

SLIDE 1

Good morning everyone, I'm Luca Tomei and I am going to show the presentation of my thesis work whose title is "Design and proof-of-concept of the Integration of an Electric Scooter with a Wheelchair: from the Mechanical Design to the Firmware Upgrade".

SLIDE 2

The aim of the project is to extend the use of an electric scooter to people with disabilities so that the same family can share the same means of motion. It was decided to not make any changes to neither the wheelchair nor the electric scooter, making as easy as possible for the two parts to be coupled and used together. But we have found two main problems: First. we must allow the electric scooter to start from a standstill and Second. we need to create the coupling of the two components.

Since the firmware of an electric scooter is owned by the manufacturer and is not available, it was decided to reverse engineer it by monitoring the packets exchanged between the smartphone and the vehicle.

As will be shown, this leads to a serious security problem in this IT product and after extracting and modifying the encrypted firmware, we made a steel bracket that allows attachment to a wheelchair. Finally, road tests were carried out to analyse the behavior of this software modification in order to make improvements and obtain data that would make driving safe without causing damage to people, battery or other components.



Design and proof-of-concept of the Integration of an Electric Scooter with a Wheelchair: from the Mechanical Design to the Firmware Upgrade

Luca Tomei

Advisor: Andrea Vitaletti

October 20, 2021

Master of Science in Engineering in Computer Science

Thesis Focus

IDEA:

- Use the **same tool** for both able-bodied and disabled people in the same family.
- Realization of a working and economical prototype without the modification of the **Wheelchair** and **Electric Scooter**.

Main Contribution:

- Firmware **Reverse Engineering**
- Custom Firmware Injection
- **Realization** and creation of **metal support** to couple Electric Scooter to the Wheelchair.
- Real-Life **Experimental Results**



SLIDE 3

The popularity of electric scooters has increased in recent years. However, the structure of such a vehicle imposes several constraints that apparently cannot be overcome by disabled people: for example, they can only be ridden by who can stand up and there is no possibility to start from a standstill as a small push with the feet is required to start the ride. Finally, electric scooters have little modularity, that is they are difficult to adapt to various uses.



SLIDE 4

To overcome this problem, this project provides a different use of the wheelchair and electric scooter and, as is shown in this slide by a price comparison, it is possible to add traction to a normal wheelchair at a lower cost than a motorised one.

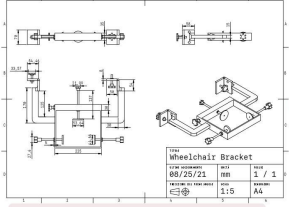


SLIDE 5


Very quickly I want to show a representative scheme that summarises the idea of my thesis project, which consists of the design and implementation of a metal support and the development of a custom firmware that allows the driveability of the electric scooter coupled to the wheelchair.

[Intro](#) [Idea](#) [Project](#) [Protocol](#) [Firmware](#) [Conclusions](#) [Background](#) [Realization](#)

Idea Development - Background




Part	Material	Qty	Unit
Bracket	Alu	2	1/2
Knob	Alu	3	1/2
Vibration Damper	Alu	3	1/2



```
def motor_start_speed(self, knob):
    ret = []
    val = int(knob * 300)
    val = val - (val % 32)
    assert val < 12, 'bit length overflow'
    sig = (0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00)
    ofs = self.firmware.get(self, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00)
    pre = self.data[ofs:ofs + 4]
    post = bytes(self.aml['0x00', 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00])
    self.data[ofs:ofs + 4] = post
    ret.append(ofs, pre, post)
    ofs += 4

    pre, post = self.data[ofs:ofs + 11], bytearray([0x00, 0x00])
    self.data[ofs:ofs + 2] = post
    ret.append(ofs, pre, post)
    return ret
```



5 / 20

Luca Tomei

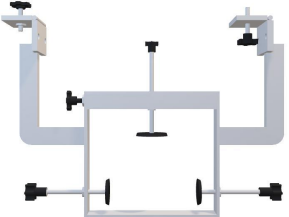
M.Sc. Engineering in Computer Science

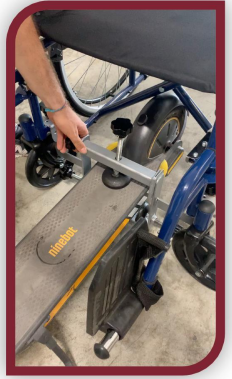
SLIDE 6

Here we can see how from the Autocad project we go to the concrete realisation of the bracket, which in the figure on the right is hooked to the wheelchair and installed near the rear wheels. This support, consisting of 3 knobs and 3 vibration dampers, allows the front wheels of the wheelchair to be raised and prevents longitudinal and transversal movements resulting from towing.

[Intro](#) [Idea](#) [Project](#) [Protocol](#) [Firmware](#) [Conclusions](#) [Background](#) [Realization](#)

Idea Development – Realization





6 / 20

Luca Tomei

M.Sc. Engineering in Computer Science

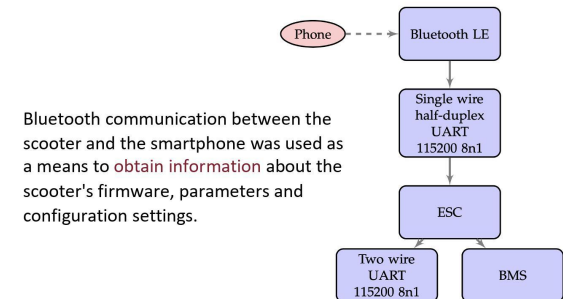
SLIDE 7

Now let's talk about the steps that made it possible to communicate with the vehicle and modify its parameters.

SLIDE 8

The process of interaction between the electric scooter and the smartphone takes place through Bluetooth connection as shown in the figure. This protocol defines the general format by which the Ninebot and Xiaomi electronic control system achieves multi-node communication via physical and Bluetooth serial ports. To use serial port communication must be used a baud rate of 1 hundred and 15 thousand 2 hundred, 8-byte word without verification and 1 stop bit. Information retrieval, control and parameter modification can be achieved through a read-write operation to the "memory control table" which is stored in the controller.

IMPLEMENTATION



SLIDE 9

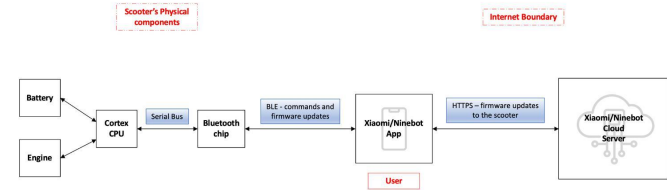
Extrapolating data from the GET/POST requests made by the application of the electric scooter connected to the phone, we realise that this technology is used as a means of issuing commands and transmitting various types of updates. Knowing how the two entities communicate is essential to proceed to modify the firmware parameters.

In fact, the figure below describes the various connection phases that the electric scooter performs while communicating with the parent cloud server. The user's application sends requests to the server, which immediately responds with updates for the vehicle. All this takes place under a Bluetooth connection between the smartphone and the chip in the electric scooter in order to issue commands to the control unit and the engine.

E-Scooter - App Communication Diagram

Block diagram of the **communication** between the processor of the **scooter** and its official **application**.

Bluetooth Low Energy as a **channel** for communicating commands, firmware updates and changes in settings.



SLIDE 10

The original firmware cannot handle towing a wheelchair due to various limitations imposed by the manufacturer. For example, it is necessary to provide a standing start, increased acceleration, and limit speed in order to be used by people with disabilities.

Here you can see the block diagram that has been followed to perform a firmware modification. In particular, after obtaining the original firmware from the analysis of HTTPS packets exchanged with the smartphone, it is necessary to decrypt the binary file then proceed to change the parameters of interest. Finally, we need to encrypt the binary firmware and flash it on the memory of the electric scooter.

E-scooter - Firmware Sniffing

Checking the scooter firmware via communication with the smartphone application allows you to obtain the **binary file** of the electric scooter firmware in clear text.



```

Get updated firmware:
{
  "code": 1,
  "data": {
    "is_test": false,
    "ctrl": {
      "task_version": "355",
      "version_content": "1. Optimize firmware compatibility
2. Improve firmware stability",
      "bin_content":
      "8fe3569337a633a58c7c399a495b2378e342984858e6861d24de453a8f68969095181bee2
138262796775583a83887137493708a6c4f75e1649c1e0827e8382252736773868d8e02
a048f4699adac4a9b11c3058f7a0823962a4d6c131774e7a8007a63376d6a7a75ac798
803fe44a9e3820867835a9e2e7e1a4ca24f3a6e55749a4a184587f5e15a3a7c7469704a05
8a45414a2a0a4c49874775448a3a53a96a597c743a4a4a2a5c5537578c74887a0a929
5177408fede47b777b36a8d2c310878a8a64c497c73f123a7bda2a078ce086a35817775c
a2d6d564a0a933a6a0b0c9744",
      "verify_code": "03650446",
      "firmware_type": 1
    },
    "bin": false,
    "bin": false,
    "forced_status": false,
    "forced_content": "",
    "special_version_status": false
  },
  "msg": "Mission accomplished.",
  "tip": ""
}
  
```

SLIDE 11

A first careful analysis of the binary file obtained from packet sniffing was performed with a decompiler that makes readable the values present in that data structure. Subsequently, various functions were implemented to encode and decode firmware of Xiaomi and Ninebot electric scooters. The first python function shown in this slide takes as input the "sniffed" file and returns as output a decrypted one that will be used to modify the parameters. The second function will be used last to make compatible the modified file with the original one, before flashing on the electric scooter.

Intro Idea Project Protocol Firmware Conclusions Encryption/Decryption CFW FW Generator

E-scooter - Firmware Encryption/Decryption

```
def encrypt(self, data):
    data = self.BinaryUtilities_0B3.pad(data)
    assert len(data) % 8 == 0, 'data must be 8 byte aligned!'
    res = bytearray()
    for i in range(0, len(data), 8):
        ct = self.BinaryUtilities_0B3
            .ecb_encrypt(self.BinaryUtilities_0B3
                .xor(self.iv, data[i:i+8]), self.key)
        res += ct
        self.iv = ct
        self.offset += 8
    if (self.offset % 1024) == 0: self.update_key()
    return res
```

The firmware obtained from the packet analysis must be **decrypted** to allow the modification of its parameters.

```
def decrypt(self, data):
    assert len(data) % 8 == 0, 'data must be 8 byte aligned!'
    res = bytearray()
    for i in range(0, len(data), 8):
        ct = data[i:i+8]
        res = self.BinaryUtilities_0B3
            .xor(self.iv, self.BinaryUtilities_0B3
                .ecb_decrypt(ct, self.key))
        self.iv = ct
        self.offset += 8
    if (self.offset % 1024) == 0:
        self.update_key()
    return self.BinaryUtilities_0B3.unpad(res)
```

Once the parameters of the custom firmware have been modified, it must be **encrypted** and then flashed on the electric scooter.

11 / 20 Luca Tomei M.Sc. Engineering in Computer Science

SLIDE 12

After decoding the binary file, we can proceed with modifying the parameters of interest like acceleration, KERS, starting speed, cruise control, motor power constant and top speed.

As you can see from the pieces of code shown, Python functions take various arguments as input, allowing you to modify those parameters according to your inclinations and needs.

Each function uses the auxiliary method "GetPattern" which is used to obtain the pattern in the binary structure of the data to be modified.

It takes the bytearray data structure as input and searches through it, returning the index to be used to modify the desired value.

Intro Idea Project Protocol Firmware Conclusions Encryption/Decryption CFW FW Generator

E-scooter - Modify Firmware Parameters

Set **KERS** Minimum Speed

```
def kers_min_speed(self, kmh):
    val = struct.pack('d', int(kmh * 390))
    sig = [0x25, 0x00, 0x40, 0x00, 0x00, 0x24, 0x00, 0x00]
    ofs = self.Firmware_Utils_0B3.GetPattern(self.data, sig)
    pre, post = self.Firmware_Utils_0B3
        .Find(self.data, ofs, 4, val, MOVW_T3_1MM)
    return (ofs, pre, post)
```

Starting From **Standstill**

```
def motor_start_speed(self, kmh):
    ret = []
    val = int(kmh * 390)
    val = val - (val % 16)
    assert val.bit_length() <= 12, 'bit length overflow'
    sig = [0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00]
    ofs = self.Firmware_Utils_0B3.GetPattern(self.data, sig)
    pre = self.data[ofs:ofs + 4]
    postbytes = self.data[ofs + 4:ofs + 1024]
    self.data[ofs:ofs + 4] = post
    ret.append((ofs, pre, post))
    ofs = 4
    pre, post = self.data[ofs:ofs + 1], bytearray([0x00, 0x00])
    self.data[ofs:ofs + 2] = post
    ret.append((ofs, pre, post))
    return ret
```

Edit **Accelerator** Curve

```
def throttle_alp(self):
    sig = [0x00, 0x00, 0x25, 0x40, 0x00, 0x24, 0x00, 0x00, 0x00, 0x40, 0x24, 0x00]
    ofs = self.Firmware_Utils_0B3.GetPattern(self.data, sig)
    pre, post = self.data[ofs:ofs + 1], bytearray([0x00, 0x24])
    self.data[ofs:ofs + 2] = post
    return (ofs, pre, post)
```

12 / 20 Luca Tomei M.Sc. Engineering in Computer Science

Intro Idea Project Protocol Firmware Conclusions Encryption/Decryption CFW FW Generator

E-scooter - GUI for automatic firmware generation

Automated custom firmware generator



[Intro](#) [Idea](#) [Project](#) [Protocol](#) [Firmware](#) [Conclusions](#)

E-scooter + Wheelchair – Experimental Results

14 / 20 Luca Tomei M.Sc. Engineering in Computer Science

SLIDE 15

During the various road tests, we have collected data and then analysed and transformed using the Python matplotlib library to generate static and dynamic graphs regarding distance travelled, speed and slope of the ground. In order to analyse the differences in behaviour between the two firmwares, we tried to follow the same route for both test drives, which were carried out on a mostly flat road at a speed as constant as possible.

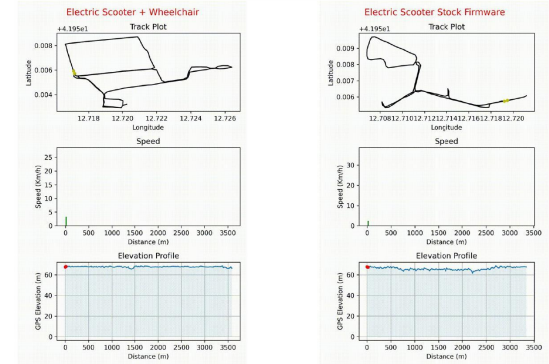
SLIDE 16

These figures show the trend in time of current and temperature. The current reaches positive peaks during acceleration and negative peaks during braking when the kinetic energy recovery system takes over. When the electric scooter is used with a wheelchair, the amplitude of these peaks is greater, so more energy will be recovered during braking. It can be seen that the temperature is in the range of 28 to 34 Celsius degrees, which is a lower range than when used without a wheelchair. Operating at a lower temperature will extend the battery life and the kilometres travelled.

In the second figure we can see that the negative peaks are smaller in amplitude due to the reduced mass. For the temperature we can observe that, as time passes, it is close to 40 degrees centigrade. This significant difference is due to the different law that regulates acceleration, a linear law that provides higher and more extended current levels in time.

Experimental Results

Distance Traveled during Road Test



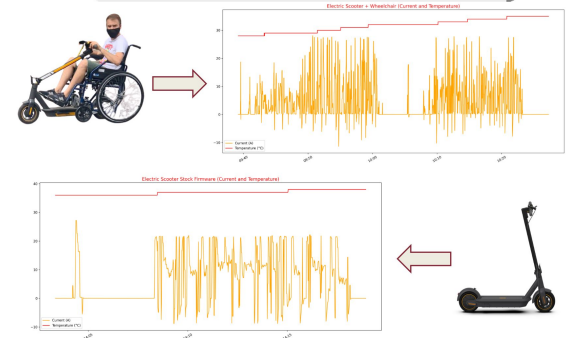
15 / 20

Luca Tomei

M.Sc. Engineering in Computer Science

Experimental Results

Current and Temperature Over Time



16 / 20

Luca Tomei

M.Sc. Engineering in Computer Science

SLIDE 17

The following graphs show the trend of battery discharge, remaining distance and available voltage over time.

Obviously these three parameters depend directly on the weight of towed mass, in fact in the first case the discharge of the battery has a much steeper trend, the energy demand is very high but is compensated by the recovery system.

Instead in the second case we notice that the standard firmware guarantees a more contained use of the available energy and the energy recovery peaks are lower in amplitude and frequency.

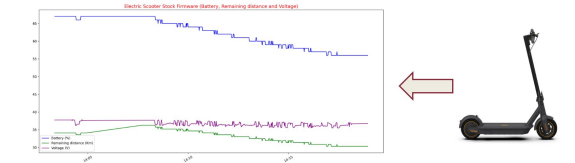
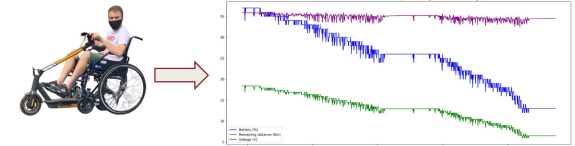
SLIDE 18

Regarding the power delivered by the engine, to ensure greater acceleration and starting from a standstill, the custom firmware generates peaks that are much higher than in the case of the stock one, who reaches the average power level of 780 watts compared to the 980 watts obtained when towing the wheelchair.

These values are related only to one of the three driving modes of the electric scooter, the sport mode, since during the customization of the firmware it was decided to leave the other two modes unchanged to ensure a longer battery life.

Experimental Results

Battery, Remaining Distance and Voltage Over Time



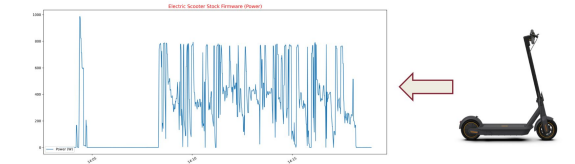
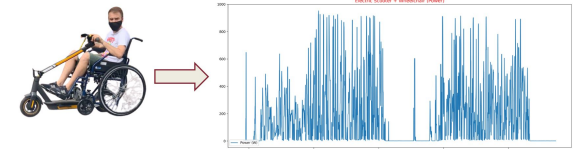
17 / 20

Luca Tomei

M.Sc. Engineering in Computer Science

Experimental Results

Power Over Time



18 / 20

Luca Tomei

M.Sc. Engineering in Computer Science

SLIDE 19

The main objective of this thesis was to demonstrate and realise the feasibility of having an economic solution that allow the use of the same means of motion by both able-bodied people and people with disabilities. Moreover, this result has been achieved without modifying in any way the structure of the electric scooter and the wheelchair. In fact, to allow the use of the electric scooter in this new field of use, it is only necessary the modification of the software and the realisation of the mechanical coupling between the two parts.

SLIDE 20

Thank you for your attention.

Conclusions - Summary

Thesis Focus:

Modify the stock firmware of the electric scooter in order to make it usable as a traction and movement tool for disabled people.

Contributions:

- **Analysis** and modification of the firmware of an electric scooter
- **Implementation** of software that automates the process of generating custom firmware
- **Realization** of metal support to ensure mechanical coupling between electric scooter and wheelchair
- Experimental **results** that show the feasibility of the project

Thesis, slides, code and videos are all available at:



Thank you!

Luca Tomei

tomei.1759275@studenti.uniroma1.it