SAPIENZA
UNIVERSITÀ DI ROMA

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, 1759275

Advisor: **Dr. 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.

Electric Scooter - Disability Problem

Only able-bodied people

No standing start

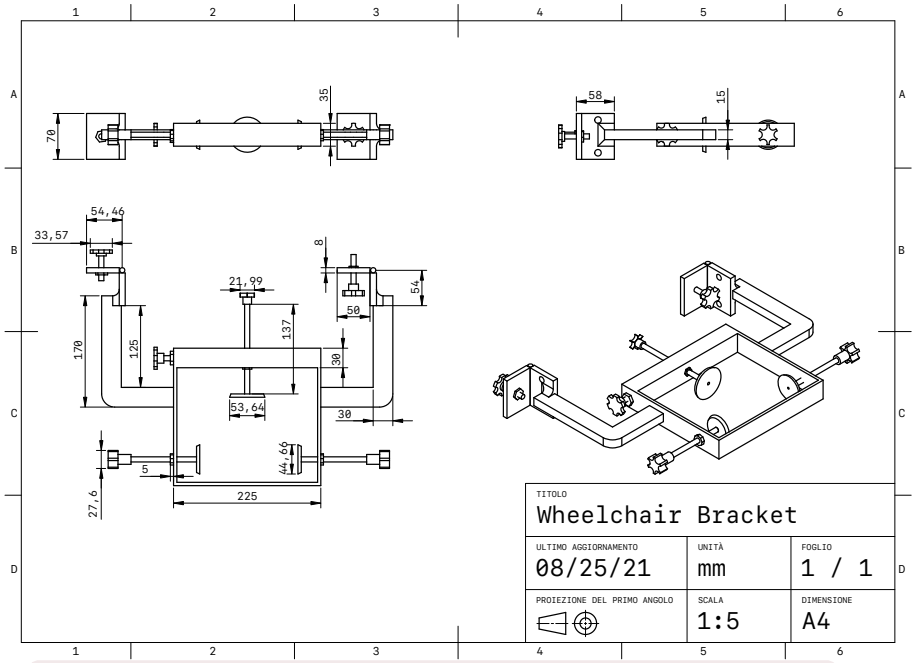
Poor modularity

Solutions - Price Comparison

want

need

Idea Development - Background

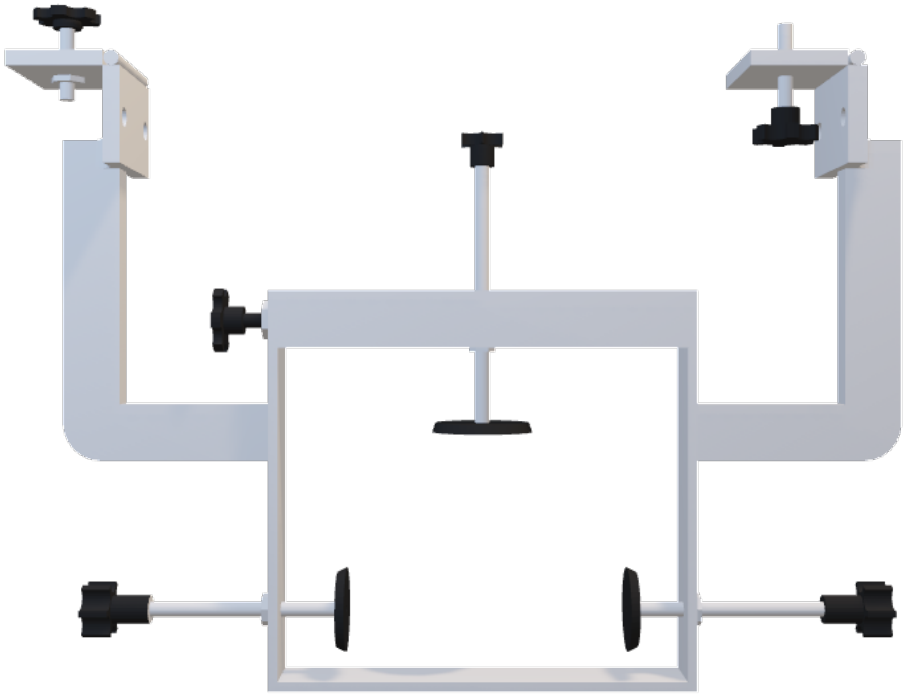



```python
def motor_start_speed(self, kmh):
    ret = []
    val = int(kmh * 390)
    val = val - (val % 16)
    assert val.bit_length() <= 12, 'bit length overflow'
    sig = [0x4B, 0x01, None, None, 0x48, 0x00, None,
           None, None, 0xB6, 0xF5, 0xC5, 0x6F, 0x0D, 0xDB]
    ofs = self.Firmware_Utils_OBJ.GetPattern(self.data,sig)+9
    pre = self.data[ofs:ofs + 4]
    post=bytes(self.ks.asm('CMP.W R6, #{:n}'.format(val))[0])
    self.data[ofs:ofs + 4] = post
    ret.append((ofs, pre, post))
    ofs += 4

    pre,post = self.data[ofs:ofs + 1],bytearray((0x0D, 0xDB))
    self.data[ofs:ofs + 2] = post
    ret.append((ofs, pre, post))
    return ret
```

Wheelchair Bracket

Project Code

Idea Development – Realization

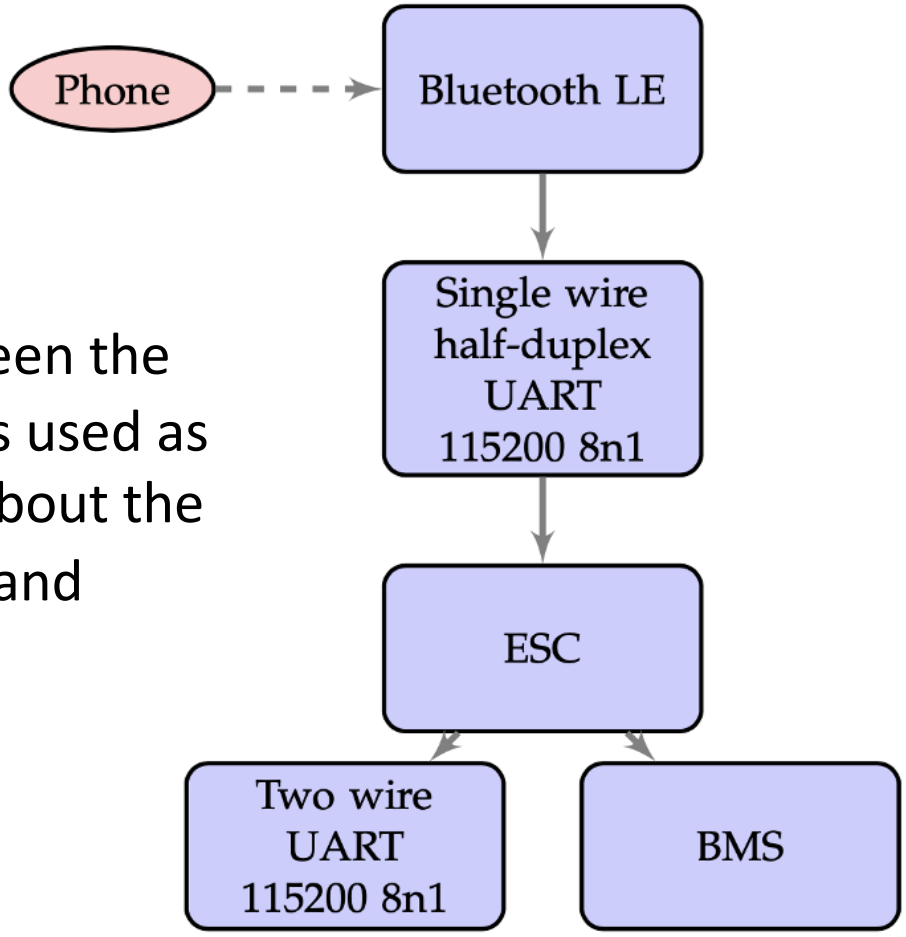


E-scooter – Firmware Modification

IMPLEMENTATION

E-scooter - Communication Protocol

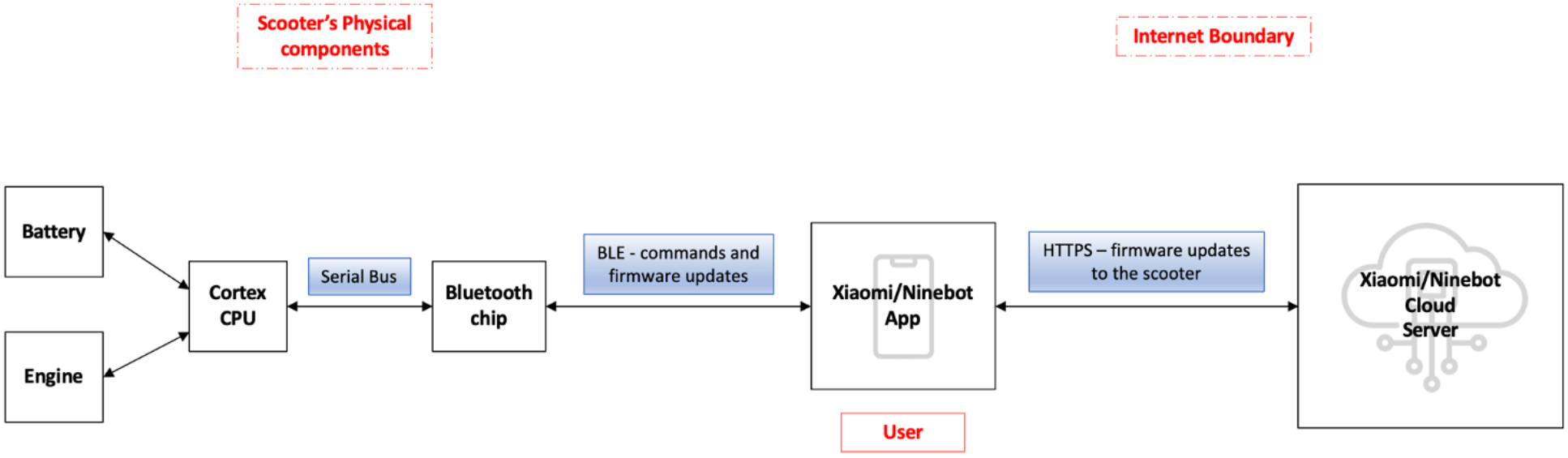
Bluetooth communication between the scooter and the smartphone was used as a means to obtain information about the scooter's firmware, parameters and configuration settings.



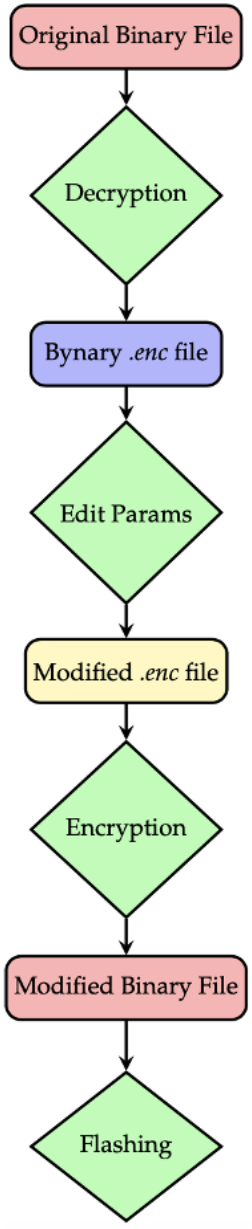
E-Scooter - App Communication Diagram

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.



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": {
      "last_version": "355",
      "version_content": "1. Optimize firmware compatibility
   2. Improve firmware stability",
      "bin_content":
"0fe35d59357a4e53c8c8d7c399a495b2370e3d2984850e6861d24de9453ad0f60969059101beee2
b3828c2796775583da8300f5174f037c90aac0f4f5ed1649c61ed62fe838b22b273677306b0aeb82
a6b4df49b9adac61e9b11cc3d5807ed8823962b4c86ec131f7f4ef5a8d0d7a633fb6047e75aec706
0b3fe434b9d302d86d7935a962be7e1b4ca42df3e9e5579494a81b6507fbe315e3afd7d6976b4065
8a4b5414a26ad0a4c89e07b477b5448ca3ee633e905d97df2e34b04a2ec5d33757027af00fab9029
5177469dfe4fdfb77ffb96ed8d2e31506788a86ac4c49fc7c3f123afbda2a070ce908a3530f7f75c
ed2e6d5564bbde4934eadb4bc99744...",
      "verify_code": "67650446",
      "firmware_type": 1
    },
    "ble": false,
    "bms": false,
    "forced_status": false,
    "forced_content": "",
    "special_version_status": false
  },
  "desc": "Mission accomplished.",
  "tip": ""
}
```

E-scooter - Firmware Encryption/Decryption

```python
def encrypt(self, data):
    data = self.Binary_Utilities_OBJ.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.Binary_Utilities_OBJ
            .ecb_enc(self.Binary_Utilities_OBJ
                        .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.

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

```python
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.Binary_Utilities_OBJ
                    .xor(self.iv, self.Binary_Utilities_OBJ
                                .ecb_dec(ct, self.key))
            self.iv = ct
            self.offset += 8
            if (self.offset % 1024) == 0:
    self.update_key()
        return self.Binary_Utilities_OBJ.unpad(res)
```

E-scooter - Modify Firmware Parameters

Set **KERS** Minimum Speed

```
def kers_min_speed(self, kmh):
    val = struct.pack('<H', int(kmh * 390))
    sig = [0x25, 0x68, 0x40, 0xF6, 0x24, *[None]*2, 0x42]
    ofs = self.Firmware_Utils_OBJ.GetPattern(self.data,sig)+2
    pre, post = self.Firmware_Utils_OBJ
      .Patch(self.data, ofs, 4, val, MOVW_T3_IMM)
    return [(ofs, pre, post)]
```

Starting From **Standstill**

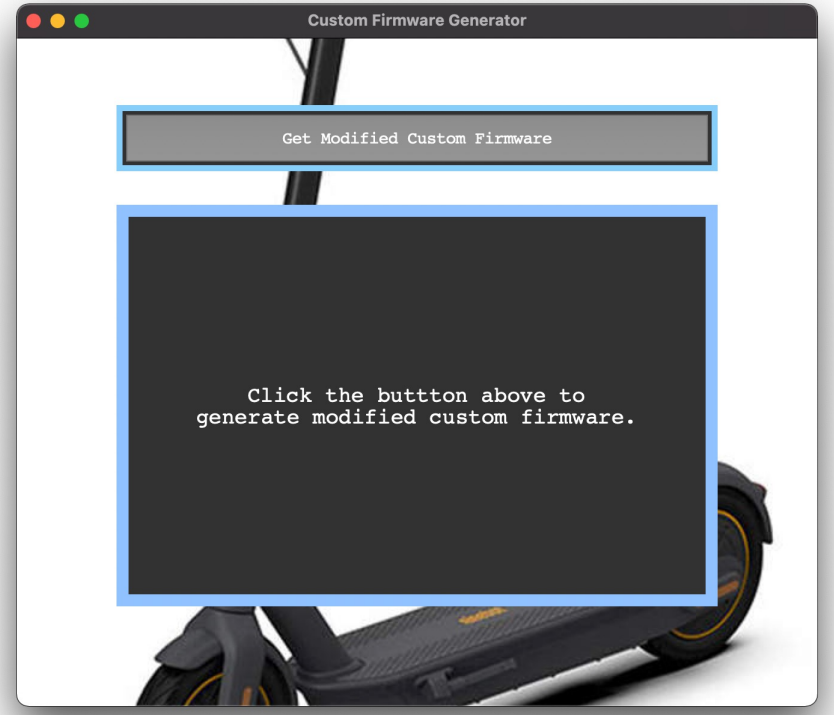
```
def motor_start_speed(self, kmh):
    ret = []
    val = int(kmh * 390)
    val = val - (val % 16)
    assert val.bit_length() <= 12, 'bit length overflow'
    sig = [0x4B, 0x01, None, None, 0x48, 0x00, None,
           None, None, 0xB6, 0xF5, 0xC5, 0x6F, 0x0D, 0xDB]
    ofs = self.Firmware_Utils_OBJ.GetPattern(self.data,sig)+9
    pre = self.data[ofs:ofs + 4]
    post=bytes(self.ks.asm('CMP.W R6, #{:n}'.format(val))[0])
    self.data[ofs:ofs + 4] = post
    ret.append((ofs, pre, post))
    ofs += 4

    pre,post = self.data[ofs:ofs + 1],bytearray((0x0D, 0xDB))
    self.data[ofs:ofs + 2] = post
    ret.append((ofs, pre, post))
    return ret
```

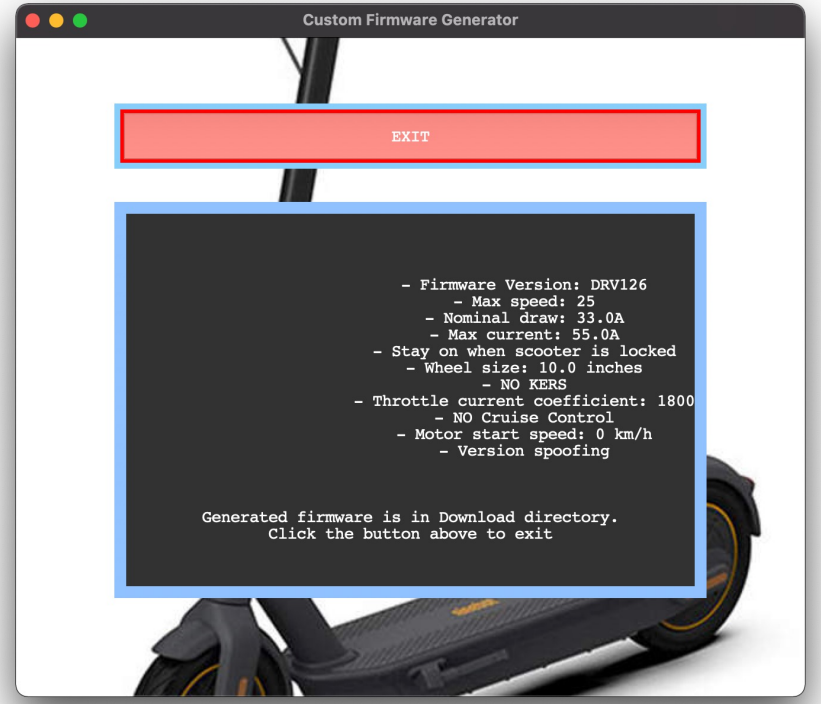
Edit **Accelerator** Curve

```
def throttle_alg(self):
    sig = [0xF0, 0xB5, 0x25, 0x4A, 0x00, 0x24, 0xA2,
           0xF8, 0xEC, 0x40, 0x24, 0x49]
    ofs = self.Firmware_Utils_OBJ.GetPattern(self.data,sig)+4
    pre, post = self.data[ofs:ofs +1],bytearray((0x01, 0x24))
    self.data[ofs:ofs + 2] = post
    return [(ofs, pre, post)]
```

E-scooter - GUI for automatic firmware generation

Automated custom **firmware** generator





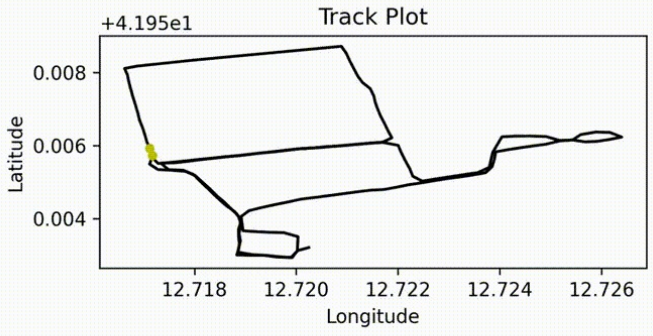
E-scooter + Wheelchair – Experimental Results

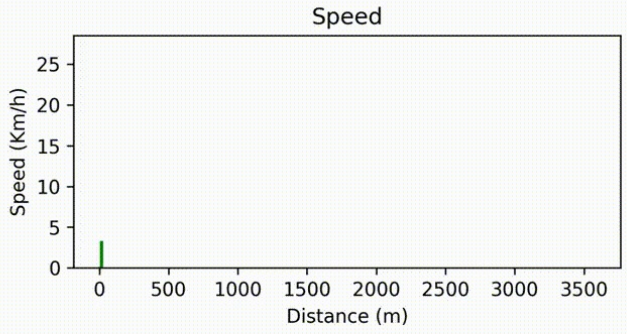
PERFORMANCE EVALUATION

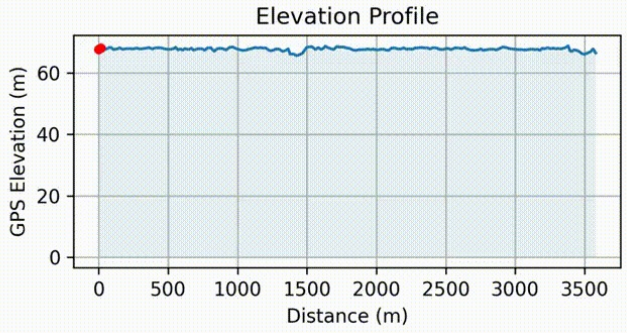
Experimental Results

Distance Traveled during **Road Test**

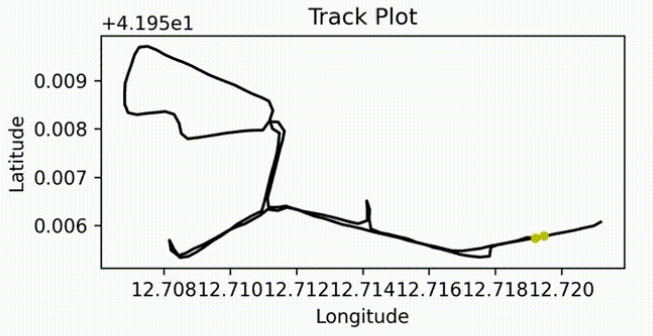
Electric Scooter + Wheelchair

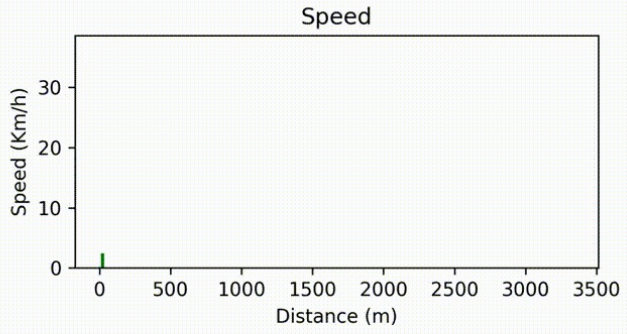


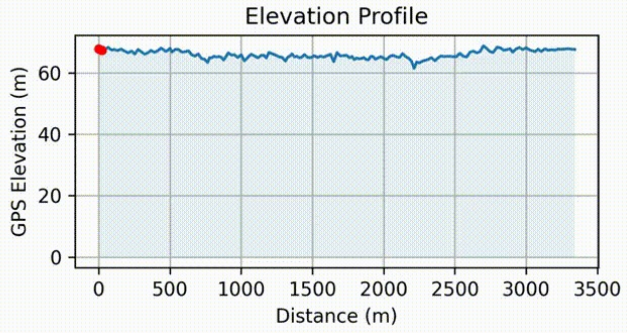




Electric Scooter Stock Firmware

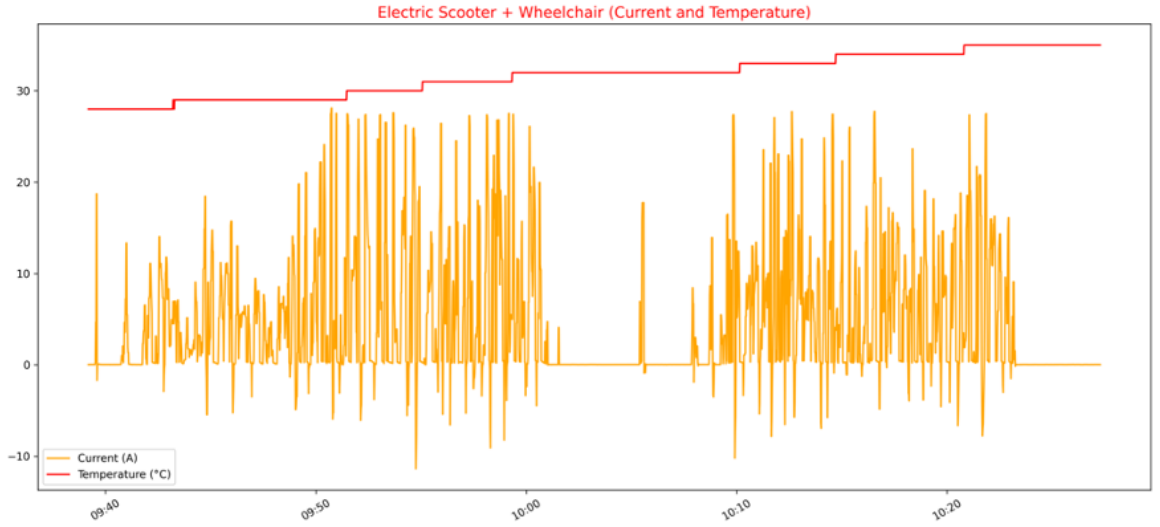


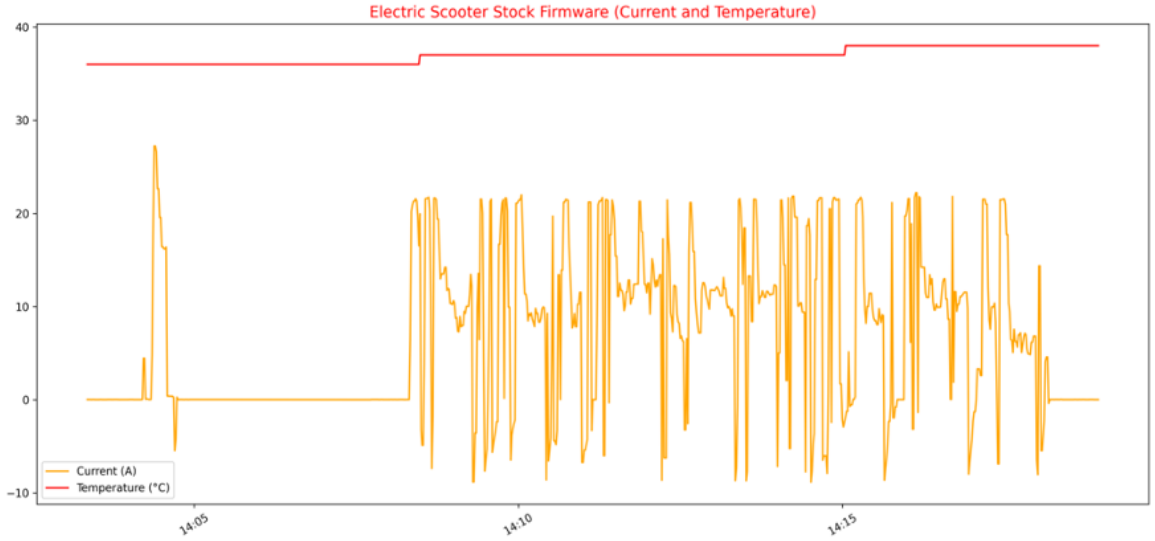




Experimental Results

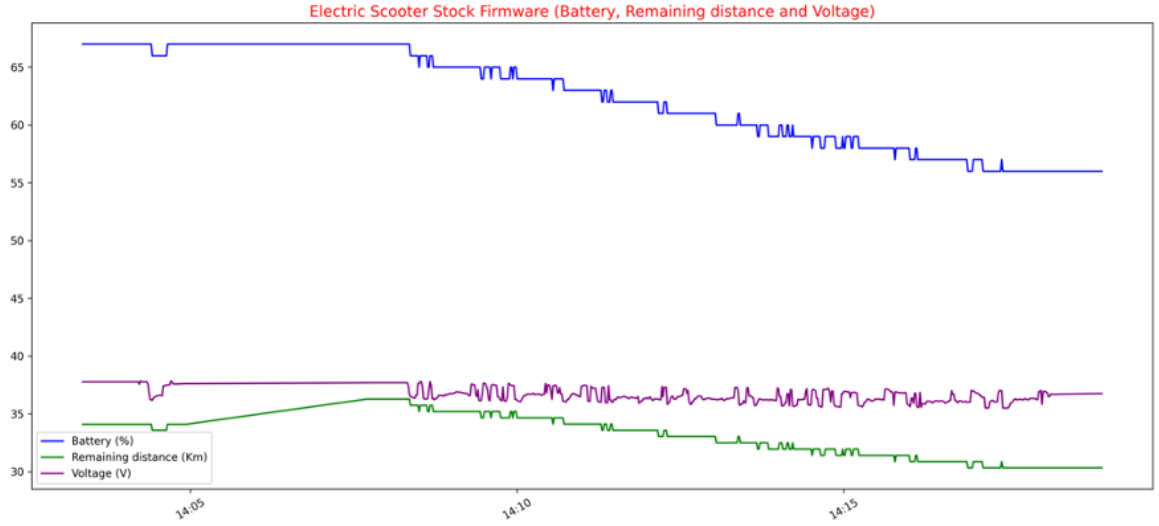
Current and Temperature Over Time





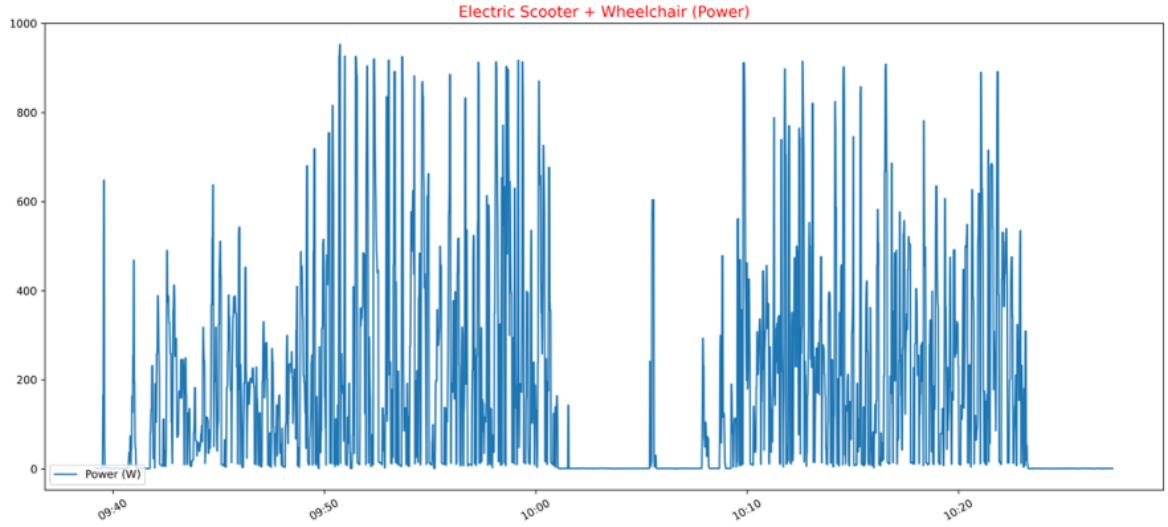
Experimental Results

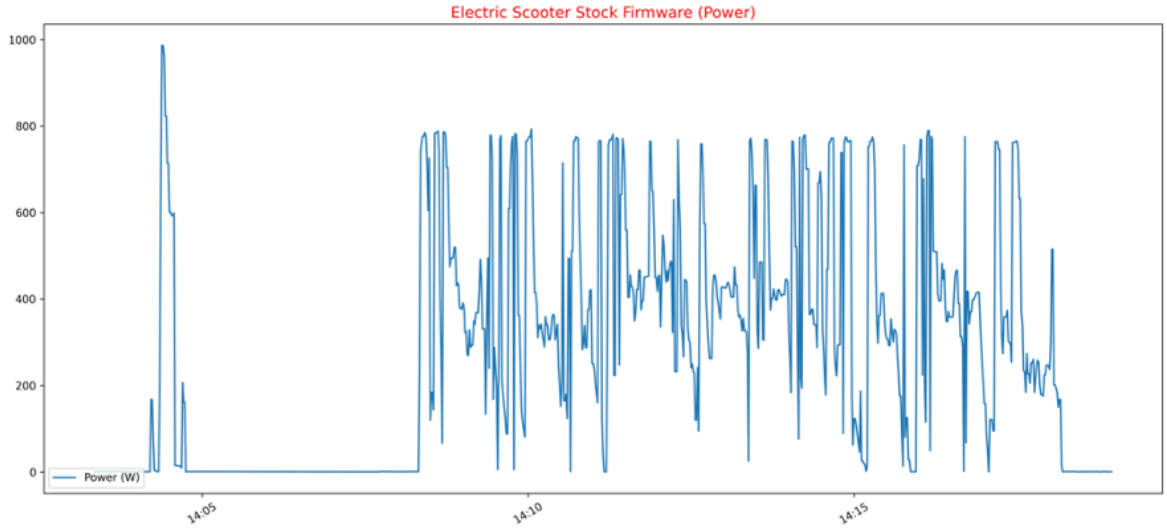
Battery, Remaining Distance and Voltage Over Time



Experimental Results

Power Over Time





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