How Do You Extend an Embedded System with a Camera Module?

There are many reasons to give your embedded system "eyes" and transform it into an embedded vision system. Image acquisition can turn many devices into 'smart' products or enable entirely new possibilities. This isn't just a trend for systems that should work more autonomously; often it's also an expedient step to make systems more intelligent and integrate artificial intelligence (AI).

In practice, this means integrating one or more camera module(s) into the system to enable image acquisition and analysis. The camera modules required for this are available as components in numerous configurations. But how are they integrated, what are the resulting possibilities and what difficulties might arise, especially considering each user's individual requirements?

Even if you want to buy or order the development of a complete system with an already-integrated camera, or a ready-made solution, there are many things to specify. This includes not only the purely optical features but also elements such as software interfaces or the desired performance of the image processing, which in turn has a direct effect on the choice of hardware.

In this White Paper, we give developers a short overview of relevant aspects as well as recommendations on the factors that should be considered to achieve a good solution that is also lean and cost-effective.

Contents

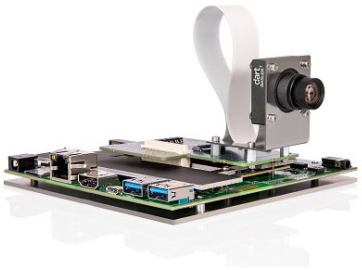

Figure 1: Embedded vision system consisting of Snapdragon 820 processing board and camera module with BCON for MIPI interface

1. What is a camera module?

1.1 More than an image sensor

The core component is the image sensor, the piece of silicone that transforms the light incidence into electronic signals and ultimately the image information. In a camera module, this image sensor is already permanently integrated and connected. In addition, a camera module contains the electronics required for the sensor operation, a housing / mount and a lens or lens mount. Other components not immediately apparent are the interface, driver software and the image signal processing (ISP), which is performed by the hardware (HW) and/or software (SW). The last two items are described in more detail below:

1.2 Interface and drivers

For the image signals to be utilized by the embedded system, they have to arrive in the appropriate form in this system's processor. This requires a HW connection, i.e. a plug with an attached cable. This connection is standardized in USB connections but not in other interfaces. An LVDS data transmission or MIPI CSI-2 connection is customary in the embedded range. For more information, please see our White Paper "The MIPI CSI-2 Interface for Embedded Vision Applications". The interfaces differ in terms of their bandwidth and maximum cable length, among other things. Transferring the data to a suitable SW interface also requires a driver. You can find more details on this in the "Software Connection" section.

1.3 Image Signal Processing (ISP)

It's often easy to forget that a camera also provides image data (pre-)processing. In color cameras, for example, this includes the color value calculation from the transferred green, red and blue values (the so-called de-bayering) for each pixel. Interpolations and corrective calculations (e.g. for noise reduction or the elimination of fringe effects) are also part of this. In order to do this, either the camera board provides a corresponding logic element or the embedded system has a specifically configured processor (which is then referred to as the image signal processor) and must be supported by the appropriate driver software.

Simple camera modules don't contain this preprocessing, so it must be performed (less efficiently) in the GPU or CPU of the embedded system. This might engage required resources, so these components may need a higher-performance configuration.

2. Which features are required for the camera module?

2.1 Resolution and pixel size

The resolution refers to the number of pixels in the image. How many pixels you need greatly depends on the application and can be calculated through the geometry. For example, to still-capture a face (20 cm wide) in an image that covers an area of ten meters with 40 pixels (horizontal), you need a horizontal width of 2000 pixels. For a square image, this would be 4,000,000 pixels, or 4 MP.

However, it's misleading to think that the more pixels you have, the better. More pixels aren't always an advantage and don't necessarily result in the ability to capture more details. There are two reasons for this: For one, each pixel must be processed. Thus, it's always better to have fewer pixels to get a quick, efficient and energy-saving image computation. The other reason is based on a physical explanation that has to do with natural image noise. This is familiar from images that you take in low light, for example candlelight. The phenomenon is created by random statistical fluctuations, the amount of incoming light and by electronic effects ("dark noise"). The resulting fluctuation is particularly obvious in low light:

If there is not much light, e.g. 50 light particles, you could expect a fluctuation of ±10 light particles. This amounts to 20 %, which is a relatively high fluctuation. As a result, a consistently light surface in the image no longer looks all that uniform. If there is a lot of light (e.g. 20,000 light particles, the fluctuation (e.g. ±145 ≈ 0.7 %) is low and the image is "clean".

But a high amount of image noise reduces the effective image content even if you have a lot of pixels. This is shown by a practical example: In a comparison test, the recognition rate of faces was worse with an 8 MP resolution than with a 2 MP camera. Even if the second image had fewer pixels per face, these had less noise (a better signal-to-noise ratio) and were thus more "informative". In any case, the recognition algorithm was able to handle it better.

Ideally, if possible, one should therefore ensure that the lighting conditions are good, e.g. with external light sources, so that the image noise is minimized. Additionally, the pixels on the image sensor shouldn't be too small, so that as many light particles as possible land on one pixel and keep the noise low. However, this means that fewer pixels fit onto the sensor surface, and therefore the sensor has a lower number of pixels.

The ideal resolution of a sensor is always a compromise between resolution and signal quality or informative content. Find more information in our White Paper "How do you assess image quality?"

Figure 2: Image with low (left) and high (right) image noise

2.2 Frame rate and sensor type

The frame rate or image refresh rate are decisive for timely resolution. They refer to how many images can be captured per second and can be vary from camera module to camera module. Here again, it's important to calculate which value is required for the specific application. Usually 30 frames/second are enough for a stream. Otherwise you should consider how quickly the captured subjects (or camera) are moving.

But this brings up another point which every developer should consider: Should the camera have a global or rolling shutter sensor? A global shutter records all image lines simultaneously and a rolling shutter does this sequentially. If the object moves during the image capture, the image shows the so-called rolling shutter effect.

If the objects are moving, a sensor with a global shutter therefore makes more sense. Such a sensor is also generally more expensive than a rolling shutter sensor. But it isn't always needed. In an image from a rolling shutter sensor used for facial recognition, the face can appear slightly skewed if the person moves during the recording. However, this distortion is so slight that a facial recognition algorithm usually achieves equally high recognition rates as when a sensor with a global shutter is used. Here, a rolling shutter would be enough. Furthermore, in modern sensors, this effect isn't as pronounced anymore, i.e. the distortions are often very slight. For that reason, a

Figure 3: The rolling shutter effect. If the recorded objects are moving, strange motion artifacts might appear from sensors with rolling shutters.

global shutter is usually only required for very fast-moving objects. Developers with limited experience should therefore test which sensor type is expedient for their application. You can find more information about the topic of global and rolling shutters in our White Paper "Global Shutter, Rolling Shutter – Functionality and Characteristics of Two Exposures Techniques."

3. How can the camera be integrated into the embedded system?

3.1 The hardware connection

Aside from USB, there are no standardizations on the hardware side – neither for LVDS- nor MIPI-based connections. A USB connection is easier, but has the disadvantage that it generally isn't the cheapest option and also doesn't use the leanest solution when it comes to the system setup (including the data transfer).

However, other connections are always specific to manufacturers. In other words: Either the camera or SOM manufacturer provides the suitable adapters, or a customized hardware adjustment is needed. This can happen through an adapter solution (cable or board) or by adapting the carrier or processing board.

3.2 The software connection

The software is similar to the hardware: only USB offers generic drivers in the form of USB3Vision. For other interfaces, the camera manufacturers usually offer their own, proprietary drivers.
The drivers then connect to manufacturer-specific SDKs that provide software interfaces (APIs) for the most common programming languages. These APIs function for the image data transfer from the camera but also – and this is often overlooked – to transfer control signals to the camera so that it can be operated and configured.

With an LVDS- or MIPI-based interface, the drivers can also be connected to the Video4Linux interface. This in turn offers an interface to programming languages and various other kinds of software. Video4Linux is a commonly used software, but isn't always optimal for many specific applications, since its possibilities are limited (for example in controlling the camera).

3.3 Software libraries

The image data can be accessed directly in the program code. Then another image analysis (e.g. facial recognition or edge detection) can be performed, through resources such as finished software libraries and tools. These products are available as open source (e.g. OpenCV) as well as commercial software. The latter is often designed for specific applications or offers special features.

4. What else should be considered?

4.1 Don’t forget the lens

A camera module needs a lens in order to function. For more information, please see our White Paper “Which lenses are available and how do you choose the right lens for a camera?”

Many modules have a permanently installed lens while others offer a lens mount. Two aspects should be kept in mind. First – larger lenses let more light in – an advantage for the image quality (see box above). And second: The accuracy of the lenses also permits only a specific maximum resolution. A cheaper lens might have a resolution of 5 MP, where more expensive ones might offer 20 MP. So if you want to use a camera module with a high pixel count, you need a corresponding (potentially expensive) lens. This is another reason why high resolutions aren’t necessarily desirable for sensors per se.

4.2 Triggering

If you don’t film continuously but selectively take individual images, you must send a trigger signal to the camera at the right time. This can happen through the software (via “software trigger”). But not every interface permits this; for example, it isn’t possible to send a trigger from the program code via the Video4Linux interface. Other SDKs, by contrast, offer this option. In addition to the software trigger, some camera modules also support the option to initiate the image capture directly with an external trigger signal (“hardware trigger”) through a control input at the camera. Such a signal can come from inputs such as a light barrier or a switch.

5. Summary

If you develop or specify an embedded system with a camera module, you should take several aspects into account to get a clearer idea of the solution and prevent stagnating in the middle of the development.

Of course, there are experts who can give more precise answers and suggest the right solutions in a direct discussion. Important aspects to consider are: how easy a camera module is to integrate, how it can be used in the application, which features it actually offers and of course how good the image quality and thus the image’s informative content really is.

As an expert for vision solutions, Basler offers not just the individual components such as camera module and software but also the entire range of services. In addition to a consultation about suitable components and support with their integration into embedded systems, Basler also helps with the adjustment or creation of system software all the way to the development of complete solutions.

Author

Dr. Thomas Rademacher
Product Manager

Dr. Thomas Rademacher has been a Product Manager at Basler since 2015. He is responsible for the new Basler dart cameras with LVDS-based BCON interfaces for embedded vision systems.

After earning his PhD in Physics at the University of Göttingen, he worked as a product manager for a leading industrial measurement technology firm in the semiconductor industry, with a focus on optical measurement technology and automated image processing and analysis. His work there centered on optical metrology and automated image processing and analysis.

Contact

Dr. Thomas Rademacher – Product Market Manager - Factory & Traffic – Product Manager

Tel. +49 4102 463 487
Fax +49 4102 463 46487
E-Mail: thomas.rademacher@baslerweb.com

Basler AG
An der Strusbek 60-62
22926 Ahrensburg
Deutschland

About Basler

Basler is a leading manufacturer of high-quality cameras and camera accessories for industry, medicine, traffic and a variety of other markets. The company's product portfolio encompasses area scan and line scan cameras in compact housing dimensions, camera modules in board level variants for embedded vision solutions, and 3D cameras. The catalog is rounded off by our user-friendly pylon SDK plus a broad spectrum of accessories, including several developed specially for Basler and optimally harmonized for our cameras. Basler has three decades of experience in computer vision. The company is home to approximately 600 employees, at its headquarters in Ahrensburg, Germany, and at its subsidiaries and sales offices in Europe, Asia, and North America.

Basler AG
Germany, Headquarters
Tel. +49 4102 463 500
Fax +49 4102 463 599
sales.europe@baslerweb.com
www.baslerweb.com

Basler, Inc.
USA
Tel. +1 610 280 0171
Fax +1 610 280 7608
sales.usa@baslerweb.com

Basler Asia Pte Ltd.
Singapore
Tel. +65 6367 1355
Fax +65 6367 1255
sales.asia@baslerweb.com

BASLER
the power of sight