WhizniumSBE Code Generation/Iteration Service

Use Case: The ICARUS Detector

Quick facts

- The example presented here shows how WhizniumSBE can be used to help developing the control software for a complex embedded system.
- The system concerned is a multi-camera detector for indoor search and rescue (SAR) operations within the framework of the European FP7 project ICARUS.
- Heavy use is made of WhizniumSBE's modeling capabilities to generate a daemon which delivers images from multiple sources to multiple targets in non-blocking fashion, e.g. for 3D reconstruction, visible/thermal image fusion and laser spot identification at the same time. The master/slave job feature is employed to avoid simultaneous commands from different sources, e.g. from the user interface and from an automated search algorithm. To make the detector a functional part of the ICARUS system of robots and other assets, the project's API is included in a number of external tools.

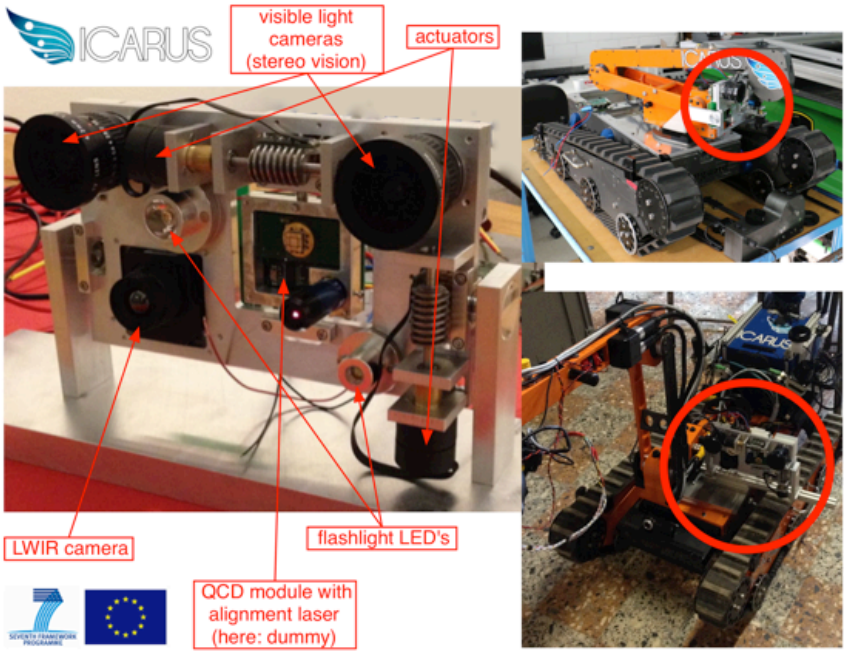

Figure 1: The ICARUS multi-camera detector and its possible mounting positions on the project's small unmanned ground vehicle (SUGV)

Introduction

Control software which smartly combines sensor information while offering convenient access from the outside is required in order to make best use of all ICARUS detector hardware features which are detailed in Figure 1. The main sensing elements are two 1.2 megapixel visible light (VIS) cameras and a long-wave infrared (LWIR) thermal imager with VGA (640x480) resolution. Spot measurements at a specific mid-wave infrared (MWIR) wavelength are supported by means of a quantum cascade detector (QCD) based 8x8 pixel array. Motorized rotation around two axes and a modulated red alignment laser help aligning and locating its narrow field of vision. Auxiliary functions include two high-power LED's and an accelerometer. A gumstix ARM-based mini-computer running Linux is used to access the hardware via a set of custom printed circuit boards (PCB's) holding an FPGA and numerous connectors, e.g. for USB, Ethernet and SPI. All control is handled by the WhizniumSBE-developed software project ICARUSDetectorControl (Idec) and its *combined engine* executable, *Ideccmbd*. *Ideccmbd* is a multi-threaded command-line tool which communicates with web-clients and external software through its built-in application server and API library.

Table 1: The project in numbers

Metric	Value
database tables	36
... of which model	21
... of which query	15
UI modules	2
UI cards	10
source files	
database	79
combined engine	425
web-based UI	663
API	209
binary sizes	
database library	12.7 MB
combined engine	67.9 MB
API library	51.8 MB

Basic features

The detector's subsystems require a number of parameters in permanent storage, for which the XML preferences file is used. Parameters range from paths and connector configurations to geometrical and optical data, to recording preferences. Binary sensor calibration data, e.g. QCD per-pixel gain, is stored in Base64 encoding, along with look-up tables (LUT's), e.g. for the non-linear relation between optical pixel power and target temperature.

Standard UI features, of which a selection is shown in Figure 3, constitute the organizational backbone of *Ideccmbd*. The UI is multi-language to allow SAR specialists to control the detector in their native tongue, while user access rights management makes sure that preferences and calibration features are accessible to maintenance personnel only. Missions, each corresponding to one disaster response, and subordinate tours, each corresponding to one trip of the host SUGV and identified by GPS coordinates, allow to classify recording files. For these, the built-in support of WhiznumSBE for a managed file archive and the netCDF file format are used.

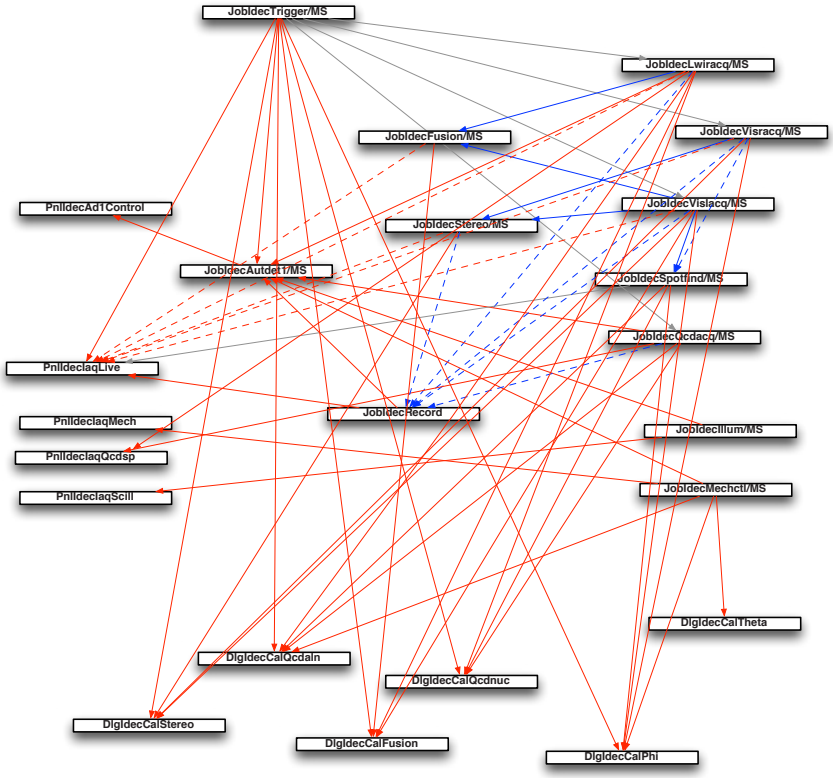

Figure 2: Hardware, processing and UI job dependencies ; preferential master (slave) functionality in red (blue), optional dependencies are dashed

Image processing, e.g. for 3D reconstruction, makes use of the OpenCV library while streaming from the optional PointGrey USB VIS cameras is handled through their proprietary FlyCapture library. WhiznumSBE supports seamless integration with these third-party libraries, including Makefile adaptation.

Hardware jobs

Besides the *jobs* responsible for UI features (named `Crd…`/`Pnl…`/`Dlg…`), the source code tree and run-time *job* hierarchy of *Ideccmbd* contain hardware *jobs*, each responsible for controlling one specific hardware feature ; for a list, see Table 2. The structured modeling approach of WhiznumSBE results in clearly defined dependencies, as indicated in Figure 2 – a *job* pointed to by an arrow receives input from (and `#include`'s) the origin *job*.

Table 2: Hardware and processing jobs

Job	Functionality
JobIdecFusion	fuses the VIS-L image and the LWIR image to form a four color channel image
JobIdecIllum	controls the flood light and spot light LED's
JobIdecLwiracq	acquires and processes images from the LWIR (FLIR) camera
JobIdecMechctl	controls the theta- and phi-axes, reads out the accelerometer / orientation sensor
JobIdecQcdacq	acquires and processes data fom the QCD detector and locates it with the help of the spot finder
JobIdecRecord	collects data from the image acquisition jobs and stores/restores them to/from a netCDF file
JobIdecSpotfind	modulates the alignment laser and tries to identify the laser spot on the images of the VIS cameras
JobIdecStereo	uses images from the VIS cameras to generate a 3D map
JobIdecTrigger	distributes the software trigger signal to all acquisition jobs and to the spot finder
JobIdecVislacq / JobIdecVisracq	acquire and process images from the VIS-L/R cameras

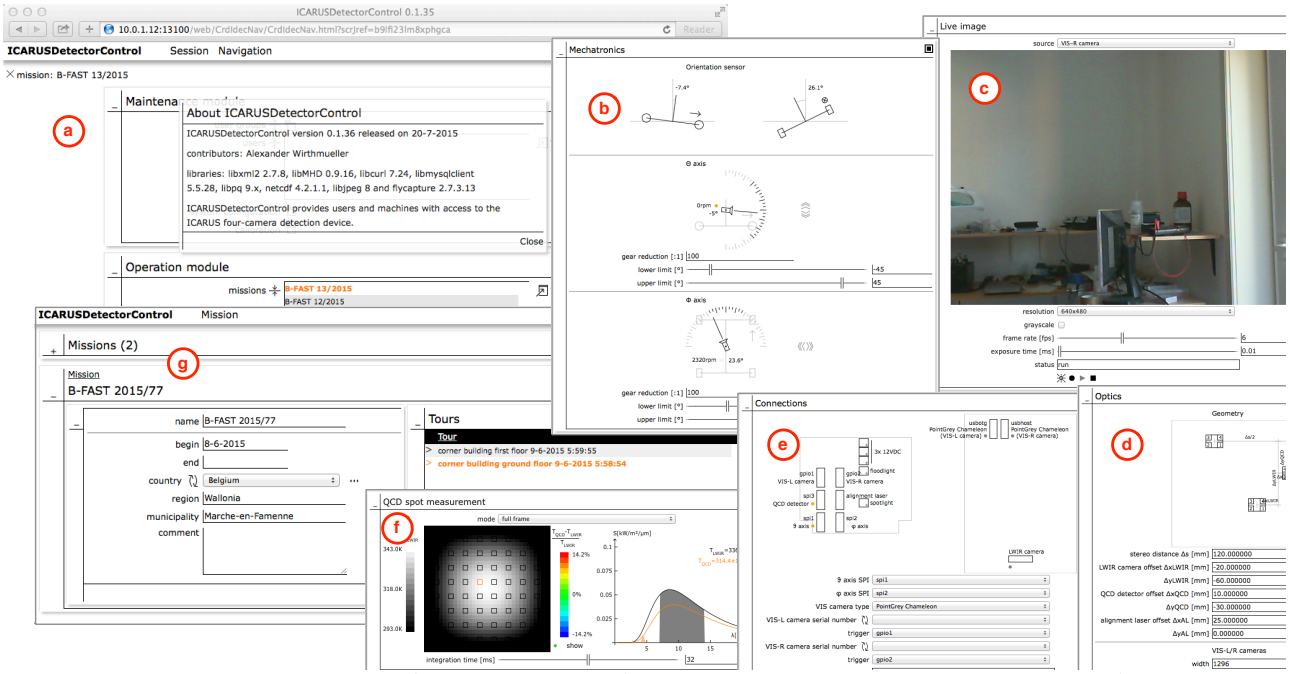

Figure 3: Web-based UI showing a) main navigation, b) mechatronics with interactive SVG controls, c) live camera image as HTML5 canvas, d) optics (geometry) settings, e) PCB connections with color status indicators, f) QCD spot measurement with clickable pixel selector, g) mission record details

The scene illumination *job* `JobIdecIllum` controls the intensities of both floodlight and spotlight LED's. Target angles for the ϑ and φ axes can be passed to the mechatronics control *job* `JobIdecMechctl`. It reads back the displacement progress, for which calibration of the axes' magnetic hall effect sensors is required. The *job* also provides read-out of the accelerometer.

Using the trigger control *job* `JobIdecTrigger`, either a hardware or a software trigger signal can be generated. Both are characterized by global frame rate and pre-/post-trigger settings (with respect to the VIS-L trigger) for the VIS-R, LWIR and QCD acquisition *jobs*. In the run-time *job* hierarchy, each acquisition *job* `#include`'s the trigger *job*. When in software trigger mode, advanced Whiznium**SBE** modeling options ensure that the notification of each acquisition *job* is handled by a call in non-blocking fashion, allowing for real-time synchronization. Firstly, a camera *job* which is busy processing a previous image is skipped. Secondly, acquisition *jobs*, all implemented as state machines, change into the - possibly CPU time-consuming - image acquisition state by using a different thread than the calling thread.

The VIS-L and VIS-R acquisition *jobs* `JobIdecVislacq` and `JobIdecVisracq` are responsible for the read-out of the VIS cameras and subsequent image processing. Standard web-cam's can be accessed via the Linux V4L2 interface and specialized cameras using their proprietary API's. Bayer-to-RGB and YUV-to-RGB conversions at full resolution constitute the first step of image processing.

As shown in Figure 2, the VIS-L acquisition *job* in particular feeds a large number of host *jobs*. Namely, its images can be displayed to multiple users and can be recorded at the same time. They are used for the generation of depth maps and of fused VIS/LWIR images. In addition, they serve to find the alignment laser spot. The Whiznium**SBE** master/slave *job* concept allows for each host to `#include` its own instance of `JobIdecVislacq`, while at any given time, only one of these instances is in master mode and thus responsible for the processing of raw sensor data. Further processing, e.g. resolution reduction or color channel extraction takes place in slave

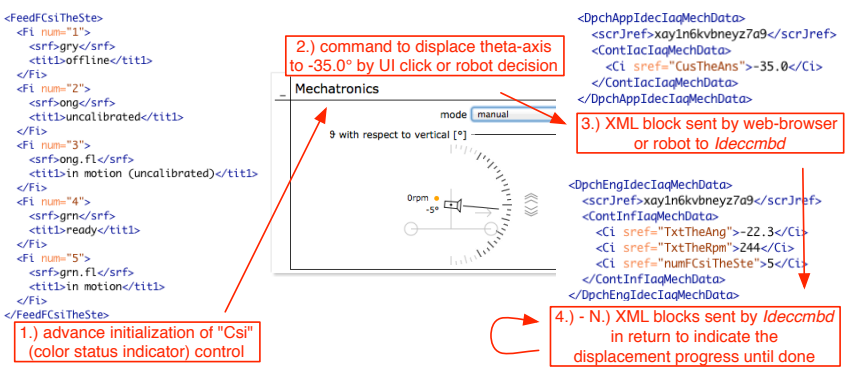

Figure 4: XML communication for a web UI or API call to displace an axis

mode and is customized to the host *job*'s needs. The master/slave-sensitive implementation of the *job*'s state machine makes it irrelevant to the host *job* whether it is receiving data from `JobIdecVislacq` in master or slave mode. When the master *job* is deleted, the master functionality is automatically passed on to the next slave *job* in line, making it the new master.

Images from the LWIR camera are acquired through the *job* `JobIdecLwiracq`. A first linear transform calculates absolute pixel power out of the raw Analog-to-Digital Converter (ADC) pixel reading, followed by a LUT conversion for the the non-linear relation between pixel power and temperature. The LUT's values (16Bit range), calculated based on the physics of filtered blackbody radiation, are stored in the preferences file as a small number of non-equally spaced supporting points. Each time *Ideccmbd* starts up, based on them, the actual LUT is calculated without loss of precision.

The QCD detector *job* `JobIdecQcdacq` is responsible for controlling all features of the QCD detector module, including acquisition of data from the 8x8 pixel detector array and detector cooling. Raw data processing starts with a linear transform from signal to pixel power with pixel-specific coefficients. This is followed by a non-linear transform to obtain the object temperature, governed by a LUT. QCD's exhibit well-known noise characteristics, so that temperature uncertainty can be calculated along.

For alignment of the QCD detector pixels with the LWIR image, the laser spot finder *job* `JobIdecSpotfind` drives the red alignment laser intensity with a sinusoidal pattern which is synchronized with image acquisition from the VIS-L/-R cameras. Image processing algorithms are used to retrieve the laser spot, and to optionally calculate object distance by means of triangulation.

Remote control

With ICARUS being an integration project, it is crucial that other systems, e.g. the SUGV or the robot command, control and intelligence (RC2I) interface, can access the detector in a simple way. This is achieved by incorporating the WhizniumSBE-generated C++ API library *ApiIdec* into external tools. It is provided as a set of methods to write *Idec*-specific XML blocks based on C++ objects on one hand, and to identify and parse XML blocks received from *Ideccmbd* on the other hand. Any action that can be triggered out of the UI, e.g. by clicking, can be initiated by the API as well. Figure 4 showcases the XML block sequence exchanged for the displacement of the ϑ axis.

Using the API, minimal effort is required to design feature-rich native apps with graphical UI, as demonstrated in Figure 5. Here, the functionality of two life image panels, mechatronics control, scene illumination and recordings is combined onto one single screen.

WhiznumSBE, through its accessor app feature, even helps writing the Objective-C code required for non-blocking HTTPS communications, a state machine implementing event-triggered data exchange sequences, and management of all data structures shared with *Ideccmbd* (similar to the DOM in a web-browser).

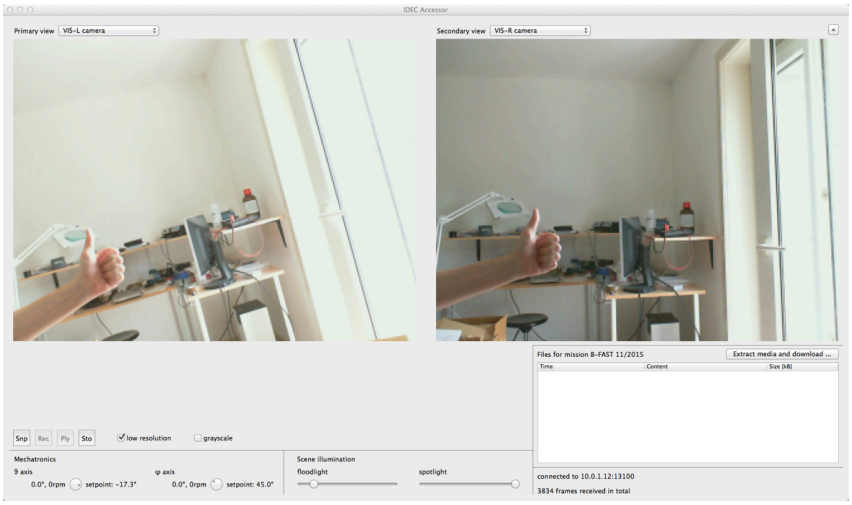

Figure 5: Native MacOS tool accessing Ideccmbd using its API

More information

- WhiznumDBE: The Device Builder's Edition for FPGA-based systems, MPSI Technologies 2018. The tool description is largely based on the *Idhw* WhiznumDBE project for the FPGA-based hardware of the ICARUS detector.

contact@mpsitechnologies.com
+49 175 918 5480
www.mpsitechnologies.com
Helene-Mayer-Ring 4, 04.19
80809 Munich
Germany