		SHT40-AD1B-R3	Sensiron	1
U3	DS28C16-TDFN-EP		DS28C16Q+U	Maxim Integrated	1
U4	MCP73831-2-OT		MCP73831T-2ACI/OT	Microchip	1
U5	TCR3UFxxx		TCR3UF30A,LM(CT	Toshiba	1
U6	AT25SF041-SSHD-X		AT25SF041B-SSHB-B	Adesto Technologies	1
U7	RFM69HCW/RFM95W	868MHz or 915MHz	COM-13909	HOPERF	1
U8	ATSAML21G18B-AUT		ATSAML21G18B-AUT	Microchip	1
U9	DS2477-TDFN-EP		DS2477Q+T	Maxim Integrated	1
Y1	32.768kHz	C_{L}=12.5pF, ESR=70k (max), 2-	X1A000141000312	EPSON	1

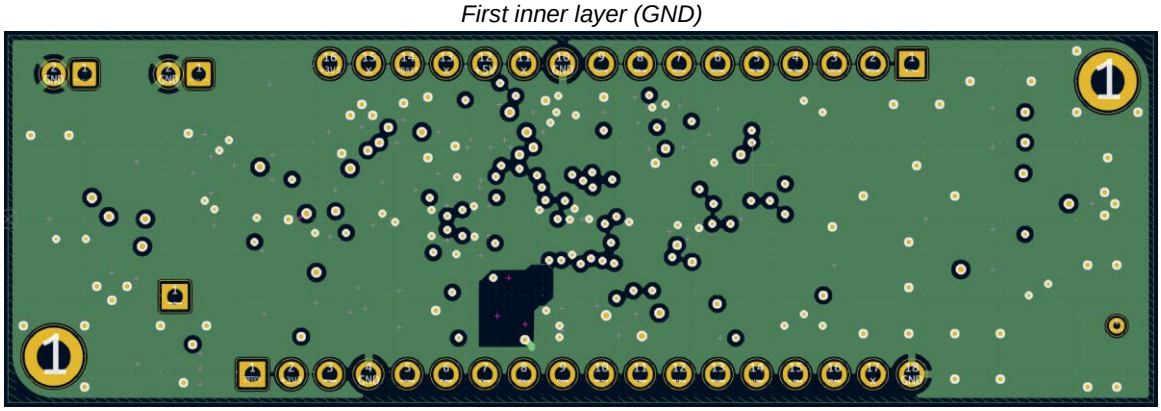
(Some components had to be modified due to the “chip shortage crisis”).

PCB layout stackup

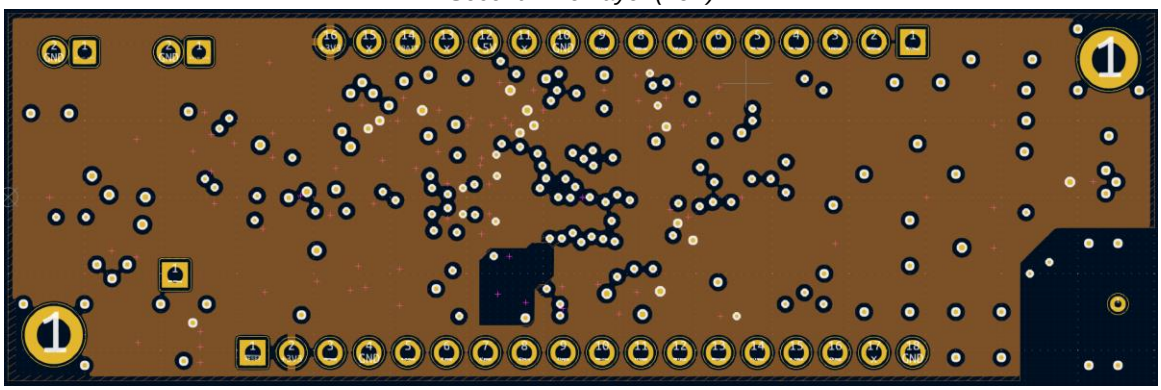
Top/Front layer (+ top silkscreen)



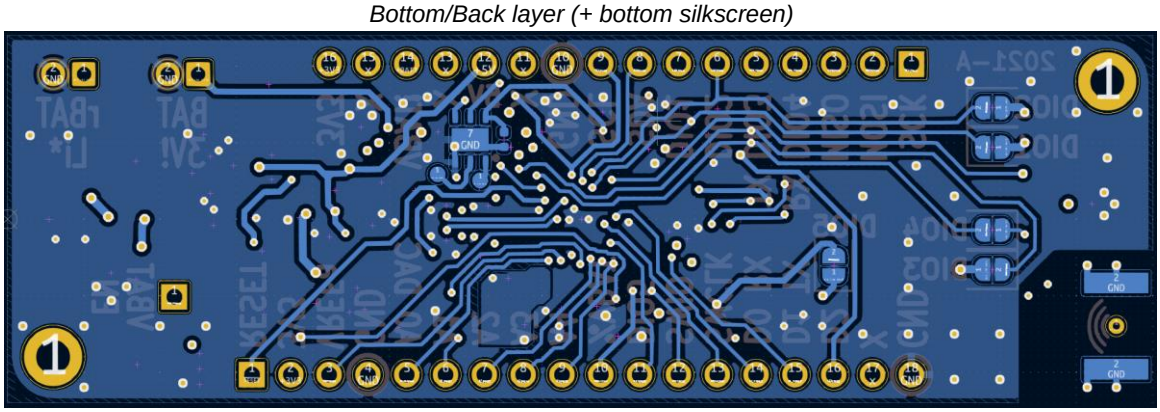
First inner layer (GND)



Second inner layer (+3V)



Bottom/Back layer (+ bottom silkscreen)

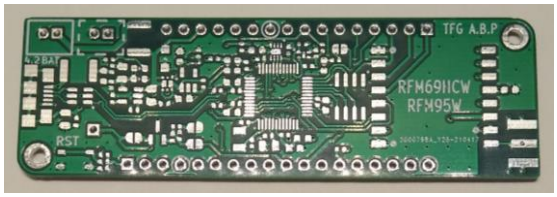


Components budget estimation

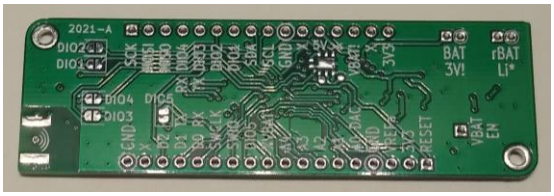
REFERENCE2	QTY	QTY5	QTY100	PRICES (5 units), €	PRICES (100 units), €
0805B472K102CT	1	5	100	0,6	1,19
GCM155R71H104KE02J	9	45	900	1,17	13,5
LMK105B7474KV-F	2	10	200	0,9	13,4
04025C102JAT2A	2	10	200	0,24	3,8
CL21A475KAQNNNE	3	15	300	0,62	9,9
GRM188R61E225KA12J	2	10	200	0,68	10,6
GRM188R61E106MA73J	5	25	500	3,75	48,5
GCM1555C1H160JA16D	2	10	200	0,31	5
CL10A105KA8NNNC	1	5	100	0,43	1,4
C1005X5R1E474K050BB	2	10	200	0,44	6
APT1608SGC	3	15	300	1,37	14,1
150060BS75000	1	5	100	0,68	10,9
PMEG10010ELRX	1	5	100	1,74	12,4
PMEG3005ESFYL	1	5	100	2,12	12,5
APT1608EC	1	5	100	1,23	5,8
MF-MSMF050/30X-2	2	10	200	5,17	78
10118192-0001LF	1	5	100	1,82	24,1
6-440054-2	2	10	200	0,3	4,2
CON-SMA-EDGE-S	1	5	100	7,2	144
22-28-4183	1	5	100	4,75	64,8
22-28-4163	1	5	100	3,81	52,3
	15	75	1500		
BLM18PG471SN1D	2	10	200	0,49	5,8
SIA817EDJ-T1-GE3	1	5	100	2,25	28,7
NTR5103NT1G	1	5	100	0,89	9,1
CRGP0402F1M0	1	5	100	0,43	3
ERJ-2GEJ271X	2	10	200	0,19	1,6
SFR01MZPF6801	1	5	100	0,43	2,9
RC0402FR-07100KL	2	10	200	0,1	0,6
RC0402JR-074M7L	1	5	100	0,43	0,3
RC0402JR-7D330RL	1	5	100	0,43	0,3
CRCW040210K0JNEDC	5	25	500	0,4	3
RC0402FR-071KL	1	5	100	0,43	0,4
RC0402FR-07100RL	3	15	300	0,18	1,2
RC0402FR-0747KL	1	5	100	0,43	0,4
PTS815 SJM 250 SMTR LFS	1	5	100	0,81	11,9
USBLC6-2SC6	1	5	100	2,76	35,2
SHT40-AD1B-R3	1	5	100	10,6	118
DS28C16Q+U	1	5	100	6,1	86,4
MCP73831T-2ACI/OT	1	5	100	2,5	41,9
TCR3UF30A,LM(CT	1	5	100	1,78	18
AT25SF041B-SSHB-B	1	5	100	1,36	20,8
COM-13909	1	5	100	25,2	504
ATSAML21G18B-AUT	1	5	100	20,4	336
DS2477Q+T	1	5	100		
X1A000141000312	1	5	100	4,83	85,2
				€ 122,75	€ 1.851,09
			For one piece:	€ 24,55	€ 18,51

(It is clear that prices of some specific components raised to the “chip shortage crisis” since demand was far higher than supply, therefore those results are based on approximated prices at mid-2021).

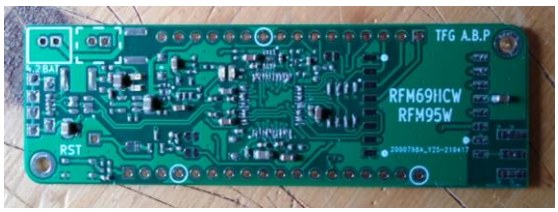
Assembly process pictures



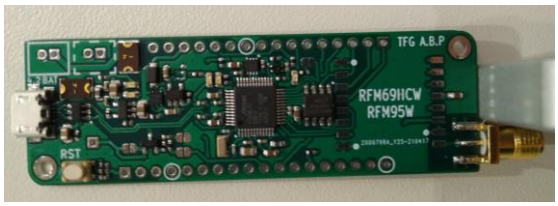
Blank PCB (top)



Blank PCB (bottom)



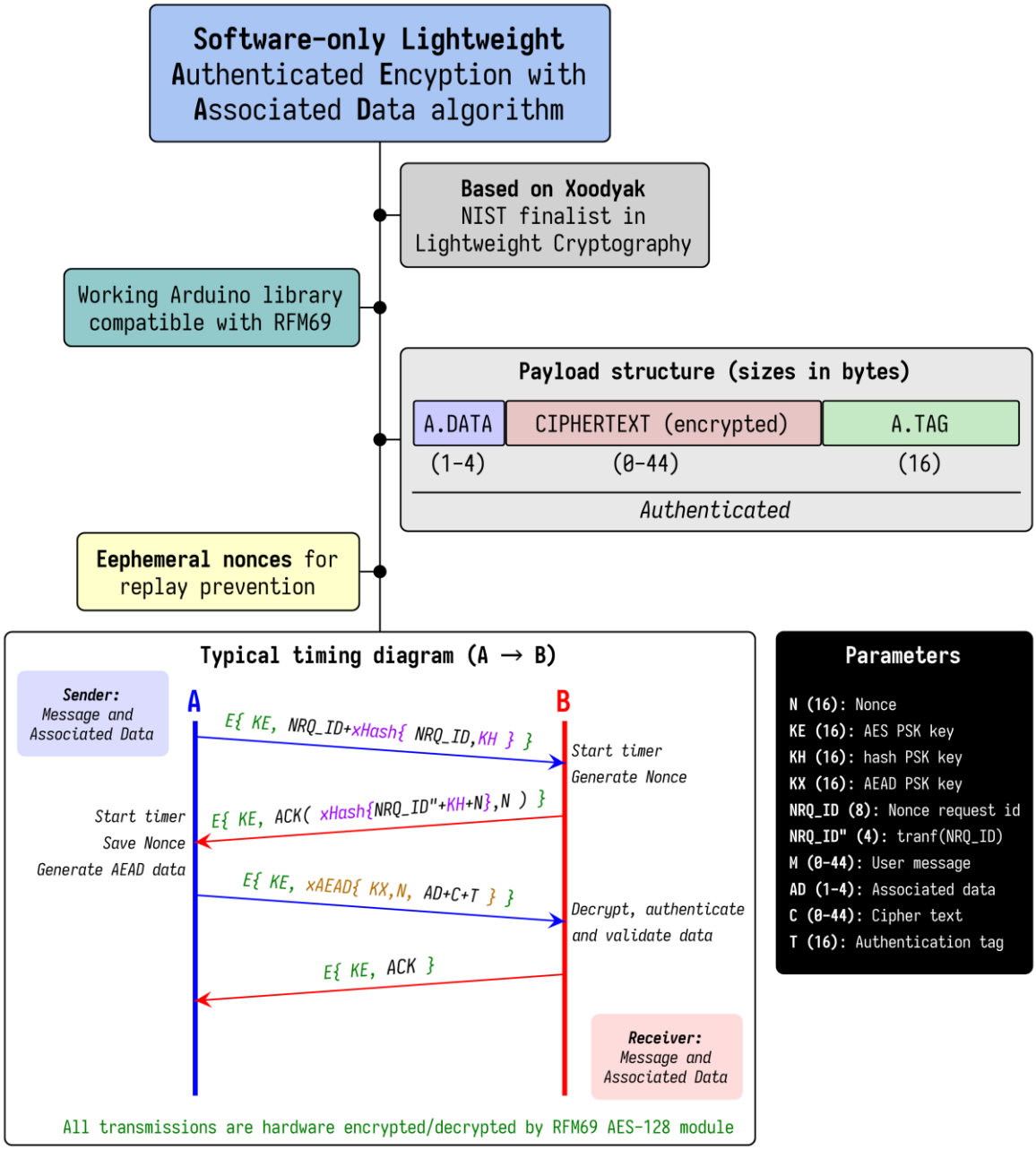
Component placement (small → large)



SMD components soldered

Completed board (top)

Secure algorithm summary diagram (SO)



SecureRF Example: “Sender” and “Receiver” (commented)

A (sender, ID: 100)	B (receiver, ID: 200)
[100]: SENDING NONCE REQUEST (RECEIVER:200) -> (12){ **4E:52:45:51:9D:E8:F6:97:BA:46:AA:12** } ^^NREQNID^^|^^NREQRID^^|^^^NREQH^^^ *N R E Q ▯ è ö — º F ª*	[200]: RECEIVED NEW DATA (SENDER:100) -> (12){ 4E:52:45:51:9D:E8:F6:97:BA:46:AA:12 } [200]: VALID NONCE REQUEST RECEIVED [200]: SENDING NEW NONCE -> (20){ **13:76:2C:6B:5D:10:7C:09:D2:63:0A:CD:FD:C7:** ^^^NRESH^^^|^^^^^^^^^^NONCE^^^^^^^^^^^^^ *v , k] | Ò c Í ý Ç* **0F:B7:F7:FB:FE:CD** } ^^^^^^NONCE^^^^^^ *· ÷ û þ Í*
[100]: VALID NONCE RECEIVED -> (20){ 13:76:2C:6B:5D:10:7C:09:D2:63:0A:CD:FD:C7: 0F:B7:F7:FB:FE:CD } [100]: PLAIN DATA THAT WILL BE SENT: -> ASSOCIATED (2): ON -> MESSAGE (16): Change LED state [100]: AEAD DATA GENERATED SUCCESSFULLY [100]: SENDING AEAD PAYLOAD -> (35){ **90:4F:4E:3C:84:36:C8:79:D7:74:F3:40:35:42:** PI|^AD^^|^^^^^^^^^^^CIPHERTEXT^^^^^^^^^^^ *▯ O N < „ 6 È y × t ó @ 5 B* **44:60:D3:77:CA:2B:08:17:FB:46:C9:EB:C0:6E:8A:E3:** ^^CIPHERTEXT^^|^^^^^^^AUTHENTICATION TAG^^^^^^^ *D ` Ó w Ê + û F É ë À n Š ã* **BA:0A:44:6F:20** } ^^^AUTH.TAG^^^ *º D o*	[200]: RECEIVED NEW DATA (SENDER:100) -> (35){ 90:4F:4E:3C:84:36:C8:79:D7:74:F3:40:35:42: 44:60:D3:77:CA:2B:08:17:FB:46:C9:EB:C0:6E:8A:E3: BA:0A:44:6F:20 } [200]: RECEIVED VALID AEAD DATA: -> ASSOCIATED (2): ON -> MESSAGE (16): Change LED state
[100]: SECURE AEAD DATA RECEIVED BY REMOTE NODE (or not)	

* AES-128 decrypted payload contents.

*** ASCII character conversion (unit8_t → char).**

SecureRF Example (TAMPERED): “Sender” and “Receiver” (commented)

A (sender, ID: 100)	B (receiver, ID: 200)
[100]: SENDING NONCE REQUEST (RECEIVER:200) -> (12){ 4E:52:45:51:E2:1C:1F:8E:86:73:00:54 }	
	[200]: RECEIVED NEW DATA (SENDER:100) -> (12){ 4E:52:45:51:E2:1C:1F:8E:86:73:00:54 } [200]: VALID NONCE REQUEST RECEIVED [200]: SENDING NEW NONCE -> (20){ 6C:CD:6A:85:FA:D3:AC:F0:90:F1:37:22:AE:B0: 8E:06:AA:7F:9C:5C }
[100]: VALID NONCE RECEIVED -> (20){ 6C:CD:6A:85:FA:D3:AC:F0:90:F1:37:22:AE:B0: 8E:06:AA:7F:9C:5C } [100]: PLAIN DATA THAT WILL BE SENT: -> ASSOCIATED (2): ON -> MESSAGE (16): Change LED state [100]: AEAD DATA GENERATED SUCCESSFULLY [100]: SENDING AEAD PAYLOAD -> (35){ **90:4F:4E:B3:F7:55:E0:CF:68:4C:26:74:43:07:** ^^ Byte which will contain flipped bit **4E:03:7B:09:0C:40:BD:F0:46:65:44:C5:3E:30:33:FD: CB:8E:92:58:DD** }	
	[200]: RECEIVED NEW DATA (SENDER:100) -> (35){ **80:4F:4E:B3:F7:55:E0:CF:68:4C:26:74:43:07:** ^^ Byte containing flipped bit **4E:03:7B:09:0C:40:BD:F0:46:65:44:C5:3E:30:33:FD: CB:8E:92:58:DD** } **[200]: AEAD DATA ERROR !** ^^ TAMPERING DETECTED ^^ * (will not send ACK) *
* (RFM69 retrying since ACK was not received) * * (retries (2) and timeouts can be configured) *	
	* (extra data received is ignored) * [200]: RECEIVED NEW DATA (SENDER:100) -> (35){ 80:4F:4E:B3:F7:55:E0:CF:68:4C:26: 74:43:07:4E:03:7B:09:0C:40:BD:F0:46:65:44:C5:3E: 30:33:FD:CB:8E:92:58:DD } [200]: RECEIVED NEW DATA (SENDER:100) -> (35){ 80:4F:4E:B3:F7:55:E0:CF:68:4C:26: 74:43:07:4E:03:7B:09:0C:40:BD:F0:46:65:44:C5:3E: 30:33:FD:CB:8E:92:58:DD }

* AES-128 decrypted payload contents.

Glossary

Ciphersuite: set of algorithms that help secure a network connection, they usually contain a key exchange algorithm, an encryption algorithm, and a message authentication code (MAC) algorithm.

Non-repudiation: is a property achieved through cryptographic methods which prevents an individual or entity from denying having performed a particular action related to data (such as mechanisms for non-rejection or authority (origin); for proof of obligation, intent, or commitment; or for proof of ownership). On public key cryptography, it can be achieved by a trusted third party.

Security level: measure of the strength that a cryptographic primitive (cipher or hash function) achieves. It is usually expressed in bits, where n-bit security means that the attacker would have to perform 2^n operations to break it.

Stackup: arrangement of copper layers and insulating layers of a printed circuit board. Power planes, pours, trace directions and signal types must be taken into account together with the stackup design. The stackup directly affects controlled impedance traces, crosstalk between traces and interplane capacitance. It can be critical for good electromagnetic compatibility.