Validation of a software controlled Pedelec controller[*]

Ralf Boeckhorst [†] Daniel Ginster [‡] Alexander Asteroth [§]

Bonn-Rhein-Sieg University of Applied Sciences

Abstract

This paper describes the development of a Pedelec controller whose performance level (PL) conforms to European standard on safety of machinery [9] and whose software is verified to conform to EPAC standard [6] by means of a software verification technique called model checking.

In compliance with the standard [9] the hardware needs to implement the required properties corresponding to categories "C" and "D". The latter is used if the breaks are not able to bring the velomobile with a broken motor controller to a full stop. Therefore the controller needs to implement a test unit, which verifies the functionality of the components and, in case of an emergency, shuts the whole hardware down to prevent injuries of the cyclist. The $\mathrm{MTTF_d}$ can be measured through a failure graph, which is the result of a FMEA analysis, and can be used to proof that the Pedelec controller meets the regulations of the system specification.

The analysis of the system in compliance with [9] usually treats the software as a black box thus ignoring its inner workings and validating its correctness by means of testing. In this paper we present a temporal logic specification according to [6], based on which the software for the Pedelec controller is implemented, and verify instead of only testing its functionality. By means of model checking [1] we proof that the software fulfills all requirements which are regulated by its specification.

Introduction

The number of electrically power assisted cycles (aka EPAC or Pedelecs) operated all over Europe is continually growing (e.g. [12, p.15]). According to ZIV (Zweirad-Industrie-Verband e.V., [15, p.18]) the number of bikes sold in 2011 in Europe amounts to 900,000 and in the last years more and more velomobiles are also equipped with electrical assistance.

European Standard [6] defines what constitutes a Pedelec, i.e. "*electrically power assisted cycles of a type which have a maximum continuous rated power of 0.25 kW, of which the output is progressively reduced and finally cut off as the vehicle reaches a speed of 25 km/h, or sooner, if the cyclist stops pedaling*" [6, p. 6]. This standard defines requirements as well as test methods aiming to assure quality and safety of the vehicle.

[*]`This is a revised version of our paper initially presented at the 7th International Velomobile Seminar and was last edited on September 27, 2012.`

[†]`ralf.boeckhorst@inf.h-brs.de`

[‡]`daniel.ginster@inf.h-brs.de`

[§]`alexander.asteroth@h-brs.de`

While the test procedures try to cover all imaginable misuses by the operator (i.e. the cyclist) to assure safety in all foreseeable situations, the same is not possible for embedded systems such as digital Pedelec controllers. Therefore the hard- and software in the Pedelec controller can cause unpredictable hazardous situations.

While a person is riding a regular EPA-cycle and is confronted with a malfunctioning motor control, one is able to jump of the bike, the same problem causes serious trouble with an EPA-velomobile, since usually you cannot just get out easily. Therefore it is important to assure the functional safety of the controller and to prevent all possible sources of malfunctions.

Just recently German VDE (Association for Electrical, Electronic & Information Technologies) claims that 95 percent of all electric bikes were sold without any safety check [14] and for augmented safety standards for electric vehicles in general.

Related Work

For the assessment and reduction of risks of dangerous failures and to increase the safety of Pedelec controlled bicycles the European Committee for Standardization introduced two standards [5] and [6], which ensure the safety of the bicycle parts of the Pedelec and the functional parts through test procedures and regulations. These regulations and test conditions comprise electromagnetic compatibility, maximum continuous rated power, maximum speed, wiring, handbreaks and pedals. The problem with standards like [5] or [6] is that they only give general regulations, but say little about which components should be used in order to call the Pedelec controlled bicycle “safe”.

An additional problem is that the general approach by utilizing [6] is not enough, because one can argue that a Pedelec can also be interpreted as a small machine and as such has to be checked against standards for machinery safety.

The industry has multiple standards to define what makes a machine safe for usage and which dangers and injuries could occur. Figure 1 shows all the needed details what machinery is and which components it usually contains. Figure 2 addresses the components used in the layout of our proposed Pedelec unit.

The operator has the ability to interact with any part of the machine and has the risk to harm oneself in the process. At this point the term of “residual risk” applies. The developer can and should not expect, that the operator of the future product knows every single possible hazardous situation and the suitable reaction to it. Therefore the risk of having no protective measures has to be reduced to an acceptable level with which developer and operator are satisfied.

For the assessment and reduction of risks of dangerous failures and to increase the safety of machinery the European Committee for Standardization introduced standards [8, 9, 10, 11, 7], from which in this paper we only address [9] directly to present our results.

At this point the reader has to remember, that the following parts deal with European standards and that only the regulations to produce a safe Pedelec are addressed. For other cases of application some important details could have been lost in the process.

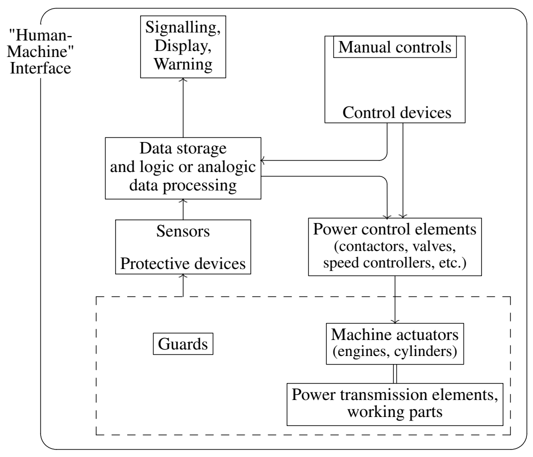

"Human-Machine" Interface
Signalling, Display, Warning
Manual controls
Control devices
Data storage and logic or analogic data processing
Sensors
Protective devices
Power control elements (contactors, valves, speed controllers, etc.)
Guards
Machine actuators (engines, cylinders)
Power transmission elements, working parts

Figure 1: Schematic representation of a machine (based on [8, A.1])

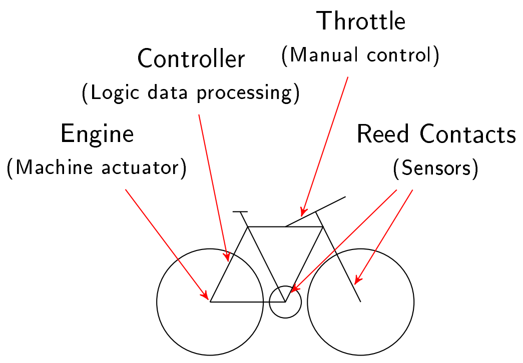


Figure 2: Components used in our layout

Methods

Because Pedelec controlled bicycles are machinery, it is obvious that they fall within the scope of regulations for the safety of machinery, too.

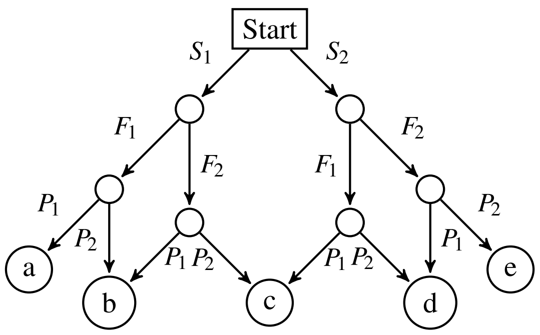
As stated in the last section utilizing just the standard [6] cannot ensure the safety of a Pedelec controlled bicycle, because as a basic machinery it falls in the scope of [9] and has to be reviewed as such.

If the result of the risk assessment of [11] is an insufficient protection of the operator and at the same time the safety function is dependent on a controlling-unit, the risk of a dangerous failure has to be reduced and the standard [9] applies. A safety function is a procedure of the controller to ensure the safety of the operator like switching the power supply off or shutting down the motor controller. For each needed safety function the required Performance Level (PL_r), the reached Performance Level (PL), the needed category to meet the requirements of the PL_r, the mean time to a dangerous failure ($MTTF_d$) and the average diagnostic coverage of the safety function (DC_{avg}) need to be calculated.

For the calculation of the PL_r figure 3 shows a simplified procedure with “a” as the lowest PL_r-level and “e” as the highest. In situations where multiple selections can be applied (like S1 and S2 at the same time), both paths are selected and the highest resulting PL_r must be taken at the end as the final result.

An easy example would be the safety verification of garden shears. The wounds can be severe, but at the same time the frequency of exposure to this hazard is rare, because plants do not grow that fast. The chance of danger prevention is possible under the condition that protective gloves or suitable clothings are used. So the PL_r for garden shears would be “c”, because of the resulting path S2 $\rightarrow$ F1 $\rightarrow$ P1.

To calculate the category that matches the given PL_r requirements with “B” as the lowest and “4” as the highest level we take all the possible categories from [9, 6.2.3 to 6.2.7] and make a “worst case” selection. The results are presented in table 1.



Wound severity
S1: light (usually reversible injuries)
S2: severe (usually irreversible injuries including death)

Frequency and/or duration of exposure to hazard
F1: rarely and/or the time of exposure to hazard is short
F2: frequently and/or the time of exposure to hazard is long

Chance of danger prevention or harm limitation
P1: possible under certain conditions
P2: nearly impossible

Figure 3: Simplified procedure to calculate PL_r [9, A.1]

Required category		Maximum realizable PL_r		Characteristics
B	⇔	b		
1	⇔	c		
2	⇔	d	with	category 2 requirements
3	⇔	d	with	category 3 requirements
4	⇔	e	with	category 4 requirements

Table 1: The categories and their maximum possible PL_r-values

This means that with our example of garden shears the required category would be “1”, because “c” is the maximum realizable PL_r. Of course we could choose a higher category than nessessary, but this would imply stricter regulations and a different controller layout.

The resulting controller setup, that needs to be realized to meet the safety function regulations, are grouped into 3 layouts, that are displayed in the figures 4, 5 and 6.

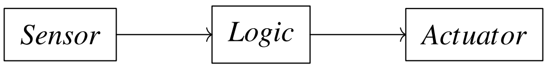


Figure 4: Layout of a category B and 1 controller [9, pp. 6.2.3, 6.2.4]

The layout of a controller that fits into category “B” and “1” contains only a small amount of components that consists of sensors, logic and actuators, because the hazard that is based on a dangerous failure of the system is so small, that simple protection mechanisms are enough to ensure the safety of the operator.

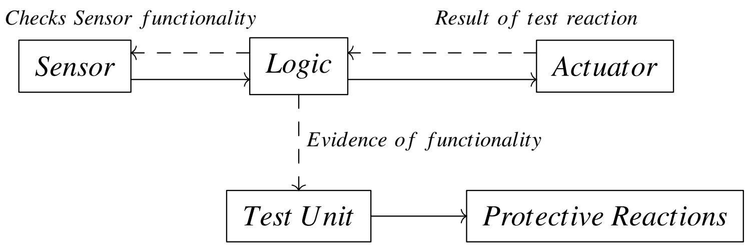


Figure 5: Layout of a category 2 controller [3, p. 243]

While dealing with hazards that have a larger impact on the operator the layout of figure 4 is not enough to ensure safety, because when the logic malfunctions the whole safety function could be missing out. So a test unit is added to periodically check the sensor functionality and the test results of the actuator by getting the test reactions delivered from the main logic to reason the evidence of functionality of the controller.

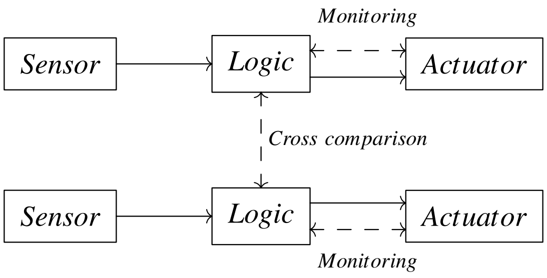


Figure 6: Layout of a category 3 and 4 controller [9, pp. 6.2.6, 6.2.7]

If the hazards are so threatening that a simple test unit is insufficient to ensure enough safety for the operator the whole sensor-logic-actuator parts needs to be doubled to get double sampling and double precision on the test results through cross comparison.

The $MTTF_d$-value represents the average time that a mechanic, pneumatic or electric part takes until it produces an error. While there are manufacturers that provide these numbers in their data sheets of their products, others use different systems like the mean time between failures (MTBF) or the cycles until 10 percent of the given parts are malfunctioning (B_{10d}), from which all can be translated into a $MTTF_d$-value. Table 2 shows all the possible ranges for the $MTTF_d$.

Label	Scope
not appropriate	0 years $\leqslant MTTF_d <$ 3 years
low	3 years $\leqslant MTTF_d <$ 10 years
medium	10 years $\leqslant MTTF_d <$ 30 years
high	30 years $\leqslant MTTF_d \leqslant$ 100 years
not permitted	100 years $< MTTF_d$

Table 2: Scaling of the $MTTF_d$-value [3, p. 53]

The DC-value represents the diagnostic coverage with which hazardous situations or malfunctions of the system are detected. A DC-value of less than 60 percent is said in [9] to have no diagnostic coverage at all. Table 3 shows the required ranges.

Label	Scope
no	$DC < 60\ \%$
low	$60\ \% \leqslant DC < 90\ \%$
medium	$90\ \% \leqslant DC < 99\ \%$
high	$99\ \% \leqslant DC$

Table 3: Scaling of the DC value [9, Table 6]

Having presented all the necessary methods and definitions we can now utilize them to show, that the used controller layout and the given parts can assure the safety of the operator. This is done by validating that the resulting PL is greater or equal to the PL_r in the meaning of providing enough safety for the operator. For this we use the "Parts-Count" method proposed by [9], in which the developer takes the $MTTF_d$-values of all components and calculates the System-$MTTF_d$ by considering their amount. With the System-$MTTF_d$ and the DC_{avg} as the average of the DC-values we are now able to calculate the PL of our safety function. This is done in the next section.

Unfortunately [9] regulates only what is necessary to build hardware for safe machinery and what constitutes a suitable software developing process. The software itself will only be tested by simulation and real life tests. We want to take this a step further and verify that our software is safe at any time by fulfilling its specification in every reachable state. For this we will use a technique called model checking and arcade to solve the model checking problem. arcade is based on [mc]Square, a model checking tool developed at Embedded Software Laboratory, RWTH Aachen [13]. arcade can check the actual microprocessor's binary code against a specification stated in the temporal logic CTL.

The software related part of the specification for EPA-cycles is briefly worded, because the requirements are mostly based on hardware aspects. The only software requirement is "which [. . .] output is progressively reduced and finally cut off as the vehicle reaches a speed of 25 km/h, or sooner, if the cyclist stops pedalling." [6, p. 6].

We use an external clock quartz crystal with 32.768 kHz and a 8-bit timer, which delivers a signal every 0.0078125 seconds. Instead of calculating the precise speed, we define time ranges and calculate once the `time_kph` for 6, 21, 22, 23, 24, and 25 km/h. After each full revolution the software compares the current time (`time_period`) against these values. In every time interval the resulting actions are selected according to the specification.

For software verification we need to transform km/h into a more suitable unit of measurement to represent the actual speed. Since our vehicle's wheels have a circumference of 1.46 m, 25 km/h correspond to a period of 0.21024 seconds per revolution, resulting in 26 ticks per period approximately. If less than 26 clock ticks are counted in one revolution the vehicle is driving faster than 25 km/h. For model checking the condition of cutting of the output at 25 km/h can thus be presented in temporal logic as the following temporal logic formula:

$$\mathbf{AG}\ ((\text{time\_period} \leqslant \text{time\_kph25}) \rightarrow \text{motor\_support} = 0)$$

has to hold in every state
For all paths from the current state
Then motor assistance must be turned off
If the current time of a wheel revolution is smaller or equal to 26

The specification also requires, that the motor assistance is progressively reduced before it is cut off. After 21 km/h the motor support will be stepped down into four portions. As an example we will show the formula for the step 23 km/h where the motor assistance is 50 percent approximately with the value of the variable `motor_support` as 128. We are working with a 8-bit variable, for which the values "0" and "255" mean no and full motor support respectively, which leads to 0 V and 5 V output voltage.

$$\mathbf{AG}\ ((\text{time\_period} \leqslant \text{time\_kph23}) \rightarrow \text{motor\_support} \leqslant 128)$$

has to hold in every state
For all paths from the current state
Then the motor assistance cannot be bigger as 2.5 volt
If the current time of a wheel revolution is smaller or equal to 29

In Germany it is allowed to drive an EPAC with a starting aid up to 6 km/h without pedalling. We choose to include this feature into our building process. This starting aid can be presented in temporal logic as:

$$\mathbf{AG}\ ((\text{time\_period} < \text{time\_kph6} \land \text{foot\_pedal} = 0) \rightarrow \text{motor\_support} = 0)$$

has to hold in every state
For all paths from the current state
Then the motor assistance must be off
If the current time of a wheel revolution is smaller than 112 and no pedaling occurs

For the detection of a full revolution we use a permanent magnet on one wheel and a reed contact connected directly to the controller. We also use reed contacts to determine, if the cyclist is pedalling forward.

Every time a signal from the foot pedal occurs we set the value of the variable `foot_pedal` accordingly. Between two wheel revolutions the pedal must signal forward movement or there will be no motor assistance.

Results

The resulting PL_r for the Pedelec controller is “d”, because road traffic wounds can be severe and the exposure to this hazard is frequently occuring, but the chance of harm limitation is possible under certain conditions, when the handbrakes are stronger than the motor torque.

Because our controller corresponds to category “2” requirements, it has to utilize a test unit that audits the main units actions. Figure 5 shows the resulting layout that is required to build a safe Pedelec unit.

The system’s resulting $MTTF_d$, the used parts to build the controller implementing the safety functions and the DC_{avg} can be read off table 4.

Component	Count n	$MTTF_d$	$\frac{1}{MTTF_d}$	$\frac{n}{MTTF_d}$
Sensor				
Reed Switch	2	2391 [4, Section 14.1]	0.000417	0.000834
			$\Sigma(\frac{n}{MTTF_d})$	0.000834
			$MTTF_d$	1199
			DC	0.9
Logic				
ATmega16	1	1929 [4, Section 5.1]	0.000518	0.000518
LED	1	22831 [4, Section 6.1]	0.000044	0.000044
Ceramic Capacitor	2	4566 [9, Table C.4]	0.000219	0.000438
Carbon Film Resistor	1	22831 [9, Table C.5]	0.000044	0.000044
Quartz Crystal	1	91 [4, Section 19.1]	0.011038	0.011038
			$\Sigma(\frac{n}{MTTF_d})$	0.012082
			$MTTF_d$	83
			DC	0.9
Actuator				
Motor	1	139 [4, Section 12.1]	0.007199	0.007199
			$\Sigma(\frac{n}{MTTF_d})$	0.007199
			$MTTF_d$	139
			DC	0.9
Test Unit				
ATmega16	1	1929 [4, Section 5.1]	0.000518	0.000518
LED	2	22831 [4, Section 6.1]	0.000044	0.000088
			$\Sigma(\frac{n}{MTTF_d})$	0.000606
			$MTTF_d$	1650
			DC	0.9
			$MTTF_{d,avg}$	50
			DC_{avg}	0.9

Table 4: "Parts-Count" and DC_{avg} results

The last step is the verification that the PL-value meets the PL_r requirements. This is done as shown in figure 7 by selecting the System-$MTTF_d$ on the x-axis and reading the resulting PL from the y-axis. The result is a PL of “d” which is equal to the PL_r value of "d". This means that the safety functions implemented by the hardware ensure the safety of the operator.

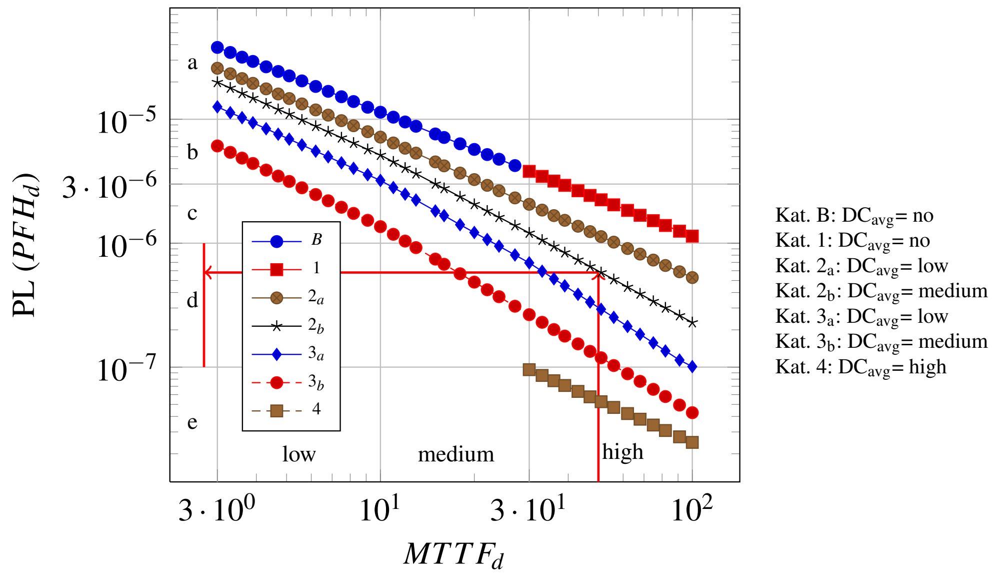


Figure 7: Calculating resulting PL by selecting 50 years on the x-axis

After proving the hardware fulfilling the requirements of the safety of machinery standard, the last step is to show that the software satisfies its specification by means of software model checking.

Model checking verifies that in every possible state the specification holds. Every full revolution of the wheel triggers an updating process of the motor voltage. This is a multi-step process resulting in a sequence of program states.

During this process the condition can be violated for some processor clock ticks. After the speed and voltage calculations are finished, all the values of the used variables are set properly and fulfill the given requirement, assuming the software is working correct. Therefore the calculation as critical section must be ignored during the model checking process. To mark admissible states we thus define the following temporal logic formula:

$$\text{calculation} := (\text{PC} > \text{start\_calculation} \wedge \text{PC} < \text{end\_calculation})$$

Defining the calculation range

The program counter must be smaller as the beginn of the calculation an bigger than the end of it

Now it is possible to check our software against the specification:

$$\mathbf{AG}\ ((\neg\text{calculation} \wedge \text{time\_period} \leqslant \text{kph25}) \rightarrow \text{motor\_support} = 0)$$

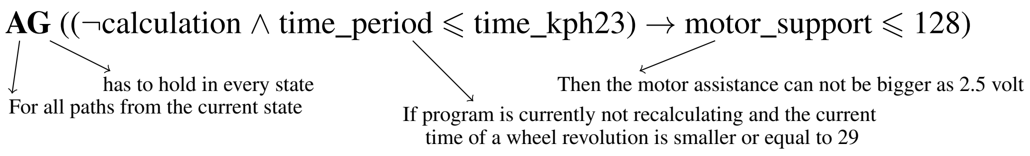
has to hold in every state

For all paths from the current state

If program is currently not recalculating and the current time of a wheel revolution is smaller or equal to 26

Then the motor assistance must be off

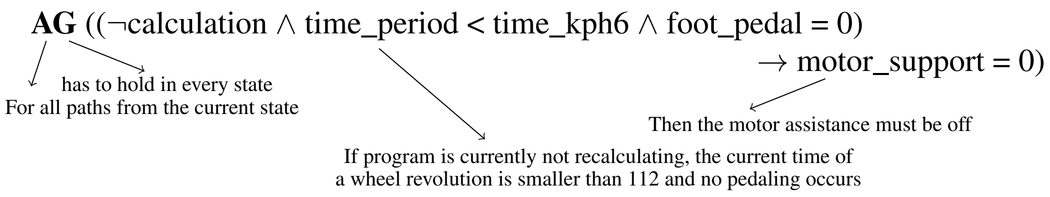
The model checking process proves that the given result is valid. Its not possible that the speed is 25 km/h or higher and at the same time the motor assistance is active. The next step is to verify the progressive reduction of the motor assistance:



$$\mathbf{AG}\ ((\neg \text{calculation} \land \text{time\_period} \leqslant \text{time\_kph23}) \rightarrow \text{motor\_support} \leqslant 128)$$

has to hold in every state
For all paths from the current state
If program is currently not recalculating and the current time of a wheel revolution is smaller or equal to 29
Then the motor assistance can not be bigger as 2.5 volt

Model checking proofs that this formula is indeed valid for our system. The result for time_kph22 $\rightarrow$ motor_support $\leqslant$ 192 and time_kph24 $\rightarrow$ motor_support $\leqslant$ 64 can also be proofed to be valid. The progressive reduction of the motor assistance is thus given and verified. Our software therefore fulfills all requirements of the [6]. The last step will now be to verify the starting aid.



$$\mathbf{AG}\ ((\neg \text{calculation} \land \text{time\_period} < \text{time\_kph6} \land \text{foot\_pedal} = 0) \rightarrow \text{motor\_support} = 0)$$

has to hold in every state
For all paths from the current state
If program is currently not recalculating, the current time of a wheel revolution is smaller than 112 and no pedaling occurs
Then the motor assistance must be off

The given formula is also valid. When the cyclist stops pedalling at the the speed of 6 km/h or faster, the motor assistance is cut off. Our software fulfills all requirements of the specification and is now verified.

Conclusions

The hardware was designed after taking [9, 6] into consideration and is therefore able to protect the operator in a way that is requested by these standards. Our initial software had to be altered in multiple iterative steps to make model checking possible, preventing state explosion. It showed that principles for well designed software did not apply or even were misleading in our case. In the end our software was verified against the specification of [6] under the condition that the hardware works correct.

A few shortcomings should also be mentioned, whose solution is subject to future work:

The value of the Motor-MTTF_d could be too optimistic. If the Motor produces more dangerous failures at average, the value needs to be altered and therefore the System-MTTF_d could be not high enough to ensure the safety of the operator. A solution in such a case would be to increase the PL_r-value to the more appropriate value “e”, which leads to a modification of the needed category and the DC_avg.

Our current approach cannot assure the requirement of [6, 4.2.4.1.a], since it is dependent on the meters covered per crank revolution.

The current method of utilizing hard- and software model checking separately and proving the software specification against perfect hardware without the possibility of hardware errors, is missing some very important facts. If both model checking approaches would be joined together, the new results would deal with problems like reading erroneous values from sensors or having destroyed memory cells in SRAM at critical positions, that are missed out at present.

Therefore a better way of dealing with software model checking in the context of a progressively malfunctioning hardware has to be investigated. In [2] it was proven that the validation of the PL against its PL_r can be achieved through probabilistic model checking. Future work can deal with these results and research how the software behaves under the influence of the hardware delivering incorrect values over time. The software on the other hand could detect those errors and should be able to react with countermeasures.

References

[1] C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.

[2] R. Boeckhorst. *Projektstudie einer Sicherheitsprüfung von eingebetteten Systemen durch probabilistisches Model Checking*. Project Report. Bonn-Rhein-Sieg University of Applied Sciences, 2012–8.

[3] Deutsche Gesetzliche Unfallversicherung DGUV, ed. *BGIA-Report 2/2008 – Funktionale Sicherheit von Maschinensteuerungen – Anwendung der DIN EN ISO 13849*. Druckerei Plump OHG, 2008–12.

[4] Department of Defence – United States of America, ed. *Military Handbook – Reliability prediction of electronic equipment*. 1991–12. URL: `http://snebulos.mit.edu/projects/reference/MIL-STD/`.

[5] EN 14764. *City and trekking bicycles – Safety requirements and test methods*. 2005.

[6] EN 15194. *Cycles – Electrically power assisted cycles – EPAC Bicycles*. 2009.

[7] EN 62061. *Safety of machinery – Functional safety of safety-related electrical, electronic and programmable electronic control systems*. 2005.

[8] EN ISO 12100-1. *Safety of machinery – Basic concepts, general principles for design – Part 1: Basic terminology, methodology*. 2003.

[9] EN ISO 13849-1. *Safety of machinery – Safety-related parts of control systems – Part 1: General principles for design*. 2008.

[10] EN ISO 13849-2. *Safety of machinery – Safety-related parts of control systems – Part 2: Validation*. 2008.

[11] EN ISO 14121-1. *Safety of machinery – Risk assessment – Part 1: Principles*. 2007.

[12] PRESTO, ed. *Cycling Policy Guide – Electric bicycles*. 2010–2. URL: `http://www.eltis.org/docs/tools/presto_cycling_policy_guide_electric_bicycle.pdf`.

[13] B. Schlich. "Model Checking of Software for Microcontrollers". PhD thesis. RWTH Aachen, 2008–6. URL: `http://aib.informatik.rwth-aachen.de/2008/2008-14.pdf`.

[14] Verband der Elektrotechnik Elektronik Informationstechnik e.V. VDE, ed. *VDE fordert mehr Sicherheit für Elektrofahrzeuge*. 2012–7. URL: http://www.vde.com/de/Verband/Pressecenter/Pressemeldungen/Fach-und-Wirtschaftspresse/Documents/2012/12-51_PI_BatZentrum_Standards.pdf.

[15] Zweirad-Industrie-Verband e.V. ZIV, ed. *Statistik Informationen für das Jahr 2011*. 2012–3. URL: http://www.ziv-zweirad.de/public/pk_2012-ziv-praesentation_21-03-2012.pdf.