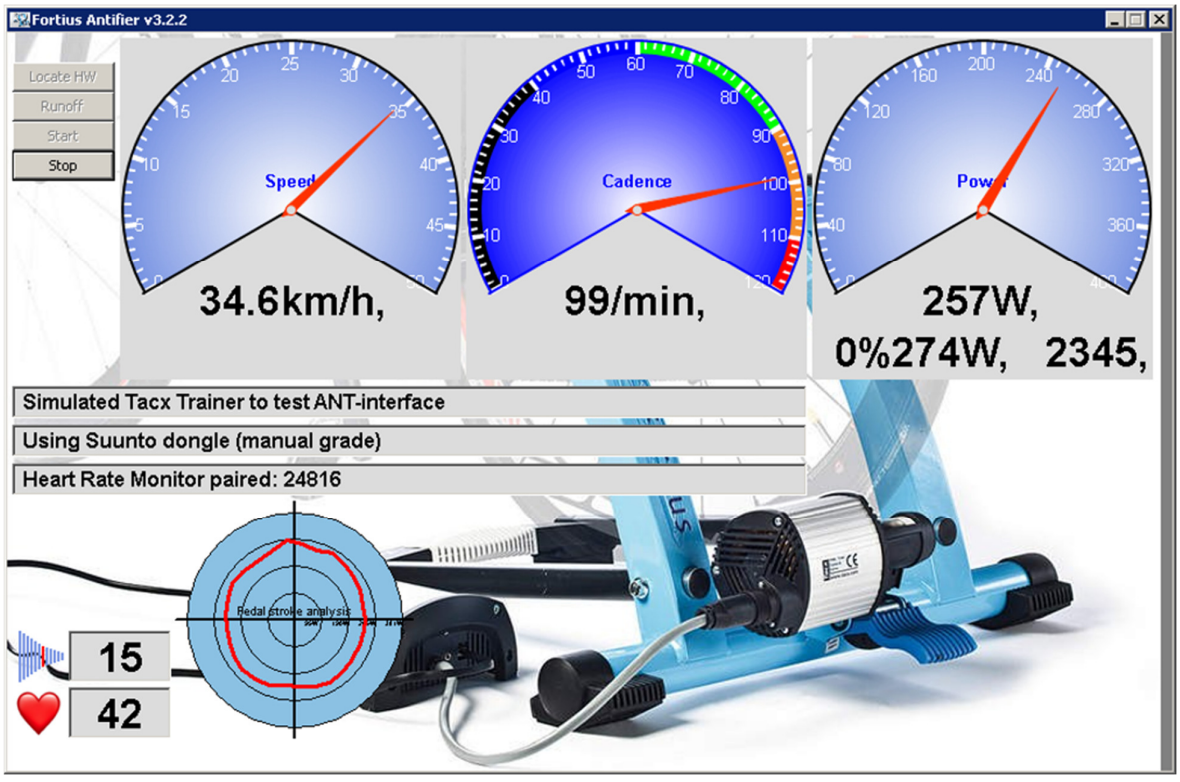
Fortius ANT

User Manual

Author and copyright: Wouter Dubbeldam



Contents

1. Introduction

1.1 Summary

Tacx created trainers and provided software (Tacx Training Software, TTS) to enable users to do structured training or ride in virtual world. Trainer and TTS were sold in a package and the interface was not available for other manufacturers.

Based upon these initial products, open standards were defined. ANT+ defines the way how CTP's (Cycling Training Programs) and FE-C's (Controllable Fitness Equipment) communicate with each other. This open standard enables that software (CTP) and hardware (FE-C) can be created by different manufacturers.

Known CTP's are Zwift, Trainer Road and Rouvy. Programs have their own specialty domain: Zwift provides the possibility to train together in a virtual world, Trainer Road has structured training programs and Rouvy allows to ride in augmented reality – and of course each product also provides functionality in the competitive area.

Trainers are provided by hardware manufacturers, like Tacx, Wahoo, Elite and others.

The open ANT+ standard allows the CTP's to communicate with FE-C's.

BUT: old Tacx trainers are left behind – the proprietary Tacx-interface is not supported by the modern CTP's.

This is where FortiusAnt has its place: **FortiusAnt enables users of old Tacx Trainers (like Fortius, Magic, Flow, Vortex and others) to use modern Cycling Training Programs, [CTP], (like Zwift, Trainer Road, Rouvy and others).**

Important: only one computer (laptop or desktop) is required to run FortiusAnt and a [CTP].

1.2 Thanks

FortiusAnt is based upon the methods and architecture from **[Antifier]** and uses the interface description from **[TotalReverse]** and without their work, FortiusAnt would not have existed. Same is valid for Golden Cheetah for i-Magic's resistance formulas.

Thanks to **@darkpotpot** and **@iepuzaur** for testing the i-Vortex, **@yegorvin** for testing the iMagic power curve and **@ElDonad** for cracking the CYCPLUS dongle issue and **@mattipee** for enthusiasm on programming, correctness and testing as well as valuable github- and vsc-lessons. Of course, thanks to all who have reacted and added to FortiusAnt improvements.

If you use FortiusAnt, let me know and add yourself to the map!

FortiusAnt has matured in 2020's corona era, locked-down@home - sporting@home - programming@home. It shows where github code-sharing and joint effort can bring us; a new and inspiring experience for myself.

If you use FortiusANT, let me know and you add stars on the map!

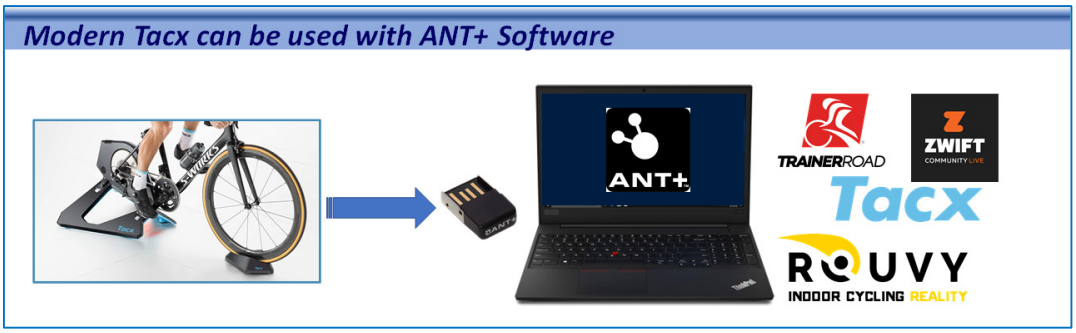
1.3 The structure of this manual

After the introduction you are now reading; the manual has the following chapters

- Background information; explaining concepts
- Installation instructions; how to install FortiusAnt
- Operating instructions; how to run FortiusAnt
- Questions and special situations; to describe anything else

I hope the manual helps in operating FortiusAnt and I'm always happy to hear from you @github!

1.4 The story in a picture:



1.5 References, abbreviations, and terminology

Term	Explanation	See also
[Antifier]	The predecessor of FortiusAnt, created by “John”.	https://github.com/john-38787364/antifier
[ANT+]	ANT+ is a wireless technology that allows devices to talk to each other. The following documents are most interesting to study when digging into the python code: • D00000652_ANT_Message_Protocol_and_Usage_Rev_5.1.pdf • D00001198_-_ANT+_Common_Data_Pages_Rev_3.1.pdf • D000001231_-_ANT+Device_Profile-Fitness_Equipment-Rev_5.0(6).pdf • D00000693_-_ANT+Device_Profile-_Heart_Rate_Rev_2.1.pdf	www.thisisant.com
[CTP]	Cycling Training Program, such as Zwift, Trainer Road or Rouvy	
[FE]	Fitness Equipment, like legacy Tacx trainers	
[FE-C]	Controllable Fitness Equipment, the ANT+ name for an indoor trainer. A **[FE]** + **[FortiusAnt]** becomes an FE-C.	
[FortiusAnt]	FortiusAnt enables a usb-connected Tacx trainer to communicate with TrainerRoad, Rouvy or Zwift through ANT.	https://github.com/WouterJD/FortiusAnt
[Python]	Python is a high-level programming language	www.python.com
[TotalReverse]	Invaluable source of information regarding Tacx USB interfaces.	https://github.com/totalreverse/ttyT1941
[TTS]	Tacx Training Software; proprietary **[CTP]** connecting to Tacx trainers only.	

1.6 Revision History

Date of this revision: November 6th, 2020 Version: v3.g Published

Version	Revision Date	Summary of Changes
3.i	November 26th, 2020	Supported tacx trainertypes described in more detail (see 2.3) - Command line parameter -r (resistance) introduced; to send TargetPower directly to brake for test-purpose - Command line parameter -G offset/factor (ModifyGrade) introduced; default is 0/1, 4.3/2 for Magnetic Brake - Power calculations implemented for magnetic brakes
3.h	November 15th, 2020	Debugging section added
3.g	November 5th, 2020	In manual power/grade mode, a tcx file is created. Sections added: - 2.6 Power curve validation - 5.6 Fortius without cadence sensor - 5.7 Two ANTdongles – disturbed communication
3.f	October 21st, 2020	i-Vortex and -P option added
3.e	October 16th, 2020	Command line parameter -u uphill added Section added: 5.5 Tacx head unit with firmware to be loaded. Minor textual modifications
3.d	October 5th, 2020	Section Requirements added
3.c	October 2nd, 2020	Only ONE computer is required to run FortiusANT and a **[CTP]**, two computers are drawn in the pictures to explain the concept.
3.b	October 1st, 2020	First version published
3.a	June, 17th 2020	First version

2. Background information

2.1 FortiusAnt Sensor- and Monitor devices

ANT+ terminology is Master and Slave which may be replaced with other names in near future.

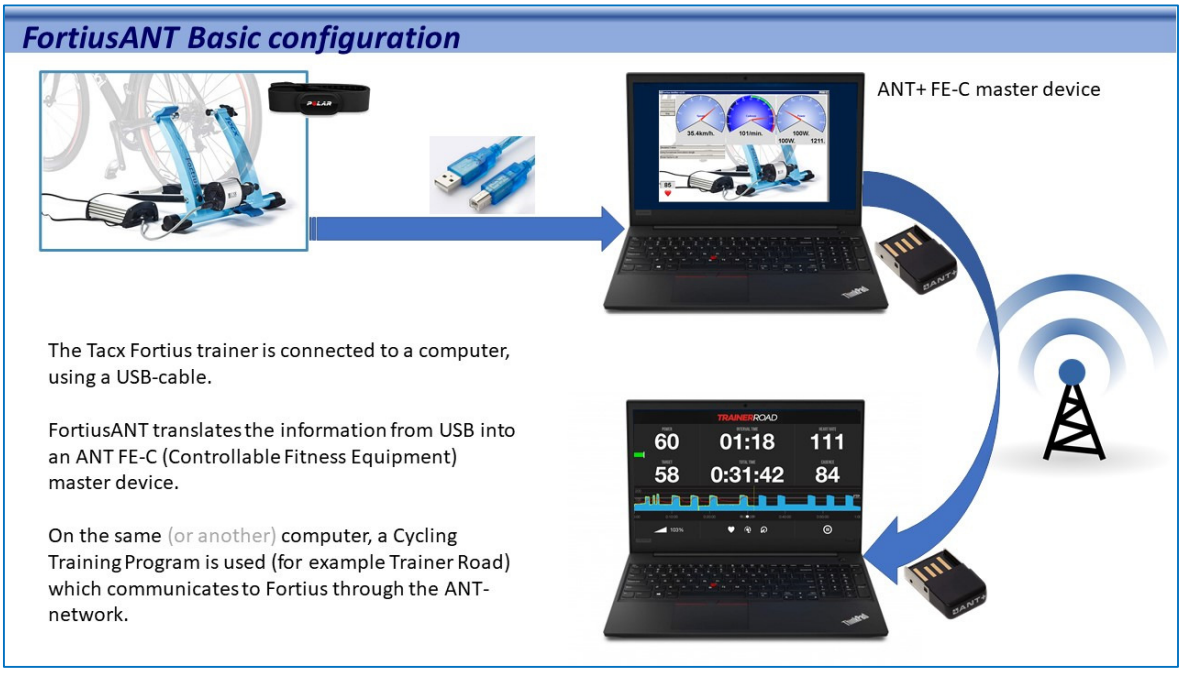
ANT+ Master devices are Heartrate monitor, Powermeter, Speed- and Cadense sensor and **[FE-C]**'s. "Master" could be replaced by "Sensor": they measure and transmit the results through ANT+.

ANT+ Slave devices receive the transmitted signals and display the results, like a speed/cadence/power display on a bicycle; they may also send commands to the sensor – for example a **[CTP]** sends commands to a **[FE-C]**. "Slave" could be replaced by ""Monitor": the display/monitor/control what the sensor does.

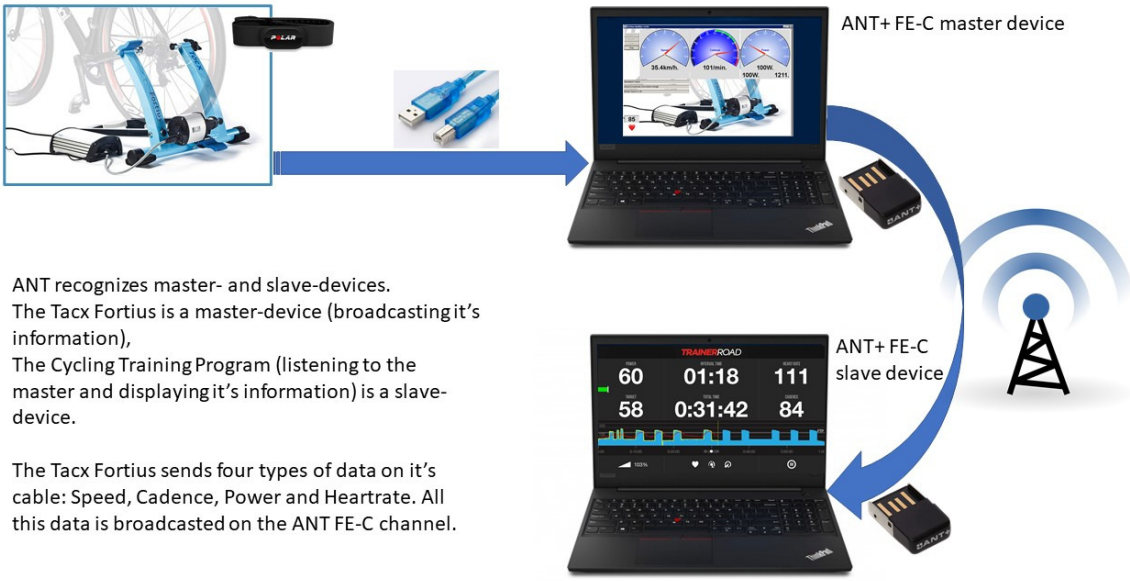
FortiusANT follows ANT+ terminology also when the terms Master and Slave are replaced in future.

To understand "what we are doing"; the following pictures explain at a high level what happens "under the hood".

Important: only one computer (laptop or desktop) is required to run FortiusAnt and a **[CTP]**. Two computers are shown to explain the concept. One computer with three USB-connections will do the job: (1) the USB-cable to the Tacx Trainer, (2) the ANT+ dongle for FortiusANT and (3) the ANT+ dongle for the **[CTP]**. And indeed: the two programs on one computer (FortiusANT and [CTP]) communicate with each other externally and wirelessly, using two ANT+ dongles.
See also section 3.2 "Requirements".



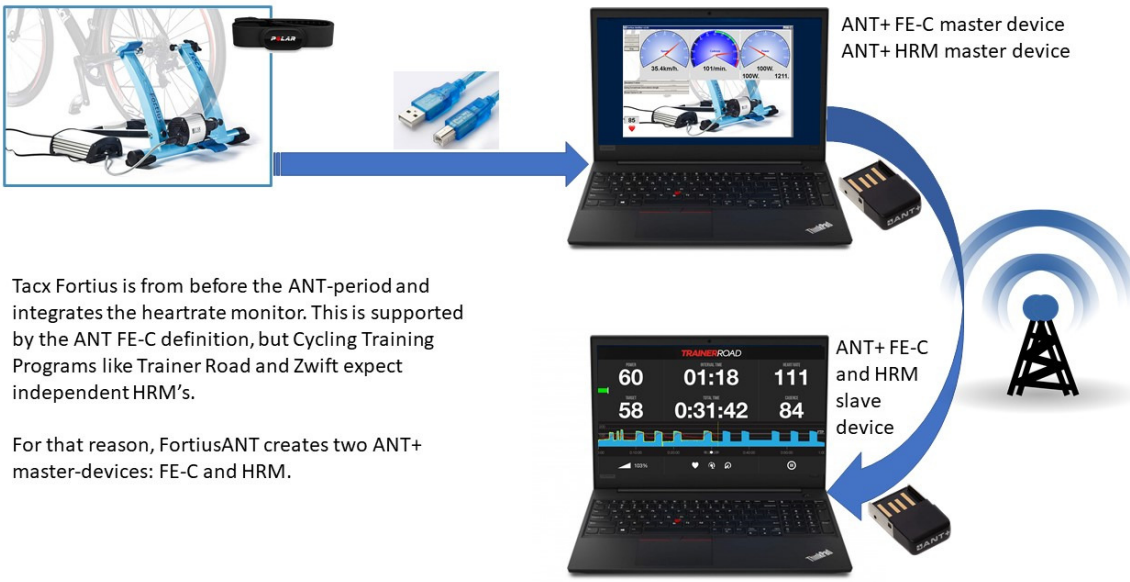
ANT master and slave devices



ANT recognizes master- and slave-devices.
The Tacx Fortius is a master-device (broadcasting it's information),
The Cycling Training Program (listening to the master and displaying it's information) is a slave-device.

The Tacx Fortius sends four types of data on it's cable: Speed, Cadence, Power and Heartrate. All this data is broadcasted on the ANT FE-C channel.

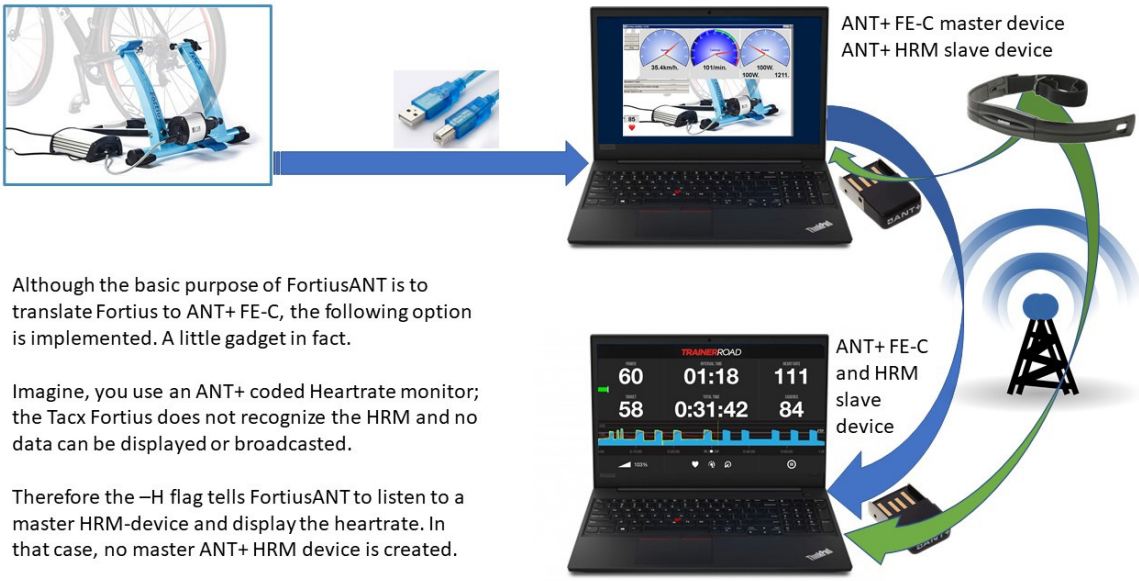
FortiusANT creates two ANT+ master devices (FE-C and HRM)



Tacx Fortius is from before the ANT-period and integrates the heartrate monitor. This is supported by the ANT FE-C definition, but Cycling Training Programs like Trainer Road and Zwift expect independent HRM's.

For that reason, FortiusANT creates two ANT+ master-devices: FE-C and HRM.

FortiusANT can also listen to an independent ANT+ HRM strap



Although the basic purpose of FortiusANT is to translate Fortius to ANT+ FE-C, the following option is implemented. A little gadget in fact.

Imagine, you use an ANT+ coded Heartrate monitor; the Tacx Fortius does not recognize the HRM and no data can be displayed or broadcasted.

Therefore the –H flag tells FortiusANT to listen to a master HRM-device and display the heartrate. In that case, no master ANT+ HRM device is created.

Zwift works slightly different

As explained, FortiusANT creates two master devices

ZWIFT shows the following devices that can be paired individually:

- ANT+ Heartrate monitor
- ANT+ Controller
- ANT+ Cadence sensor
- ANT+ Power meter

The latter three can all be paired to the FortiusANT FE-C device, but if your bicycle has an own power-meter, you could choose to connect to that device instead.

2.2 Description

A **[CTP]** (Cycling Training Program) send commands to the **[FE-C]** (Controllable Fitness Equipment) through ANT+. There are two modes: Power- or Ergo-mode and Resistance- or Slope-mode.

2.2.1 Resistance

It is important to understand that an **[FE-C]** only knows 'resistance' which is the torque to turn the braking axle. From physics we know that power (Watt) = torque (Nm) * speed (km/hr). For a given resistance, the power required is linear with the speed (=cadence), provided you do not change gears. Also, for a given resistance and cadence, the power required is linear with the gear-ratio.

PS. This is especially valid for the older trainers; Tacx i-Vortex can be set natively to a defined Power.

2.2.2 Power- or Ergo-mode

In Powermode the **[CTP]** sends the required power to the **[FE-C]** and regardless gear or cadence, the requested power is constant. FortiusAnt calculates the resistance = power/speed (with some constants applied). Note that, if you change gears and/or cadence, the required power will remain equal because the resistance is adjusted.

2.2.3 Resistance mode

In resistance mode, FortiusAnt receives the required grade from **[CTP]**. Grade may vary from -20% (downhill) to +20% (uphill). FortiusAnt calculates the resistance to be sent to the **[FE]**.

The power required to ride up a hill is based upon the given grade with a weight of 90 kg (rider + bike) at a given speed'. Input parameters are grade and weight (from **[CTP]**) and speed (as measured by **[FE]**). Result is Power and conversion to resistance is described above.

2.3 Tacx trainers

Tacx has created a large variety of trainers with commercial names like Magic, Flow, Fortius, Vortex, Bushido, Genius, with or without i-. Technically, you will find three numbers: one for the configuration, one for the head unit, one for the brake and then for other optional equipment parts. For more information see **[TotalReverse]**, “Tacx product number overview” where the T-numbers describe the part they represent.

FortiusANT is developed for the configuration I own: a **T1930** Tacx Fortius Multiplayer bundle with **T1941** brake, **T1932** PC-head unit and **T1905** steering unit.

FortiusANT converts the USB-interface to an ANT+ interface. In this USB-mode, FortiusAnt only “knows” the head unit, not what brake is behind it or what “commercial name” is used, that is completely transparent. For the user it should be transparent as well, but note that, where we talk about a Tacx Flow, Magic or Fortius, FortiusANT only knows the head unit T1902, T1932, T1942 (etc). The head unit comes in two flavors: T1902 head unit (the so-called USB legacy interface) and all other USB head units.

Behind the head unit there are two types of brakes: magnetic and motor. We assume that the characteristics are hidden by the head unit. Note however that today’s (Nov 2020) experience suggests a correlation between the T1932/T1904 head-units and issues with the magnetic brakes.

Bundle	Brake	Head unit	Remarks
T1930 Tacx Fortius	T1941 Motor	T1932	USB interface tested, see 2.6.2
T1900 Tacx i-Magic	T1901 Magnetic	T1902	Legacy USB interface tested, see 2.6.3
T???? Tacx Flow	T1901 Magnetic	T1932	Issues: #102 #143
T???? Tacx i-Magic	?	T1904	Issues: #128

When directed to do so (using the -t flag), FortiusANT does not look for a USB-trainer, but tries to pair with a proprietary ANT+ Tacx trainer. FortiusANT then communicates with the brake and optionally with the head unit.

FortiusANT supports the following trainers, connecting to the **USB head unit**:

- Head unit T1902: Old "solid green" iMagic head unit (with or without firmware)
  This head unit uses a so-called legacy-USB protocol, the others the New-USB-protocol.
- Head unit T1904: New "white, green" iMagic head unit (firmware inside)
- Head unit T1932: New "white, blue" Fortius head unit (firmware inside)
- Head unit T1942: Old "solid blue" Fortius (firmware inside)
- Head unit 0xe6be: Old "solid blue" Fortius (without firmware)
  This head unit requires software to be loaded when FortiusANT is started.
  See also section 5.5 “Tacx head unit with firmware to be loaded”.

FortiusANT supports the following trainers (-t flag), connecting to an **ANT+ brake**:

- -t i-Vortex: T1961 brake and T2172 head unit are supported (see issue #46)
- -t Bushido: Under development, (see issue 117)
  Head unit T1982
- -t Genius: Under development, (see issue 101)
  Head unit T2020

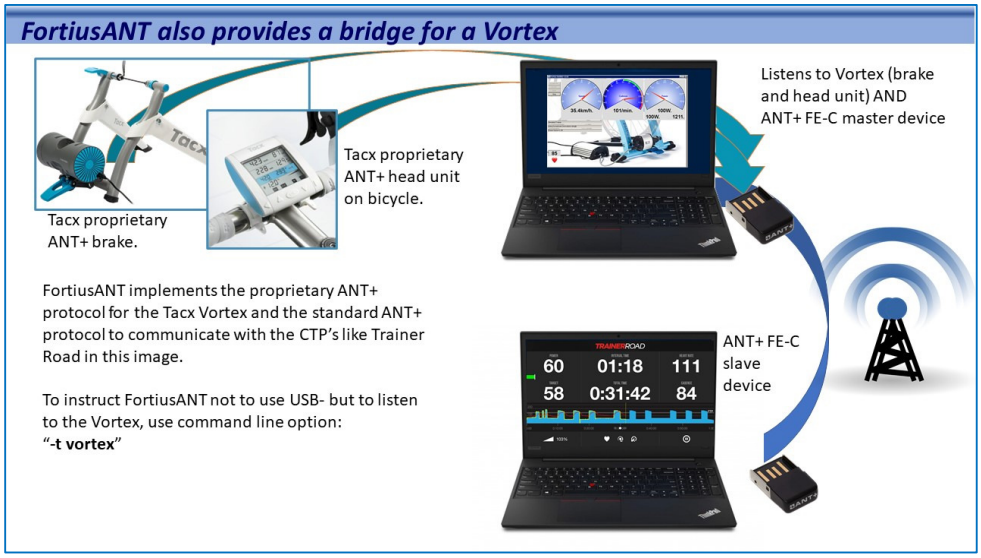
2.4 Tacx proprietary ANT trainers

The ANT-interface was initially used to connect a heartrate monitor to a watch and from there developed into a low-energy wireless protocol in the sport-environment. For more info, refer to www.thisisant.com.

When the whole world went wireless, the USB-connection became obsolete. ANT+ enabled to connect trainers with a variety of computers: windows, tablet, smartphone, etc. Tacx followed and built trainers with an ANT-connection but decided to keep the interface between trainer and software proprietary.

And hence Bushido, Genius, Flow and Vortex were created, communicating with an ANT+ protocol, but not following the standard regarding transmission details and data.

FortiusANT can communicate with those trainers. Functionally you may see it that instead of a USB-cable, the ANT-dongle is used. Further there are no functional differences.



2.5 Calculations

2.5.1 Power mode

When the **[CTP]** is in power-mode, a required number of Watts is sent to **[FE-C]**, in our case FortiusAnt. “The head-unit of the Tacx-trainer” in short: “The **[FE]** requires a resistance to be set and hence a function TargetPower2Resistance(Power, Speed) is used to convert. The function is different for legacy- and new-USB trainers.

So, if you want to ride with a power of 100Watt and the bicycle wheel runs at 10kmh, the **[FE]** needs to receive another required resistance than when the wheel is rotation at 40kmh.

Similarly, the **[FE]** returns the currently realized resistance and a function is used to calculate the realized Actual Power.

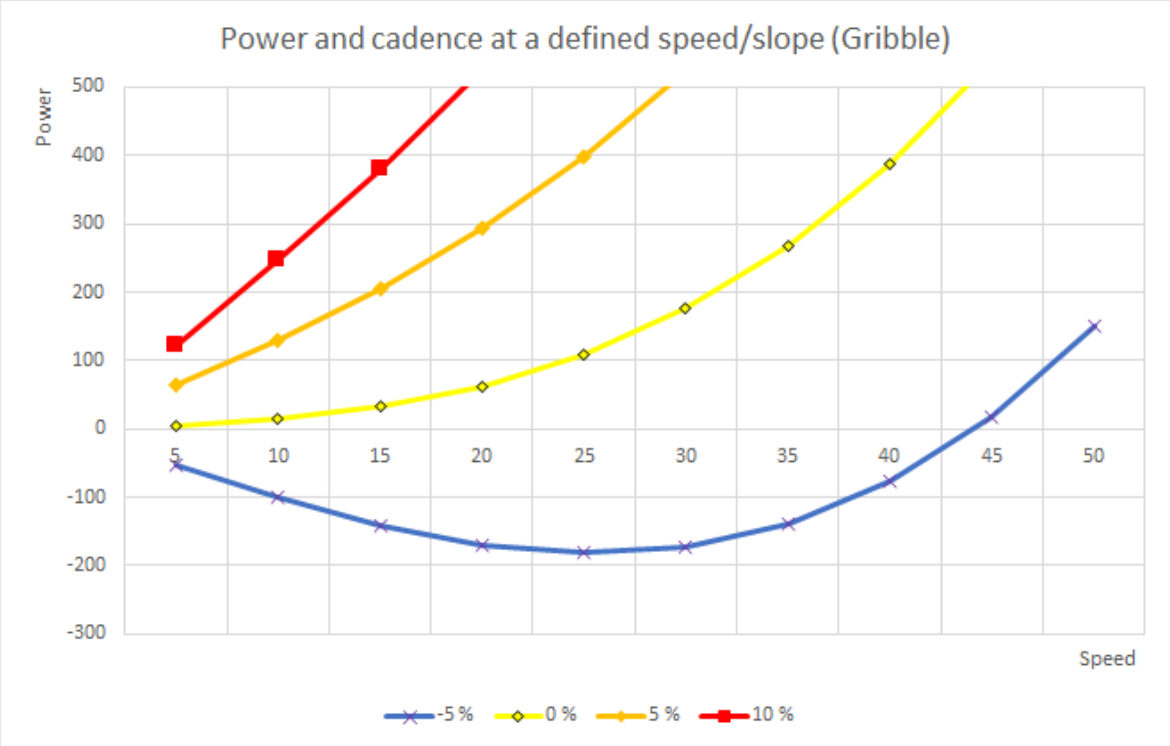
2.5.2 Grade mode

In Grade mode, the **[CTP]** communicates the slope where you are riding: flat = 0%, up hill (e.g. 10%) or downhill (e.g. -10%).

An additional step is required, using function Grade2Power(Grade, Speed, UserAndBikeWeight). First the Grade is converted to power and then the same applies as described in the previous paragraph.

The function also depends on RollingResistance, WindResistance, WindSpeed and DraftingFactor. Constants are used if **[CTP]** does not explicitly specify these parameters. The default value for UserAndBikeWeight is 85kg.

The formula results in the power curve as shown below. Note that, on a flat surface (yellow curve) you need 100Watt to ride 25km/hr. On a hill of 5% (orange curve) you need 300Watts at 20km/hr. And if you ride downhill -5% (blue curve) you would ride around 44 km/hr without adding additional power.



2.6 Power curve validation

2.6.1 General

As explained **[CTP]**'s require a Power or Grade and either requirement is translated into a required resistance as explained in the previous section. While developing the calculations were validated and it may be useful to check for your own trainer whether the calculations are correct.

"The Power Curve" is the relation between requested Power and the resistance set on the trainer.

The formula basically is: Resistance = c * Power / Speed; the resistance of the brake goes up with increasing power and reduces with increasing speed (of the bicycle wheel).

Overall conclusion from the tests is that the algorithm does what it should do, which is confirmed by users who are happy how their trainer works. Individual situations are reported however (e.g. issue 102) that the power curve is not satisfactory. This section can be used to validate your own trainer.

2.6.2 Test for Tacx Fortius (T1932)

The algorithm is tested on a Tacx Fortius with a T1932 head unit, using a bicycle with power-meter. FortiusAnt is started in manual mode (-m flag) so that power can be selected.

Test protocol.

Select gears for 10, 20, 30, 40, 50 km/hr at reasonable cadence (e.g. 90 rpm).

For each speed perform the following test:

Manually select power 50Watt (up/down button on head unit).

Ride until reading from FortiusAnt and Power Meter is stable.

Write down power from Power Meter (46 in table).

Increase power on head unit 100, 150, 200, 250, 300 Watt and repeat test.

Test Results (Target Power in column header, result power in the table)

	50W	100W	150W	200W	250W	300W
10 km/hr	46	97	145	194	245	285
20 km/hr	50	100	145	197	245	290
30 km/hr	63	102	154	196	245	295
40 km/hr	105	120	160	210	260	305
50 km/hr	123	130	165	210	250	310

With the assumptions:

- Attempts to improve the algorithm may be useless, since it should not be more exact than the Tacx Fortius (was designed for).
- Changing the algorithm remains giving an empirical result since the technical specs from Tacx are not available to validate

The conclusions are:

- 50Watt at 50km/hr gives odd readings but that is not too strange.
- Overall measured power corresponds with Target Power.
- Multiple measurements give different results within 5%

Tests are done intermittently and hence have different brake and tire temperatures.

2.6.3 Test for i-Magic (T1901-T1902)

@yegorvin has performed a performance calibration test using his iMagic with T1901 magnetic brake, T1902 head unit and a power meter. Note that the T1902 head unit is different from (all) other head units **[TotalReverse]**.

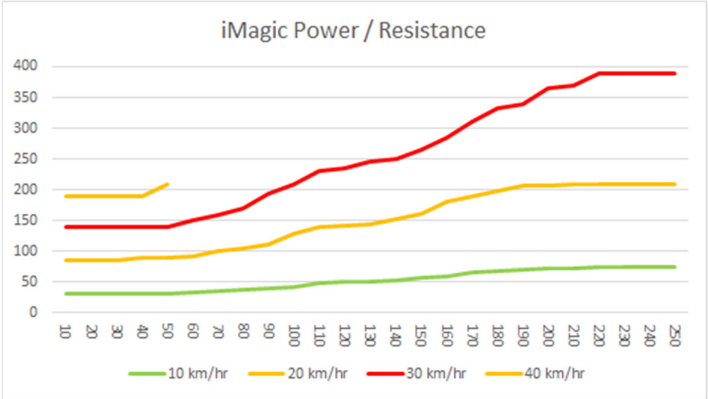
The test is performed as follows:

- FortiusANT is operated in manual mode (-m flag), modifying the power from 10...250 in 10 Watt steps.
- The power as selected is transferred directly to the iMagic (Resistance = TargetPower) without intermediate formula. (The python code is changed for this)

Results of the test are:

Res	10km/hr	20km/hr	30km/hr	40km/hr
10	32	85	140	190
20	32	85	140	190
30	32	85	140	190
40	32	89	140	190
50	32	89	140	210
60	34	93	150	
70	35	100	160	
80	38	105	170	
90	40	112	193	
100	42	128	210	310
110	48	139	230	
120	50	142	235	
130	50	145	245	
140	52	152	250	
150	58	162	265	~410
160	60	180	285	
170	65	190	310	
180	68	198	332	
190	70	207	340	
200	72	207	365	~550
210	72	210	370	
220	75	210	390	
230	75	210	390	
240	75	210	390	
250	75	210	390	

The power curve of the iMagic is as shown below, where the range R=50...200 seems quite linear.



Convert Resistance to Power formula's

The assumption is that Power = a * Resistance + b, when using the target resistance of 50W we have three formula's, one for each speed:

```
Speed = 10:  32 = 50 a + b
                  72 = 200 a + b ==> P10 = R * 0.27 + 19
Speed = 20:  89 = 50 a + b
                  207 = 200 a + b ==> P10 = R * 0.78 + 50
Speed = 30: 140 = 50 a + b
                  365 = 200 a + b ==> P10 = R * 1.50 + 65
```

And with this, we could create the function as follows:

```
Function Resistance2PowerLegacy(Resistance, Speed) As Double
  If Speed = 30 Then
        Resistance2PowerLegacy = Resistance * 1.50 + 65
  ElseIf Speed = 20 Then
        Resistance2PowerLegacy = Resistance * 0.78 + 50
  ElseIf Speed = 10 Then
        Resistance2PowerLegacy = Resistance * 0.27 + 19
  End If
End Function
```

But of course, this would work for three distinct speeds only.

A good formula for the constants 65, 50 and 19 is 2.2 * Speed and validation shows that it's good enough.

But there is no linear solution for 1.50, 0.78 and .27, so let's go for F = ax2 + bx + c (where x=Speed)

```
0.27 = 100 a + 10 b + c
0.78 = 400 a + 20 b + c
1.50 = 900 a + 30 b + c    ==> F = 0.001543 Speed2 + 0.0001848 * Speed + 0.1058
                           <==> Speed2 / 648 + Speed / 5411 + 0.1058
```

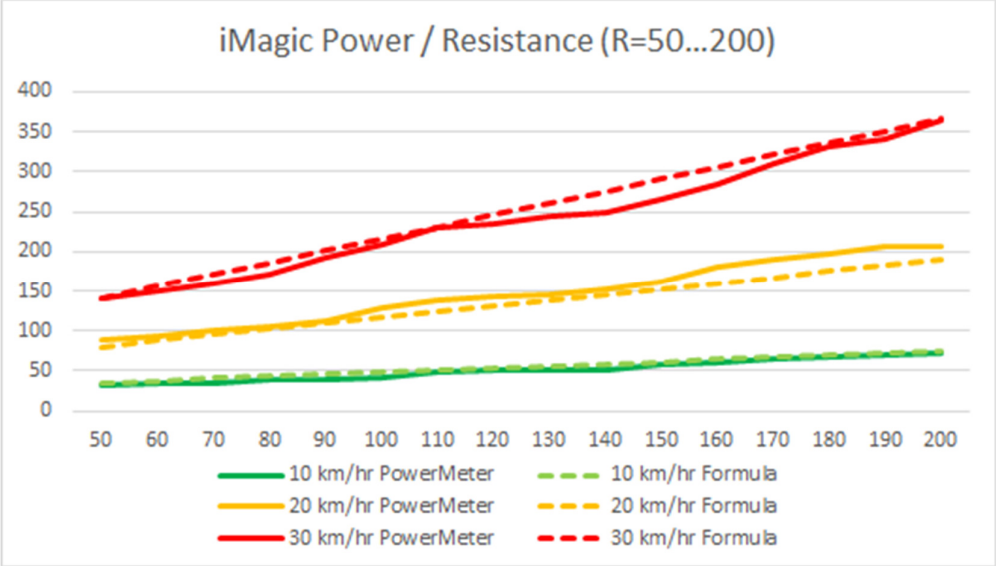
and the final formula then becomes:

```
Function Resistance2PowerLegacy(Resistance, Speed) As Double
  Resistance2PowerLegacy =
        Resistance * (Speed * Speed / 648 + Speed / 5411 + 0.1058) + 2.2 * Speed
End Function
```

To check:

```
S = 10,   R =  50; P =  50 * (10 * 10 / 648 + 10 / 5411 + 0.1058) + 2.2 * 10 =  35
          R = 200; P = 200 * (10 * 10 / 648 + 10 / 5411 + 0.1058) + 2.2 * 10 =  74
S = 20,   R =  50; P =  50 * (20 * 20 / 648 + 20 / 5411 + 0.1058) + 2.2 * 20 =  80
          R = 200; P = 200 * (20 * 20 / 648 + 20 / 5411 + 0.1058) + 2.2 * 20 = 190
S = 30,   R =  50; P =  50 * (30 * 30 / 648 + 30 / 5411 + 0.1058) + 2.2 * 30 = 141
          R = 200; P = 200 * (30 * 30 / 648 + 30 / 5411 + 0.1058) + 2.2 * 30 = 366
```

With the Powermeter-measured and Formula-calculated values into one graph, the result is as shown and close enough to be used (or at least tested):



Conclusion:

The formula based upon @yegorvin's measurements can be used by the 6 known Tacx i-Magic T1902 users at this moment.

2.6.4 PowerCurve for i-Flow (T1901-T1932)

Antifier defines 14 distinct resistance values for Magnetic Brakes, connected to the T1932 head unit. For each resistance, a factor and an offset is defined to calculate the resulting power in that range:

		0	1	2	3	4	5	6	7	8	9	10	11	12	13
level4res	ResistanceLevel	1039	1299	1559	1819	2078	2338	2598	2858	3118	3378	3767	4027	4287	4677
sendResList	ResistanceSend	1900	2030	2150	2300	2400	2550	2700	2900	3070	3200	3350	3460	3600	3750
PowerFactors4Level1	Factor	5,3065	5,6008	5,9874	6,6848	7,5043	8,45	9,3906	10,46	11,751	13,015	14,2	15,3	16,55	17,763
PowerFactors4Level2	Offset	-29,348	-33,1	-37	-43,813	-51,6	-59,1	-65,5	-73,1	-80,1	-86,508	-94,3	-102,49	-112	-124,1
Grade4Level		-4,3	-3,4	-2,5	-1,6	-0,7	0,5	1,4	2,9	4	5,6	7,4	8,6	9,8	11
Resistance -> Power (antifier table)	10 km/hr	24 W	23 W	23 W	23 W	23 W	25 W	28 W	32 W	37 W	44 W	48 W	51 W	54 W	54 W
	20 km/hr	77 W	79 W	83 W	90 W	98 W	110 W	122 W	136 W	155 W	174 W	190 W	204 W	219 W	231 W
	30 km/hr	130 W	135 W	143 W	157 W	174 W	194 W	216 W	241 W	272 W	304 W	332 W	357 W	385 W	409 W
	40 km/hr	183 W	191 W	202 W	224 W	249 W	279 W	310 W	345 W	390 W	434 W	474 W	510 W	550 W	586 W
	50 km/hr	236 W	247 W	262 W	290 W	324 W	363 W	404 W	450 W	507 W	564 W	616 W	663 W	716 W	764 W

The formula is: Power = Speed * Factor + Offset

Using a table is unfortunate, because creating a neat function Resistance2Power() is not straight forward.

The ratio and the factor appear have a relation to the resistance:

Actual ratio	Resistance / factor	196	232	260	272	277	277	277	273	265	260	265	263	259	263
	Resistance / offset	-35	-39	-42	-42	-40	-40	-40	-39	-39	-39	-40	-39	-38	-38

Which can be generalized as:

Average ratio	Resistance / factor	216	233	251	268	268	268	268	268	268	268	268	268	268	268
	Resistance / offset	-40	-40	-40	-40	-40	-40	-40	-40	-40	-40	-40	-40	-40	-40

Where only the gray fields differ from the others.

Resistance → Power

The head-unit returns the CurrentResistance, which must be converted to CurrentPower (which is displayed or sent to a **[CTP]**). The constants in the formula's match the tables before as good as possible and (very likely) within the accuracy range of the magnetic brake itself.

The simple overall formula is:

```
Resistance2Power(SpeedKmh, Resistance):
    return (SpeedKmh / 268 + 1 / 40) * Resistance
```

And to adjust for the three deviating fields:

```
Resistance2Power(SpeedKmh, Resistance):
    Factor = 268
    Offset = -40
    if Resistance < 1819: Factor = Factor - (1819 - Resistance) / 15
        return (SpeedKmh / Factor + 1 / Offset) * Resistance
```

Power → Resistance

We also need the inverse formula, where the simple version is:

```
Power2Resistance(Speed, Power):
  return Power / (SpeedKmh / 268 + 1 / 40)
```

and the more sophisticated version:

```
Power2Resistance(Speed, Power):
    if SpeedKmh > 20 And Power > SpeedKmh * 6:
        rtn = Power / (SpeedKmh / 268 + 1 / 40)
    else:
        for rtn in (1039, 1299, 1559, 1819, 2078, 2338, 2598, 2858, 3118, 3378, 3767, 4027, 4287, 4677):
            if Resistance2Power (Speed, level4res(i)) >= Power: break
    return rtn
```

Grade -> (Power ->) Resistance

For the Magnetic Brake, also 14 distinct Grades are defined and to each grade a resistance is assigned.

FortiusANT works differently: a grade (while riding at a speed) is translated to power and that power is translated to a resistance.

Apparently, the choice is made to define an offset of -4.3 degrees to be the lowest grade and a maximum grade of 11 degrees. The power is distributed between them. To accommodate a similar approach, FortiusANT allows to modify the requested grade with an offset and a factor (-G flag):

```
TargetGrade = (RequestedGrade + Offset) / Factor
```

The default for Offset and Factor are 0 and 1; neutralizing this formula.

De values for a Magnetic brake are 4.3 and 2:

```
TargetGrade = (RequestedGrade + 4.3) / 2
```

Which means that -4.3 degrees results in 0 and +20 degrees results in 12 degrees.

Grade4Level		-4,3	-3,4	-2,5	-1,6	-0,7	0,5	1,4	2,9	4	5,6	7,4	8,6	9,8	11
Grade2Power	Offset 4,3			Factor 2											
	Adjusted grade	0	0,45	0,9	1,35	1,8	2,4	2,85	3,6	4,15	4,95	5,85	6,45	7,05	7,65
	10	37	47	56	66	75	88	98	114	125	143	162	175	188	201
	20	37	56	75	93	112	137	156	187	210	244	282	307	332	358
	30	37	65	93	121	149	186	214	261	295	345	401	439	476	514
	40	37	74	111	148	186	235	272	334	380	446	521	571	620	670
	50	37	83	130	176	222	284	331	408	465	547	640	702	765	827

Is if 11% is requested, the TargetGrade is 7.65% and the requested power at 10km/hr = 201Watts.

2.6.5 Test for i-Flow (T1901-T1932)

Unless what we thought initially, the T1932 head unit does not make the brake transparent and therefore the formula's as described in 2.6.2 do not work well for the T1901 magnetic brake.

@ BikeBeppe64 has performed a performance calibration test using his iMagic with T1901 magnetic brake, T1932 head unit and a power meter.

The test is performed as follows:

- FortiusANT is operated in manual mode (-m flag), modifying the target power from 10...4000 in 50 Watt steps (read next step!).
- The power as selected is transferred directly to the i-Flow (Resistance = TargetPower) without intermediate formula. (The -r flag is created for this purpose). The assumption is that the i-Flow allowed resistance ranges from 1000…4000 and that is not Watts!
- The data from FortiusANT is saved as json-file (-d127 flag, creating a full logfile).
- A bicycle is used with power meter and Garmin Edge, registering speed and power (and cadence, heartrate). This data is available as FIT file and converted to TCX.

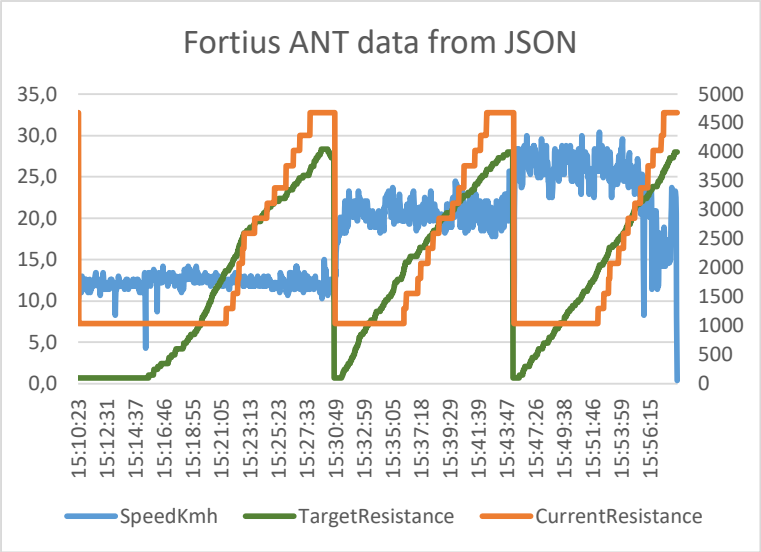
- FortiusANT and Garmin Edge are started.

- Initial TargetPower in FortiusANT is 100Watt (→ Resistance = 150)
- Start cycling at 12km/hr.
- Increase TargetPower on the headunit every 10 seconds until 4000 (or that maximum power is reached). This will take 79 steps = 13 minutes.

- Reduce TargetPower to 1000 (I think that is the minimum value, which we will see from the first test). Note that the head unit is sampled every 250ms, so you can press two times per second only! (It's not as fast as a keyboard).
- Redo the test at 24 and 36 km/hr.

- Stop FortiusANT and Garmin Edge and save .log, .json and .fit files.

The Resistance for a magnetic Brake



Performing the test, varying the TargetResistance from 0…4000 (green line) it appears that the T1932 headunit returns 14 distinct values for the CurrentResistance.

Important conclusion from this is that all resistances can be set, but they are translated to the distinct values as suitable for the magnetic brake. This is important for establishing the interface as described in the previous paragraph.

3. Installation instructions

3.1 Introduction

FortiusAnt is written in **[Python]** and can be run on any computer for which a python runtime environment is created. For Windows, FortiusAnt is also available as an executable, containing the python runtime environment. Windows users can therefore decide which version to use.

FortiusAnt communicates with the Tacx Trainer through a USB-interface, which needs some special care.

And at the other side, FortiusAnt communicates with a CTP using an ANTdongle.

3.2 Requirements

3.2.1 You, the Tacx Athlete

Getting the Tacx connected to Zwift, Trainer or Rouvy –**[CTP]** in short – needs some knowledge:

- Download documentation and software from https://github.com/WouterJD/FortiusANT/
- Be able to connect Tacx and ANT+ dongles
- Be able to install drivers on your computer
- Be able to start an executable
- Understand basic concepts, this manual only explains the FortiusANT specifics

If this is complex, ask a friend to assist!

3.2.2 Hardware

Either:

- One computer with three USB ports, running Windows, Linux or MacOS
- Two ANT+ dongles
- FortiusANT and **[CTP]** both run on this computer

Or:

- One computer with two USB ports, running Windows, Linux or MacOS to run FortiusANT
- One computer with one USB port, running Windows, Linux or MacOS to run **[CTP]**
- Two ANT+ dongles

Or:

- One computer with two USB ports, running Windows, Linux or MacOS to run FortiusANT
- One mobile device supporting ANT+ to run **[CTP]**
- One ANT+ dongle

3.3 Download FortiusAnt from github

Goto https://github.com/WouterJD/FortiusAnt/

In section <code> click on [↓Code] and download the code as a zipfile.

Create a folder on your computer called C:\Github\FortiusAnt (windows) or .../Github/FortiusAnt (unix). From now on, this folder will be referred to **<the FortiusAnt folder>**.

The downloaded file contains a folder called **FortiusAnt-master** copy the contents into **<the FortiusAnt folder>**.

3.4 Install python

Note: Python is NOT required when you use the Windows executable.

- Goto https://www.python.org/downloads/
- Follow the installation instructions for your system. It's impossible to handle all operating systems specific instructions. For Windows it's easiest NOT to install for "All users".
- Noted by @msjnaessens: "If you installed python 3.9, you cannot install numpy. I chose to install python version 3.7.9 instead."

Note that, Python version 3 is required. If you have multiple instances of python installed, mind the PATH settings!

If the python path is NOT specified there, next commands may give the error "command not found". On windows the PATH settings can be checked using Computer, Properties, Advanced, Environment variables.

After installation, the python version can be checked with the following commands:

```
python --version

pip --version
```

You can check whether the most recent pip is installed with the command:

```
pip install --upgrade pip
```

As soon as python is installed correctly, the modules that are required for FortiusAnt can be installed as follows:

```
pip install -r requirements.txt
```

the requirements file is available in code/pythoncode in **<the FortiusAnt folder>**.

3.5 Install USB-driver

3.5.1 Windows

On the system where FortiusAnt is running, the easiest is that TTS is not installed, since the two programs require different USB-drivers which may be conflicting. There are studies to have both programs installed, for sake of simplicity of this installation instruction will describe a FortiusAnt-only installation. **Refer to** 5.4 "Can TTS4 and FortiusAnt coexist?" for more information!

You have to (re)install your trainer as a libusb-win32 device.

Download the libusb driver

- Download software from https://sourceforge.net/projects/libusb-win32/
- Read the wiki, sections download and Device Driver Installation
- Download libusb-win32-bin-1.2.6.0.zip from files/libusb-win32-releases/1.2.6.0/
- Unzip the file

Uninstall TTS-driver (refer to section 5.4 "Can TTS4 and FortiusAnt coexist?" first)

- Open device manager.
- Right click on the device and click "Uninstall". It may be listed as a "Jungo" device (see http://www.tacxdata.com/files/support/Windows10driverissues.pdf - DO NOT RUN TacxDriversSetup.exe!)
- Unplug the trainer, wait 5 seconds, and plug it back in again

Install option 1

- Find it again (usually under other devices>VR-interface)
- Right click and select "update driver software"
- Select "Browse my computer for driver software"
- Select "Let me select from a list of device drivers on my computer"

- Select libusb-win32 devices
- Select ANT USB Stick 2, then OK in the warning, then close Your USB-trainer is now installed as "ANT USB Stick 2" which works, but perhaps is not a very clear name.

Install option 2

- Go to the ..\libusb-win32-bin-1.2.6.0\bin folder
- Start inf-wizard.exe, click next
- Select "VR-Interface" (which is the USB-device you plugged in again), click next
- Specify the name for Manufacturer (Tacx) and Device (VR-Interface), click next
- Store the results in the same folder; this creates a file like VR-Interface.inf
- Complete the installation

Now the USB-trainer is installed as "VR-Interface" (you could have chosen for "Fortius Virtual Trainer").

Technically, it is the same as option 1, but especially if you have two ANT sticks and one Tacx Trainer a named device is nicer. The names will only show up in device manager and ExplorANT and have little significance for the end-user.

3.5.2 MacOS

** to be supplied, since I have no MacOS environment to test

Hints: libusb can be installed using brew install libusb; Get brew if you don't have it already: https://brew.sh/.

3.5.3 Linux – General

Hints:

- Root required
- Refer to AntBridge installation instructions for hints (https://github.com/pepelkod/AntBridge).
- Instructions supplied by FortiusAnt users, since I have no linux environment to test.

3.5.4 Linux Ubuntu 20.04

As provided by @msjnaessens; thanks.

Fresh install of Ubuntu 20.04; installed in Oracle VM VirtualBox 6.1; 2048 MB ram; 4 cpu cores (i7 8750H); 16 MB video memory; installed on MSI GV62 8RC laptop.

```
sudo apt install git

git clone https://github.com/WouterJD/FortiusAnt

sudo apt upgrade python3

sudo apt install python3-pip

pip3 install --upgrade pip

sudo apt-get install python3-pygame
```

```
sudo apt install make gcc libgtk-3-dev libgstreamer-gl1.0-0 freeglut3 freeglut3-dev python3-gst-1.0 libglib2.0-dev ubuntu-restricted-extras libgstreamer-plugins-base1.0-dev ubuntu-dev-tools
```

```
sudo apt install python3-wxgtk4.0

pip3 install -r ./FortiusAnt/pythoncode/requirements.txt

pip3 install --upgrade wxpython
```

```
git clone https://github.com/pepelkod/AntBridge
LOC=/lib/modules/uname -r/kernel/drivers/usb/serial/
sudo mv $LOC/usb-serial-simple.ko ~/Documents
sudo mv $LOC/usbserial.ko ~/Documents
sudo rmmod usb_serial_simple usbserial
sudo apt install libusb-dev
sudo apt install libgoogle-glog-dev
sudo apt install libusb-1.0-0-dev
sudo apt install pkg-config
```

Open terminal in AntBridge folder

make; make;

```
sudo make install
```

```
git clone https://github.com/Tigge/openant
git clone https://github.com/Tigge/antfs-cli
```

```
open terminal in openant folder:
sudo python3 setup.py install
open terminal in antfs-cli folder
sudo python3 setup.py install
```

Now run FortiusAnt:

```
sudo python3 ./FortiusAnt/pythoncode/FortiusAnt.py
```

Done!

3.6 Install ANTdongle

ANTdongles are a lot easier to use than a USB-interface, since they are plug&play; insert the dongle in your computer and the required drivers will be installed automatically.

Dongles from manufacturer=CYCPLUS are reported not working well with FortiusAnt, refer to github FortiusAnt issues (#61, #45 and #65).

3.7 Start FortiusAnt

Now FortiusAnt should be able to operate now. Since we start FortiusAnt without additional command-line variables, FortiusAnt will use the default (best-practice) settings. To start FortiusAnt, you have to make a shortcut, a menu-entry or a command-file, containing the command which is then executed.

<The FortiusAnt folder> contains a folder “StartUp” where you find some command-files, which can be used as an example. When the debug-option is used, logfiles are created in this folder, this will be described later.

“FortiusAnt.bat” contains the following two lines:

```
..\WindowsExecutable\FortiusAnt.exe
pause
```

and “FortiusAnt (exe).bat” contains:

```
..\WindowsExecutable\FortiusAnt.exe
pause
```

Start FortiusAnt by double-clicking the file from Windows-explorer. A “console” is opened and then the FortiusAnt user-interface is started. After completion of FortiusAnt you will get a prompt to press enter (the pause command) so you can see messages in the “console”.

Note that starting FortiusAnt on other operating systems is done in a similar way.

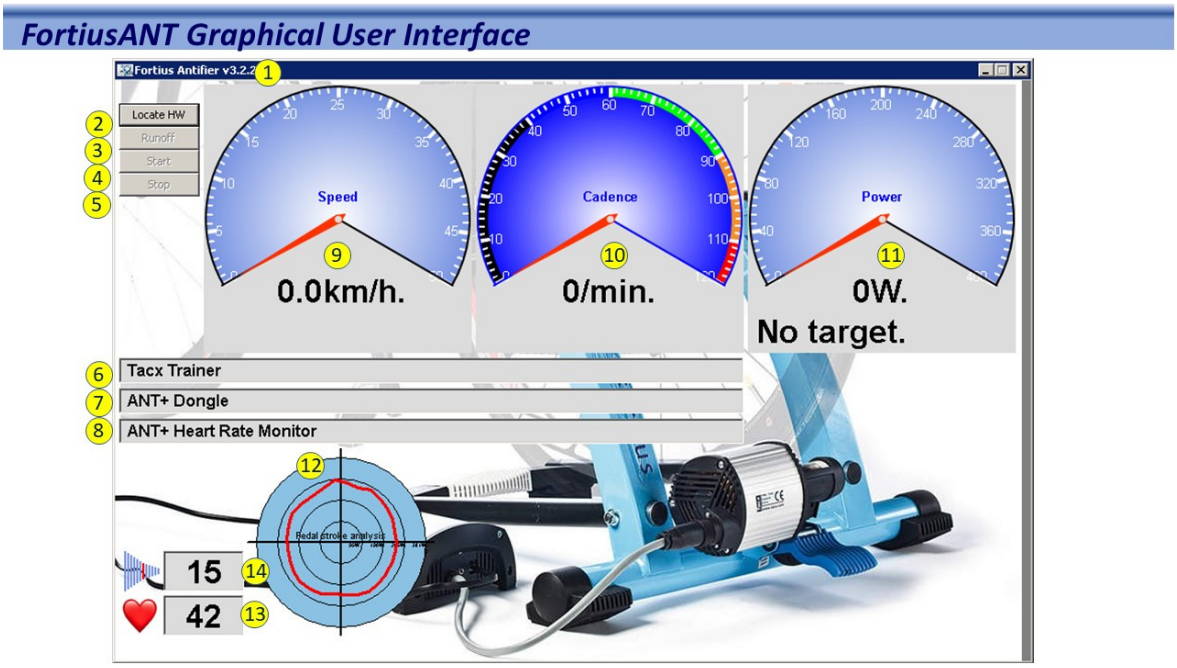
See section 4.2 “Command line” for information on parameters that can be passed on the command-line.

3.8 Check FortiusAnt

When FortiusAnt is started without additional command-line parameters, the following text is displayed in the console:

```
-----------------------------------------------------------------
Hello!
You have started FortiusAnt without command-line parameters.

Therefore, we start with a best practice setting:
    FortiusAnt.exe -a -g -H0 -A

If you want to start without the graphical user interface:
    FortiusAnt.exe -a

For more info, please refer to the wiki on github.
Succes!
-----------------------------------------------------------------
FortiusAnt is open source and can freely be used.

A free gift would be appreciated:
Put yourself on the FortiusAnt map by making yourself known
by leaving a message with name/location/trainer on
https://github.com/WouterJD/FortiusAnt/issues/14

Just for the fun of knowing where we are training.
-----------------------------------------------------------------
```

And then the graphical user interface appears:



When the GUI is shown, you know that FortiusAnt is installed correctly.

The user interface contains the following elements:

1. The window title displays name and version of the software you run
2. <Locate HW> is a button and when pressed, FortiusAnt will search for a Tacx USB-device and an ANT-dongle. When found, the result is displays in (5) and (6), the button is disabled and <Runoff> and <Start> are enabled.
3. <Runoff> activates the user-driven calibration
4. <Start> activates FortiusAnt to bridges USB data to ANT+ and vice-versa; then <Start> is disabled and <Stop> enabled.
5. <Stop> stops the FortiusAnt bridge.
6. Shows what USB-trainer is found
7. Shows what ANT-dongle is found
8. Shows what heartrate is used
9. Displays the speed of the bicycle wheel (returned by the Tacx trainer)
10. Displays the cadence of the pedals (returned by the Tacx trainer)
11. Displays the power as returned by the Tacx trainer.
    Also, the target is displayed as requested by the CTP.
12. Displays the Pedal Stroke Analysis, as calculated by FortiusAnt.
13. The heartrate.
14. The virtual gearbox; displayed only in Grade-mode.

4. Operating instructions

4.1 The main functions of FortiusAnt and the head unit.

After that FortiusAnt is started (see 3.7 Start FortiusAnt) you see the user interface with the buttons

- Locate HW
- Runoff
- Start
- Stop

4.1.1 Locate HW

Checks for the presence of USB-trainer and ANT-dongle. If successful, results are displayed and the button is disabled. For more info read section 4.3 “Locate HW”.

4.1.2 Runoff test

To ensure comparable training sessions, the trainer should provide the same relative resistance each time

1. Aim for about 7 bar (100psi) in tire when cold
2. Warm up for 2-3 minutes to warm tire
3. Increase speed and pass the 40 km/hr speed, then stop pedaling and let wheel slow down
4. Ideally, the wheel should stop after 7 seconds from 40kph

If the rundown is too short, the role may be too tight and if the rundown is too long, the role may be too loose. In that case, adjust the role and retry the rundown test.

4.1.3 Start

Pressing this button starts FortiusANT to listen to the FE (Tacx USB-trainer) and CTP (ANT-dongle, Zwift, Trainer Road, Rouvy, …) and exchange info between them.

FortiusANT starts to calibrate the trainer (if supported and -n is not specified). Calibration means that the brake rotates the wheel at 20 km/hr and returns the resistance found. As soon as the resistance is constant, the calibration stops. The calibration time is at least 30 seconds (warming up the tire) and stops when the resistance value is constant.

Note that the calibration starts when you turn the pedal as if starting to cycle, which is the only physical action to take. Note that, starting the motor automatically would be a risk for physical injury, therefore the confirmation with a pedal-kick is required.

When calibration is started, do not pedal, the process completes automatically.

After calibration, the Fortius is ready for training.

4.1.4 Stop

Pressing this button stops the currently running process (runoff, calibration or operational mode).

4.1.5 Buttons on the Tacx head unit

There are four buttons: Cancel, Enter, Up, Down.

- **Cancel** is active in all modes.
- If not in an active mode, **Up**/**Down** navigate through the menu, **Enter** activates the selected button and **Cancel** exits FortiusANT.
- In runoff or manual mode, **Up**/**Down** modify the required power with ±50Watt. **OK** *) resets the power to the initial value of 100Watt.
- In manual grade mode, **Up**/**Down** modify the slope with ±1degree. **OK** *) resets the slope to the initial value of 0 degrees.
- In normal resistance mode, **Up**/**Down** modify the resistance of the Fortius by ±10%. **OK** *) resets the resistance is reset to the initial value of 100%.

*) Tacx has a variety of trainers with different head units, not all head-units have an OKbutton.

4.2 Command line

FortiusANT is started with a command (see 3.7 Start FortiusAnt). In addition to the examples shown there, parameters can be passed on the command-line:

```
FortiusAnt.py [-h] [-a] [-A] [-d DEBUG] [-g] [-H HRM] [-m] [-M] [-n] [-p FACTOR] [-P] [-s] [-t TACXTYPE]
              [-u] [-x]
```

Basic arguments:

-h	Show a help message and exit.
-a	Automatically start; “Locate HW” and “Start” if the required devices were found.
-g	Run with graphical user interface.

Advanced arguments:

-A	Pedal Stroke Analysis.
-G offset/grade	Modify the requested grade with an offset and a factor; see section 2.6.4 “PowerCurve for i-Flow (T1901-T1932)”
-H HRM	Pair this Heart Rate Monitor (0: any, -1: none).
-m	Run manual power (ignore target from ANT+ Dongle).
-M	Run manual grade (ignore target from ANT+ Dongle). When -m or -M is specified, a .tcx file will be created for every exercise.
-n	Do not calibrate before start.
-p FACTOR	Adjust target Power by multiplying by this factor for static calibration.
-P	Power mode has preference over Resistance mode (for 30 seconds). Run Zwift or Rouvy to ride a route, the target is transmitted as a grade and you see where you ride. In parallel run Trainer Road to do a structured training, the target is transmitted as power. FortiusANT “listens” to TR and transmits power and cadence to both. Now you can do a structured TR-training in the virtual world of Zwift or Rouvy.

-r	Target Resistance = Target Power (to create power curve)
-t TACXTYPE	Specify Tacx Type; e.g. i-Vortex, default=autodetect.
-u	Uphill only; negative grades are ignored to avoid motor-drive (on Fortius).
-x	Export TCX file to upload into Strava, Sporttracks, Training peaks. -x is implicit in manual (grade) mode.

Developer arguments:

-d DEBUG	Create logfile with debugging data (see section 4.6 “Debugging FortiusANT”).
-s	Simulated trainer to test ANT+ connectivity.

Examples:

FortiusAnt.py	FortiusANT is started without user-interface, -g -a -A -H0 are assumed.
FortiusAnt.py -g -a	FortiusANT is started with user-interface and starts automatically.
FortiusAnt.py -g -m	FortiusANT is started with user-interface. No **[CTP]** is required, power can be set using the console. Although intended for interface testing, you could do a manual ride this way.
FortiusAnt.py -g -M	Same as -m but now the slope-grade can be adjusted.
FortiusAnt.py -g -s	FortiusANT is started with user-interface. No **[FE]** is required, automatic response to **[CTP]** is generated. This is intended for interface testing.

Values for parameters:

DEBUG	Is a binary flag list what to write to the logfile, 0=nothing, 127=everything. The values below can be added together. No = 0 Application = 1 Function = 2 antDongle = 4 usbTrainer = 8
FACTOR	Correction factor 0.9 ... 1.10
HRM	The device ID of the Heart Rate Monitor to be used. 0: pair with first found, -1 do not pair at all.

4.3 Locate HW

When the “Locate HW” button is pressed, the following happens:

Find ANT+ dongle

A check is done whether an ANT+ device with DeviceID 4100, 4104 or 4105 is found. If found, an attempt is made to use the dongle, if in use another dongle will be searched for.

FortiusANT always needs an ANT-dongle, unless -m or -M is specified in that case you can set power or slope with the trainer’s head unit buttons.

The following messages can be displayed:”

> No (free) ANT-dongle found
>
> Using <manufacturer> dongle

or messages indicating what interface-error occurred.

Find Tacx Trainer

Then a check is done whether a Tacx device is used with one of the DeviceID’s as listed in section 2.3 “Tacx trainers”.

The following messages can be displayed:”

> No Tacx trainer found
>
> Connected to Tacx Trainer T<DeviceID>

or messages indicating what interface-error occurred.

Note that, when the -s command-line parameter is specified, the following message is displayed:

> Simulated Tacx Trainer to test ANT-interface

Note that, when the -t i-Vortex command-line parameter is specified, the following messages are displayed:

> Pair with Tacx i-Vortex and Head unit (pairing can take a minute)
>
> Tacx i-Vortex paired: %s, Head unit: %s

Heartrate monitor

Old Tacx trainers paired with a heartrate monitor (HRM) and passed the heartrate through the USB-interface to **[TTS]**. Even though this option is supported on the **[FE-C]** ANT+ interface, it is not used by **[CTP]** since this software pairs with a HRM itself.

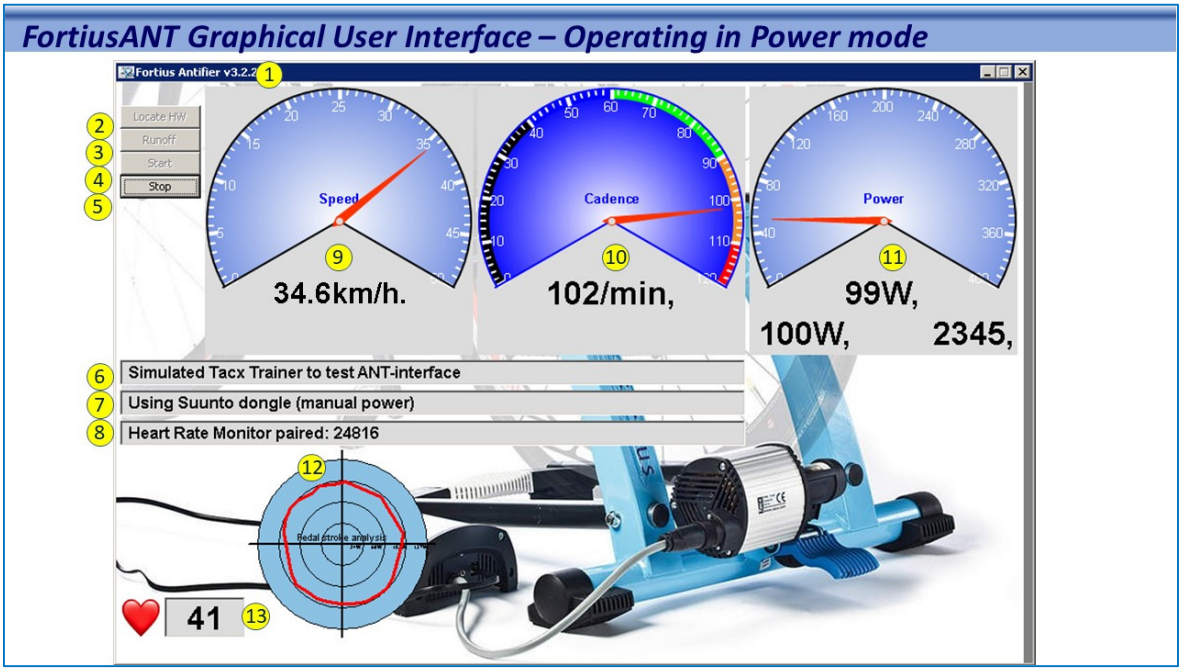
The FortiusANT display shows the heartrate and therefore the following options exist:

- No command-line option: use the heartrate from the Tacx trainer
- -H0: pair with an ANT+ HRM”, use the first HRM that is found
- -Hnnnnn: pair with the ANT+ HRM with DeviceID=nnnnn
- -H-1: no HRM.

The following messages can be displayed:”

> Heartrate expected from Tacx Trainer
>
> Heartrate expected from ANT+ HRM
>
> Heart Rate Monitor paired: <DeviceID>

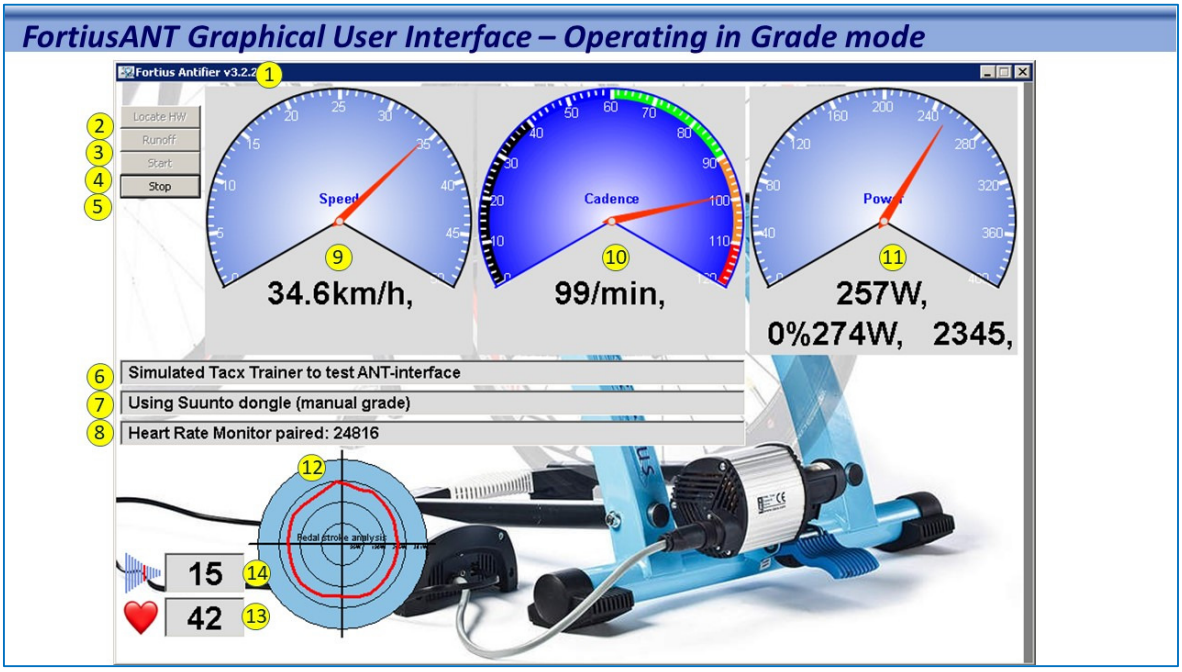
4.4 The FortiusAnt display in power mode



The elements on the screen are explained in section 3.8 “Check FortiusAnt”.

In addition to the standard display, under the power gauge (11) the actual power (99W), the requested power (100W) and the resistance set in the trainer (2345) are displayed.

4.5 The FortiusAnt display in grade mode



The elements on the screen are explained in section 3.8 “Check FortiusAnt”.

In addition to the standard display, under the power gauge (11) the actual power (257W), the requested grade (0%) and resulting required power (274W) and the resistance set in the trainer (2345) are displayed.

Here you also see the “digital gearbox”. In addition to the gears on you bicycle imagine you have a second drive train with a 15x15 ratio. So, the displayed situation is neutral. When you press Up/down the ratio is changed with ±10%, increasing and decreasing the required resistance on the bike. The digital gearbox shows the number of teeth on your digital cassette.

If your **[CTP]** requires to ride uphill with a slope of +10% and you have a high wheel-speed (because that is how the Fortius works well, see section 5.1 “Low cadence on Fortius”) you would need a high power. If that required power is higher than you can produce, you can reduce the required power using the down-button without reducing speed. If you are using Zwift you will see that the difference in Speed in Zwift (based upon power) and the speed displayed by FortiusAnt will increase).

The digital gearbox is not active in power-mode, because if 100W is required, 100W you must give. At a high wheel speed, the resistance is already calculated accordingly.

Thanks to Erik OT for the magnificent suggestion!

4.6 Debugging FortiusANT

FortiusANT has a debugging flag -d specifying what output must be written to the logfile.

The options are defined as a decimal number, being the sum of the options desired:

No logfile	= 0x00	= 0
Application	= 0x01	= 1
Function	= 0x02	= 2
antDongle	= 0x04	= 4
usbTrainer	= 0x08	= 8
Multiprocessing	= 0x10	= 16
JSON	= 0x20	= 32

If you want all options activated, -d127 will do the job.

To log the ANT+ calls: -d4 and if you want ANT+ and the USBtrainer to be logged -d12 does the job.

If -d32 is specified, a JSON file will be created with the relevant data from FortiusANT. This may be helpful for further analysis.

Although books could be written to explain the content of logfile and JSON file, I leave it to the user's creativity to understand the content. If you want to interpret the JSON file,
"FortiusANT JSON Analysis.xlsx" in the supportfiles section on github may be helpful.

5. Questions and special situations

5.1 Low cadence on Fortius

One of the limitations of the Fortius is the resistance at low wheel-speed, which implies a low rotational speed of the brake.

If you have a high wheel-speed, the Fortius can realize a resistance of 1000Watt. This works fine and is good to train your power.

At a low wheel-speed however, the brake does not work very fine and therefore it needs some thinking to train “Uphill training at 20%”, requiring high power and low speed. You would be inclined to reduce gears as you would do in real world.

What I do is always use a high gear, causing a high wheel-speed, and drop the cadence without shifting. For example, if you want to simulate a steep uphill, requiring 500Watt you still go in the highest gear and use a cadence of 50 rpm.

FortiusAnt will reduce resistance for the trainer and the result is that you ride at 30km/hr with 500Watt and 50 rpm. Just ignore the wheel-speed!

Note that modern trainers have a direct drive and no wheel; there the wheel is ignored completely.

5.2 Zwift speed does not match Garmin

If you ride your bicycle on your trainer; the speed that Zwift displays does not match Garmin's speed.

Your Garmin receives the speed from the bicycle and displays the real speed of the wheel on the trainer. (Probably you realize, with the same effort, you would not achieve that speed in the real world)

Zwift receives the realized power from the trainer and uses its own algorithms to conclude what speed you are riding: If riding on a flat surface, without headwind, the simple formula would be: speed=power/resistance. But when you ride uphill and you take air-resistance into account, the formula is far more complicated.

Interested in the power you need to ride?

- https://www.fiets.nl/2016/05/02/de-natuurkunde-van-het-fietsen/
- https://www.gribble.org/cycling/power_v_speed.html

The gribble formulas are used by FortiusAnt to convert grade to power, see also section 2.5.2 “Grade mode”.

5.3 Average speed in Trainer Road

Jerome uses TrainerRoad and notices that during a ride his average speed is 20km/hr and his friend is going at 40 km/hr. How is that possible?

Imagine that TrainerRoad requires to ride with a Power of 200Watt, the two riders have the same bike and the same trainer and have selected the same gear-ratio.

If Jerome choses to ride with 50 rpm, his speed will be low, and the trainer will increase the resistance so that the required power of 200Watt is realized.

If his friend rides at 100 rpm, his speed will be higher than yours and the trainer will decrease the resistance so that the required power of 200Watt is realized.

Note that power = resistance * speed! Jerome rides at half speed of his friend, but with a higher resistance and hence both athletes produce the same power at a different speed and different distance. Note therefore that, on a **[FE]** speed and distance are irrelevant, time, power and cadence are the deciding factors.

For you it's a pity that your Strava statistics for the end-of-year applause run behind; but having read the explanation - that would be a lesser issue.

The situation in real world is different; if you go for a ride together time, speed and distance will be the same and therefore you will come home with the same average power. Difference choice in gears will change the resistance and cadence which is less observed in the Strava results.

Why would you ride at 50rpm or 100rpm?

When TrainerRoad demands 200Wattt it may instruct you (through the displayed texts) to ride at a high or a low cadence, depending what purpose is intended. It is not correct to say that the speed is not relevant: it may be that either you or your friend did not do the intended training (high power or high force).

5.4 Can TTS4 and FortiusAnt coexist?

[TTS] is Tacx' own training software which uses it's own USB-driver and/or ANT+ interface. The installation instruction suggests to de-install (for simplicity), see 3.5 "Install USB-driver".

The following information is of interest for everybody who would like to work with **[TTS]** and other **[CTP]**'s.

iepuzaur: I am running on Windows10 TTS4 and FortiusANT, at first I also thought I have to uninstall the Jungo driver TTS uses, but then I noticed you can have both drivers installed and there is no issue on my system.
Of course, you cannot use them concurrently, if you start TTS4 it will use the Jungo driver, while when you start FortiusANT the libusb drive will be selected.
So as long as you do not intent to use them at the same moment, the drivers shall not conflict (at least this is my case). (Thanks to @iepuzaur, june 2020).

Note however: If your TTS4 software is already installed and registered then that should work. If it isn't, it won't as the registration servers have been switched off. The only software that works without registration is Fortius v2.02.
Never uninstall TTS4 if you have a working registered copy as there is no way to reinstall it and re-register it. (Thanks to Shaun Murray, @aegisdesign, sept 2020).

frenske8: I have been using FortiusAnt for 2 months now in combination with the Tacx Desktop App (Windows 10). It works fine! The Tacx Desktop App connects via ANT with ANT-ID 57591 (identified as Tacx Neo2T!).

Incidentally, I also own the 4iii Viiiva HRM with ANT to BT bridge, but I expect the bridge to work in one direction only, so not from BT to ANT. So, using BT via the 4iii HRM is unlikely to provide a solution, I failed. However, this is not necessary, because FortiusAnt can work directly via ANT with the Tacx Desktop App.

By the way: I'm using FortiusAnt and Tacx App on two different laptops, because I sometimes want to use Tacx TTS4 (installed on the same laptop as Tacx Desktop App). I failed to get FortiusAnt and TTS4 to work on the same laptop.

Note that, TTS4 and Tacx Desktop App are 2 different applications. TTS 4 is outdated software, but like the Tacx Desktop App. A few years ago, Tacx replaced TTS4 with the Desktop App and they changed the revenue model: with the Desktop App they introduced the subscription model (monthly / annual fee). TTS4 uses real life videos that had to be purchased separately.

FortiusAnt also works with TTS4 (with 2 laptops), but that is not necessary, because TTS 4 also works via the USB connection of the Fortius trainer (with the Jungo USB driver that is installed by TTS4). It is **tricky to uninstall the jungo driver** (to try to run TTS4 and FortiusAnt on 1 laptop), because Tacx has ended the support of TTS4 and reinstallation of TTS4 is therefore no longer easily possible (because Tacx servers for authentication are out of operation).
(Thanks to @frenkse8, August 2020)

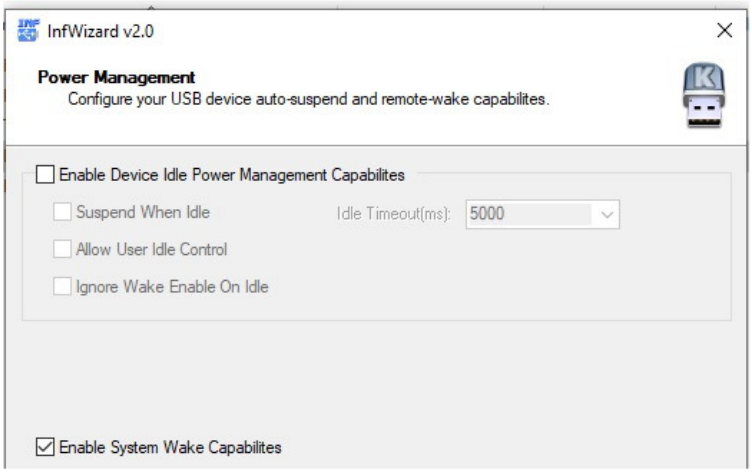
5.5 Tacx head unit with firmware to be loaded

Some head units (0xe6be, Old "solid blue" Fortius) do not have firmware; the firmware must be loaded.

When FortiusANT discovers a 0xe6be head unit, the firmware is loaded automatically and (after a wait time of 5 seconds to let the head unit reboot) a 1942 head unit is expected to be present. The following messages will be displayed, before/after loading the firmware:

```
....Stop CPU
....Start CPU
```

Some users have reported that the head unit is not found after the reboot and tell that this can be resolved with the following settings in the Power Management Options of the libusbK inf wizard:



5.6 Fortius without cadence sensor

Although it appears that the cadence sensor is for information purpose only, the Fortius (and perhaps other Tacx trainers as well) does not work if no cadence sensor is connected and/or no cadence is measured. Note that the magnet must be well placed compared to the sensor.

5.7 Two ANTdongles – disturbed communication

Two laptops are used:

- The left laptop is connected to Tacx Fortius, is running FortiusANT and has the ANTdongle at the left side.
- The right laptop is running Zwift and has the ANTdongle at the right side.

Sometimes the connection is lost and FortiusANT does not display the required grade as transmitted by Zwift or Zwift appears not to receive the power from FortiusANT.

After exchanging the dongles, so that these are close to each other, the transmission errors were gone.

Perhaps the presence of two computers between the two dongles disturb the ANT-transmission; note that ANT is low-power.

So in case of transmission errors; check whether there may be interference.

5.8 Tacx returns insufficient data

This error can be displayed and has different reasons:

- FortiusANT reads too often from the headunit and therefore the headunit is not ready to provide the next buffer
- Loose or faulty cabling

- Other hardware errors

The following message is given

```
Tacx returns insufficient data, len=XX
To resolve, try to run without Pedal Stroke Analysis
```

When pedal stroke analysis is active (-A flag), FortiusANT reads more often from the headunit and this MAY cause the error.

or

```
Tacx returns insufficient data, len=XX
To resolve, check all cabling for loose contacts
```

FortiusANT is developed, using a 1932 headunit on a Tacx Fortius. In that environment, the message is given very seldomly and the built-in retry (4 times) appears to works. Some systems, even without PDA, show the short message error more often which needs further investigation.

5.9 Sudden drop of requested power

If you are experimenting with multiple CTP's (for example Zwift on one system and Trainer Road on another) it may occur that multiple CTP's send commands to FortiusANT. FortiusANT listens to both and changes behavior as requested. If one CTP asks to perform 100Watt and the other sends Grade=-20% the required power will bounce between the two requestors.

This can be intended, for example you ride a structured training in Trainer Road and entertain yourself with a virtual ride in Zwift. In that case specify -P so that the ERGmode commands from Trainer Road prevail over the grade-mode requests in Zwift.

Of course, if two CTP's both send ERGmode requests (100Watt and 200Watt) then you'd better switch off one of the CTP' 😊.