Functional Safety

12 January 2021 15:59

Book : SIL and Functional Safety in a NUTSHELL by Dr. Michel J.M. Houtermans

© RAJAT KUMAR
https://www.linkedin.com/in/imRajat/
https://github.com/im-Rajat

Functional safety is a property of an active safety function, carried out by a safety system.

Functional safety is the part of the overall safety of a system or piece of equipment that depends on automatic protection operating correctly in response to its inputs or failure in a predictable manner (fail-safe). The automatic protection system should be designed to properly handle likely human errors, hardware failures and operational/environmental stress.

An **active safety function** detects some kind of undesired situation and then takes action (Airbags)

Functional safety does not apply to passive safety functions.

The brakes of your car can fail after many years of operation but when exactly is unknown. We call those types of failures **random hardware failures.**

Hardware can also fail due to environmental events like earthquake, flooding or lightning. When hardware fails in this way we call it a **common cause hardware failure.**

The real trouble maker in industry though is the so called **systematic failure.**

Safety function is one hundred percent functionally safe if all its random, common cause, and systematic failures are one hundred percent under control and therefore cannot lead to malfunction of the safety function.

SIL stands for safety integrity level and is a performance measurement of functional safety and thus of how well the safety function achieves functional safety

Hazard and Risk Analysis :

Identify the problem, analyse the problem (which means establish the risk associated with the problem) and where necessary reduce the problem to an acceptable level of risk

There are many techniques available to support hazard and risk analysis. Well known techniques are Risk Matrix, Risk Graph, FMEA, FTA, HAZID, HAZOP, AHA, OHA, ETA, LOPA, Dispersion Modelling and so on.

If we decide to use additional safety systems then it is very important to clearly define and specify what those safety functions need to do. Now is the time to write a so called **safety requirements specification or SRS.**

It is very important to clearly describe what the safety function is intended to achieve. If the intention of the safety function is not clearly defined, then all the subsequent work related to the design, development, manufacturing (and verification and assessment) of the safety function will be meaningless. You could be doing a lot of work, for nothing.

A good safety function description consists of five elements :

First of all what is going to be measured : flow, the temperature, the height, the pressure, the speed, the vibration etc?

Second, what logic is going to be solved with this measurement. If the temperature goes over 100 ºC then.

Thirdly, what action needs to be taken when the logic actually solves. If the temperature for example goes over 100 ºC then switch off the burner

Fourth, how fast does this function needs to be carried out. For example do all of this within two seconds.

These four elements represent the **functional description of the safety function**.

The fifth and last element therefore describes the safety integrity level. Build this safety function according to the rules of SIL 2

Software :

Software behaves completely differently compared to hardware. Hardware can fail due to a random failure or a common environmental cause.

Software on the other hand, does not fail random, or due to flooding or a lightning strike. If software does not do what we want it to do then we programmed it in the wrong way.

Safe software is software that is able to carry out the safety function even under fault conditions.
It does not matter whether these fault conditions come from hardware or from the software itself.

In the functional safety world there are three **distinct pieces of software:**

embedded software. This software makes the safety device run. It includes the operating system, libraries, diagnostics software, support functions, etc. The embedded software is the software that you get when you buy the safety device and take it out of the box.

On top of the **embedded software sits the application software**. The application software is the software that takes care of the unique safety application. This is the logic and includes, for example, the set limits and voting blocks. This is the software the system integrator Page 20 of 48 usually programs for the end

Functional safety management:-

Activities and responsibilities

Basically functional safety management manages that the right people are doing the right work at the right time with the right tools following the right procedures and guidelines.

IEC 61508 overall lifecycle model:

The lifecycle mode**l identifies all phases the safety function** will go through during its full life. Once the phases are known it is easy to identify the necessary work activities in those phases.

the lifecycle phases help you identify which **documentation or information** is needed to do the work and which documents or output is created when the work is carried out.

When people work they can make mistakes (and that is ok, it can happen). One of the principles of functional safety is to carry out verification. **Verification** needs to be carried out for each lifecycle phase. It needs to be carried out by an independent person, i.e. a person not involved in the work in that particular phase

functional safety assessment :

the functional safety assessor checks that the correct input information has been used to carry out the work, that the same input information has been used to carry out the verification, and that the correct output has been created.

We have now the complete lifecycle under control as we systematically go through each phase of the lifecycle. For each phase, we carry out the work, we carry out the verification and we carry out the functional safety assessment. Once one phase is done, we move on to the next phase and do the same trick.

Validation Phase : to find out whether the specified safety function is actually the one that is delivered. validation is done on the real hardware and software, after it has been installed and commissioned. Because validation is a phase it needs to be verified and assessed

Modification : Until one day somebody wants to make a change. In the functional safety industry not every change is a change. Sometimes we call a change a modification
we often go back to phases that have been completed already and start to make changes in those phases

Planning :

The objective of the **safety plan** is to describe how we think functional safety will be achieved on the particular project.

Since the safety plan describes the safety lifecycle activities, the work, we need to check that the work is done correctly and the correct work was done. In other words we need to do verification and validation, and thus we need to have **a verification plan and validation plan.**

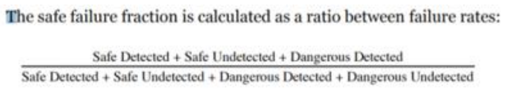
Verify that the work was done correctly, validate that the correct work was done. Per lifecycle phase we describe who is going to perform the verification/validation, how they are going to do it, which tools, procedures, guidelines they need to use or follow, and so on.

The safety plan, verification and/or validation plan are management documents. This is how we manage our work. The safety requirements specification is a technical document

The Hardware (Make it work with or without software):

Now that it is clear what we want to build, and how we are going to do it, it is time to go and deal with the hardware. In the functional safety world there are a lot of hardware concepts that need to be dealt with. Some of these concepts are applicable to the complete safety function, i.e., from sensing element, to logic solver, to final element or actuator, and everything in between. But there are other hardware concepts that apply to subsystems only. A subsystem is for example only the sensors, or only the logic solver.
Lets start with the safety function or loop concepts.

We need to know about safe and dangerous failures and about detected and undetected failures because each device has a so called safe failure fraction.


The safe failure fraction is calculated as a ratio between failure rates:

$$\frac{\text{Safe Detected + Safe Undetected + Dangerous Detected}}{\text{Safe Detected + Safe Undetected + Dangerous Detected + Dangerous Undetected}}$$

Operation, Maintenance and Repair :

Once the safety system is designed, and the correct hardware and software have been selected and built, it is time to install and commission the system.
When the system is commissioned and working as it should, we still need to do one final check before we start using this safety system. We need to validate the safety system.

Operational phase is a very long phase though , so a functional safety audit make sure get when you buy the safety device and take it out of the box.

On top of the **embedded software sits the application software**. The application software is the software that takes care of the unique safety application. This is the logic and includes, for example, the set limits and voting blocks. This is the software the system integrator Page 30 of 48 usually programs for the end-user. This is the software that the safety function is all about. But the application software cannot do any good if the embedded software is not working as it should.

utility software. This is the software in the engineering station or the hand held device. The utility software is the software that we need to actually program the application software. It might sound strange, but this software is not as important as the embedded or application software itself. The reason is very simple. In the field, there is only the embedded and application software and this is the only software that needs to be working. Whether the engineering station is working or not does not matter, as long as the embedded and application software is the software we want it to be.

three distinct pieces of software are programmed in three types of software languages
These three distinct pieces of software are programmed in three types of software languages.

1) The first type is called **Full Variability Language or FVL**. You use FVL when you program C, C++, Pascal, etc. Basically it means that the programming language is so flexible that you can program anything you want.
2) **Limited Variability Languages or LVL.** With this type of language you can program voting blocks, times, alarm messages. You do not have the flexibility as with the FVL languages and thus you cannot make as many mistakes
3) **Fixed Programming Language or FPL.** Basically this means you have a device with software inside but you cannot really program it. You might be able to set some parameters like process related parameters. Maybe you can switch from psi to Barg, or from Celsius to Fahrenheit, or you can set the range of a certain value but that is about it. You cannot program logic with it.

selected and built, it is time to install and commission the system.
When the system is commissioned and working as it should, we still need to do one final check before we start using this safety system. We need to validate the safety system.

Operational phase is a very long phase though , so a functional safety audit make sure that the safety system remains functionally safe during the operation phase.

Sometimes the safety system fails. We repair it and document the reason for failure

CO2 will collect near floor level while CO will collect closer to the ceiling. OSHA lists danger levels for CO beginning at 35 ppm, and for CO2 starting at a 5,000 ppm time-weighted average.

CO identification uses electrochemical sensors. However, CO can also be detected using NDIR sensing.

Different Sensor Links :

MQ-135 Air Quality & Hazardous Gas Sensor Module
MQ-2 Smoke Methane Gas & Liquefied Flammable Gas Sensor Module
MQ-3 Alcohol Ethanol Gas Sensor Module
MQ-4 Methane Gas Sensor Module
MQ-5 Liquefied Gas & Coal Gas Sensor Module
MQ-6 Isobutene Propane Gas Sensor Module
MQ-7 Carbon Monoxide Gas Sensor Module
MQ-8 Hydrogen H2 Gas Sensor Module
MQ-9 Carbon Monoxide Combustible Gas Sensor Module

https://www.flyrobo.in/gas-sensor-kit-includes-9-gas-sensor?tracking=ads&gclid=Cj0KCQiA9P__BRC0ARIsAEZ6irgwyCL5Zsroq4ywKCPvOLPLXbxRm_Bc0vQer3nCGmgyHoynkVjyN4AaAq5TEALw_wcB

MQ9 Carbon Monoxide, Methane and LPG Gas Sensor Module : Rs.119
https://www.electronicscomp.com/mq9-carbon-monoxide-methane-lpg-gas-sensor-module-india?gclid=Cj0KCQiA9P__BRC0ARIsAEZ6irjxG9TSzRuxLGYp79MXaobgcd6aOtYLG3qYCdEjGujOkyid3wyDgLcaArw3EALw_wcB

Carbon Monoxide Sensor - MQ-7 ($4.95)
https://www.sparkfun.com/products/9403

PM2.5 GP2Y1010AU0F Dust Smoke Particle Sensor
https://robu.in/product/pm2-5-gp2y1010au0f-dust-smoke-particle-sensor-w-cable-capacitor-resistor/?gclid=Cj0KCQiA9P__BRC0ARIsAEZ6irjZtqNitMGq0-j0m7mjvDfn7oR7kk1u8dmPhsbxztsrgNgp5MMx80UaAt4JEALw_wcB

https://www.compur.com/en/stationary-gas-detectors/gas-detector-statox-505-sil-61508/

IR Cameras :-

Why we need IR-Cut Sensor on Camera?
While normal cameras will take reddish pictures during the daylight because of lacking IR filter in the CCD, the RPi IR cut c amera can take the much better quality picture at a resolution of up to 1080p during both day and night.

1) Raspberry PI Infrared IR Night Vision Surveillance Camera Module 500W Webcam
   Features :
   i. This camera can also work at night.
   ii. Supported Video Formats: 1080p @ 30fps, 720p @ 60fps and 640x480p 60/90 video
   iii. Fully Compatible and Plug-n-Play camera with Raspberry Pi 3 Model B.
   iv. Small and lightweight camera module.
   v. Resolution : 5 MP
   vi. Sensors : Omnivision 5647 fixed-focus which is in fixed focus mode
   vii. Interface Type : CSI(Camera Serial Interface)

   Link :
   https://www.electronicscomp.com/raspberry-pi-infrared-ir-night-vision-surveillance-camera-module-500w-webcam?gclid=CjwKCAiAo5qABhBdEiwAOtGmbq_oYi2HVUMv5SJcclkFXzJLBzdVsB3u4G4ASV31pBBy_tQXNyBdGhoC8PYQAvD_BwE
   https://robu.in/product/raspberry-pi-infrared-ir-night-vision-surveillance-camera-module-500w-webcam/?gclid=CjwKCAiAo5qABhBdEiwAOtGmbr5_QWEbO03N0XEJsPNvn-J_09UYTjtzxhrM4_HeuhIQnlSjlv6qHhoCQ94QAvD_BwE

2) OV5647 5MP 1080P IR-Cut Camera for Raspberry Pi 3/4 with Automatic Day Night Mode
   Features :
   i. 5-megapixel OV5647 sensor
   ii. camera is capable of 2592 x 1944 pixel static images, and also supports 1080p 30, 720p 60 and 640x480p 60/90 video.
   iii. supports night vision.
   iv. Embedded IR-CUT filter, eliminating colour distortion due to IR light in the daylight
   v. Comes with infrared LED, supports night vision
   vi. Can Attach IR LEDs if Night Vision mode is required
   vii. It is compatible with all revisions of the Pi.

   Link :
   https://robu.in/product/ov5647-5mp-ir-cut-camera-for-raspberry-pi-3-with-automatic-day-night-mode-switching/?gclid=CjwKCAiAo5qABhBdEiwAOtGmbiHB6Zq9lA_63qiQBahaq2juXPIYVhtJfY1HXRKyuF0Q5BUgCIpqHxoCse0QAvD_BwE

3) Raspberry Pi Infrared Camera Module V2, Supports Night Vision- Waveshare

Features:

i. Official Raspberry Pi Camera, supports all revisions of the Pi
ii. Sony IMX219 8-megapixel sensor
iii. 3280 x 2464 still picture resolution
iv. Support 1080p30, 720p60 and 640x480p90 video record

Link :
https://www.evelta.com/raspberry-pi-infrared-camera-module-v2-supports-night-vision-waveshare/?gclid=CjwKCAiAo5qABhBdEiwAOtGmbtu0MkFc_-UAf5i86er9a85-NYc5PLQ4QvTFby7zsJ4RVBih0WA6mxoCy-gQAvD_BwE

Ideas for Enhancement :-

We can add a feature do alarm when drivers eyes are closed for an interval (1 -2 seconds):-
For this, we can use haarcascade_eye_tree_eyeglasses.xml for detecting open eyes on a face.

We can check using the camera for checking if the driver eyes are close for 1 -2 seconds. We it detect eyes are closed for 1 -2 seconds then it will trigger alarm/buzzer.

Eye Blinking :-

The duration of a blink is on average 100–150 milliseconds according to UCL researcher and between 100–400 ms according to the Harvard Database of Useful Biological Numbers.

A resting blink rate is estimated to be between 8 and 21 blinks per minute, but it's believed that blink rate increases when engaged in conversation,
averaging 10.5 to 32.5 blinks per minute (or 19 to 26 blinks per minute according to another estimate)

Eye blink detection :

Algorithm :

-> The frame is captured and converted to grayscale.
-> Bilateral Filtering is applied to remove impurities.
-> Face is detected with the haarcascade.
-> The ROI (Region Of Image) of Face is fed to eye detection part of algorithm.
-> Eyes are detected and resulting list is passed to if-else contruct.
-> If the length of list is more than two, means that the eyes are there.
-> Else the program is marked to be eye blinked and restarted.

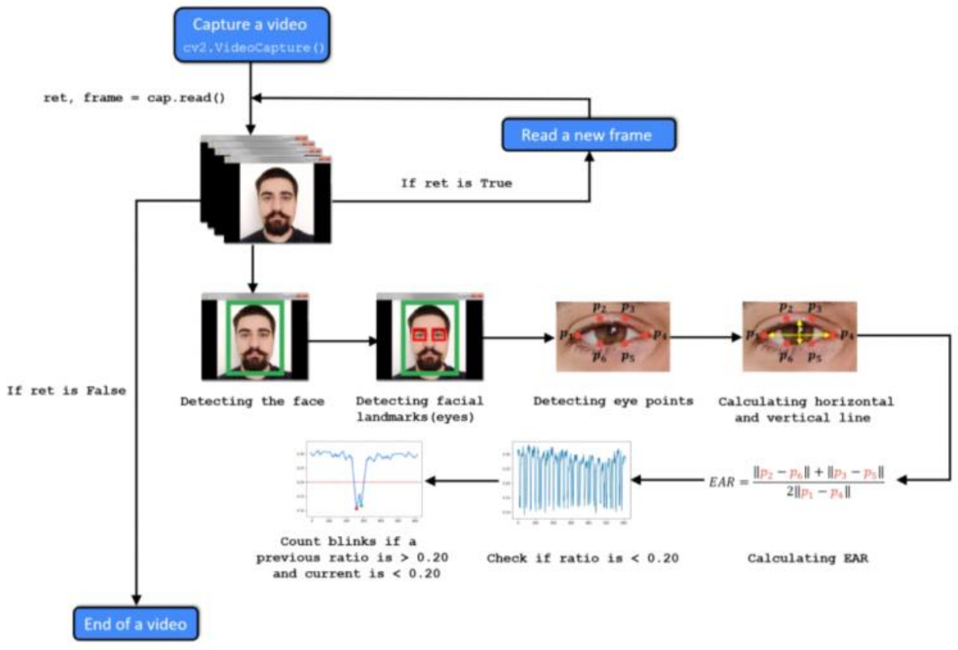
Eye Blink detection using OpenCV, Python, and Dlib (Calculating number of blinks) :

1. Import required libraries :
imutils library : FileVideoStream, VideoStream
dlib : Contains an implementation of facial landmark detection
cv2 : OpenCV library

2. Function to calculate EAR as per the research paper:
EAR = (||P2 -P6|| + ||P3 - P5||) / (2||P1 - P4||)

How to calculate the Eye Aspect Ratio (EAR)?

1) Detecting the face in the image
2) Detecting facial landmarks of interest (the eyes)
3) Calculating eye width and height
4) Calculating eye aspect ratio (EAR) – relation between the width and the height of the eye
5) Displaying the eye blink counter in the output video


Capture a video
cv2.VideoCapture()
ret, frame = cap.read()
Read a new frame
If ret is True
If ret is False
Detecting the face
Detecting facial landmarks (eyes)
Detecting eye points
Calculating horizontal and vertical line
p_1 p_2 p_3 p_4 p_5 p_6
$EAR = \frac{\|p_2 - p_6\| + \|p_3 - p_5\|}{2\|p_1 - p_4\|}$
Calculating EAR
Check if ratio is < 0.20
Count blinks if a previous ratio is > 0.20 and current is < 0.20
End of a video

We used a nested for loop to obtain the more accurate results. When EAR falls below the threshold of 0.20, we assume that a b link has occurred. But what if eyes are closed continuously for a few consecutive frames? In the case that ratio in the following frame is also below 0.20 we will detect two blinks inst ead of one. Due to this reason we created an additional for loop where we will check if the previous ratio is greater than 0.20. So, we will detect a blink if the first ratio of two consecut ive ratios is greater than 0.20 and second is smaller than 0.20. We can visualize this if we plot our ratio signal.

Reference :
https://www.geeksforgeeks.org/python-eye-blink-detection-project/
https://www.educative.io/edpresso/eye-blink-detection-using-opencv-python-and-dlib
http://datahacker.rs/011-how-to-detect-eye-blinking-in-videos-using-dlib-and-opencv-in-python/