University of Applied Sciences Pforzheim

Engineering Department

Project Work

Further development of an open-source thermal imaging system in terms of hardware, software and performance optimizations

Name	Maximilian Ritter
Student number	313709
Advisor	Prof. Dr. Karlheinz Blankenbach
Submission date	13.01.2017

Table of Contents

1 Introduction

Originally invented for the military sector in the 20th century, thermal imaging systems are an important part of today's consumer and industrial measurement technology [1]. From the aspect of hardware components, a thermal imager is similar to a digital camera. But instead of a CCD or CMOS sensor, it does contain a sensor element able of detecting and measuring the long-wave infrared radiation, which is also called heat radiation. This kind of radiation is not visible for the human eye and therefore, a special device is required to perceive it. An infrared camera uses specific colour schemes to convert the measured infrared radiation to a so called thermal image. By using this method, cold temperatures can be for example mapped to the color blue and hot temperatures to the colour red, as it is the case for the so called "Rainbow" colour scale.

Although thermographic cameras have become less expensive in the last few years due to the increased amount of produced units as well as new sensor technologies like uncooled micro bolometers, they are still quite costly for a majority of the people. For this large target group, namely private persons, small companies and educational institutes or schools, a low-cost alternative was invented. The DIY-Thermocam, which stands for do-it-yourself thermographic camera, has the purpose to offer an open-source alternative to the commercial solutions on the market. The design of the device is modular, and it is offered as a self-assembly kit containing of over one-hundred components. The first version of the device (V1) was developed during a bachelor thesis at the university of applied science Ravensburg-Weingarten in late 2014. In the meantime, the device has been built over two-hundred times. The customers provided value feedback concerning improvements on the hardware and software side. This lead to the idea of inventing a new version of the device, with improved hardware and software capabilities.

In this project work, the version 2 (V2) of the DIY-Thermocam is developed. It is based on the design from 2014, but provides many enhancements in terms of software, hardware and performance. An additional video output module was invented, to provide the possibility to create a video output signal out of the data stream from the Thermocam. The software development involves the firmware for the thermal imaging system itself, the firmware for the video output module, as well as computer software like the thermal live viewer or the adaptation of the thermal analysis software. The hardware upgrade includes a higher resolution IR array sensor, a new microprocessor, a better and faster visual camera and an exchangeable internal storage with more space for images & videos. This development was done in the form of several iterative prototypes and a small volume production in the end, delivered to customers worldwide.

2 State of the art

Figure 1 - FLIR E4

As already mentioned in the introduction, thermal imagers have become more and more affordable in the last decade, resulting from a price decline in the used infrared sensor technology. However, they are still not in the reach for everybody. The raw resolution of the thermal sensor has the biggest influence on the price. A low-end model like the FLIR E4 from Figure 1 with a thermal resolution of 80 x 60 pixel costs around 1000€. In the mid-range, a resolution of 160 x 120 pixel is common, which is what the DIY-Thermocam V2 offers. Comparable devices like the FLIR E6 cost around 2000€, what is far away from a cheap solution. In the upper range, resolutions of 320 x 240 or 640 x 480 are common, ranging from 5000€ to 30000€. In Figure 2, several technical features from the FLIR E6 are compared to the DIY-Thermocam V2.

Feature	DIY-Thermocam V2	FLIR E6
Price	499€ (Kit)	2000€
Thermal resolution	160 x 120	160 x 120
Thermal sensitivity *	< 0.05° C (50 mK)	< 0.06° C (60 mK)
Thermal temp. range	-40° C to 200° C	-20° C to 250° C
Field-of-view (FOV)	56 deg HFOV, 71 deg diagonal	45 deg × 34 deg
Display	3.2" 320x240 , touch input	3.0" 320x240, no touch input
Spot sensor temp. range	-70° C to 380° C	-20° C to 250° C
Spot sensor temp. accuracy *	0.5° C over wide range	±2 ° C or ±2%
Temp. measurement mode	every position, multiple positions	spot (center) mode
Image modes	IR image, visual image, combined	IR image, visual image. MSX
Color schemes	18 different color schemes	rainbow, iron, grayscale
Storage mode	picture and video	picture only
Storage capacity	8 GB internal storage	500 sets of images
File format	standard BMP and raw data	standard JPEG and raw data
Operation time	about 4-6 hours	about 4 hours
Weight	255g	575g

Figure 2 - Comparison table DIY-Thermocam vs. FLIR E6

3 Physical principles

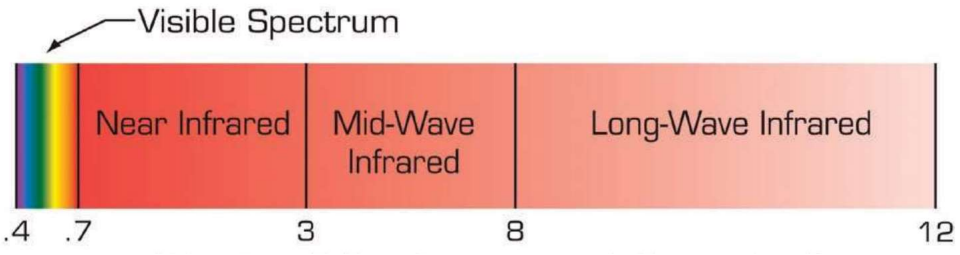


Figure 3 - Electromagnetic spectrum

With our eyes, we can only see the electromagnetic radiation in the visible range of the spectrum. The part right to this visible spectrum is hidden for the human eye, as the maximum perceivable wavelength is at around 1000nm or 1.0µm. That ranges into near infrared spectrum, but is quote different to the thermal infrared from the long-wave IR spectrum. Figure 3 shows the electromagnetic spectrum on the right side of the visible spectrum. On the left side, the also invisible UV, X-ray and gamma radiation is located, which is not relevant for the viewing point of this report.

Every object with a temperature above the absolute zero point of 0 Kelvin respectively -273.15 degree Celsius emits an electromagnetic radiation proportional to its own temperature. A part of this radiation is called infrared radiation and can be used the estimate the temperature of the object [2]. According to DIN5031, the infrared range divides into the following subranges: Short-wave IR-A radiation with wavelengths between 780nm and 1.4µm, IR-B radiation with wavelengths between 1.4µm and 3µm as well as the long-wave IR-C range with wavelengths between 3µm and 1mm [3].

Around 1800, the astronaut William Herschel discovered the existence of infrared radiation, when he analysed the temperature difference of chromatic light. To do so, he used a prism out of glass, through which he directed the sunlight to create a spectrum. Afterwards, he measured the temperature of the colours and determined, that they rise up from the violet part towards the red part of the spectrum. To approve his thesis, Herschel decided to additionally measure the temperature behind the red part of the spectrum, where no sunlight was visible. By doing this he discovered, that the area contains the highest temperature at all, what proved his initial thesis about the existence of an invisible heat radiation inside the spectrum created by the prism [4].

Humans cannot detect the infrared radiation directly over their eyes, but they can experience the heat caused by it, for example when sitting in front of a fire or sitting in the sun. The nerve

endings of the human hand are able to differentiate between temperature differences of only 0.009°. [5] This way of indirect measurement is of course only possible for objects, which are not too hot or cold for the human skin and are directly accessible. Another problem is, that humans can only perceive temperature difference relatively to the ambient temperature. So in order to measure absolute temperatures over a bigger distance, we are dependent on the help from thermos-sensitive electrical measurement devices, such as infrared cameras.

At modern infrared thermography, the radiation is captured with such devices. This works by focusing the emitted IR energy from an object or scene by an optic (typically germanium or silicon) towards the detector element(s). The detector typically consists out of semiconductor sensors, which change their resistance based on the incidental infrared radiation. [6] The advantages of such a non-contact temperature measurement are obvious: The measurement is non-destructive, resulting in no interference with the measured object or area and therefore also no wear. In addition, the response times are very fast, because the electromagnetic waves move along with the velocity of light. The so measures intensity of the radiation does cause a change in voltage or resistance, which can be evaluated by a measurement electronic. This generates a signal, that can be transformed in a thermal image, also called thermogram, and displayed on a screen. The different colours correspond to the distribution of the infrared radiation on the measured object or surface [7]. All modern thermal imagers work based on this principle.

It would be very handy, if different objects emit the same infrared radiation, when their temperature is also the same. Unfortunately, this is not the case, because the intensity distribution varies with different factors. The radiation captured by a thermal imager contains of emission, reflexion and transmission out of the long-wave IR spectrum, as Figure 4 illustrates. The degree of emission e is a measure for the ability of a material to send out infrared radiation. The reflection r is bigger on polished, flat surfaces than on mate, rough ones. And finally, the degree of transmission t is a measure for the ability of a material to feed-through IR radiation [8].

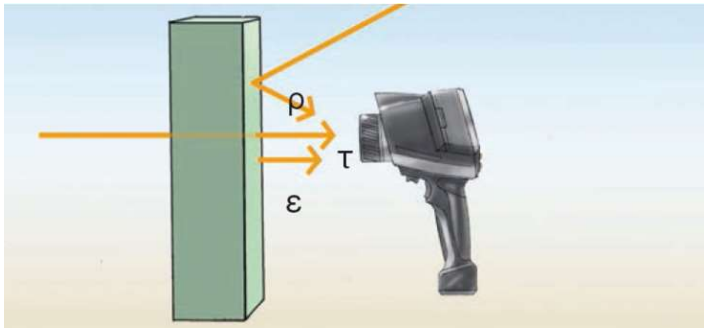


Figure 4 - Emission, reflexion and transmission

4 Realisation

The development is based on the DIY-Thermocam V1 from 2014, which was developed during a bachelor thesis in the field of applied information technology at the university of applied sciences Ravensburg-Weingarten. This chapter should give an overview about the advancements in the different fields of engineering, resulting in the release of the DIY-Thermocam V2.

4.1 Hardware improvements

4.1.1 Microprocessor

Figure 5 - Teensy 3.6 microcontroller

One of the biggest improvements in terms of hardware is the microprocessor. The first version of the DIY-Thermocam used a Freescale ARM Cortex-M4 processor from 2012 (MK20DX256VLH7). The CPU had a frequency of 72 MHz and 256KB flash memory, which were pushed to its limits by the graphical user interface of the firmware. Additionally, a RAM of only 64KB is a real challenge for rendering the thermal and visual image. CPU-intensive tasks such as a decompressing a JPEG byte stream in real-time or filtering a bitmap require a lot of computation power, therefore a newer and better solution had to be found.

In October 2016, a successor of the Teensy 3.2 was released, called the Teensy 3.6. Figure 5 shows the microcontroller from the top side. The big advantages of the Teensy microcontroller are their compact form factor, their easy-to-solder 2.54mm pin holes and the software compatibility with many Arduino libraries. It is also a lot simpler to port the existing firmware code from the Teensy 3.2 to this new version, than to rewrite everything for the SDK of another processor or company.

The Teensy 3.6 features a new and much more powerful Cortex M4F processor from NXP (MK66FX1M0VMD18). Some key features of the NXP K66 series are illustrated in the block diagram of Figure 6. The biggest improvement is the faster CPU with a core frequency of 180 MHz, that can be overclocked up to 240 MHz. Other big enhancements are 1MB of flash space and 256KB of RAM. This bigger internal memory allows full-screen rendering with a resolution of 320x240, compared to only 160x120 on the Teensy 3.2 The increased amount of I/O is nice, but was not needed for the upgrade, as the pin layout should stay the same where possible.

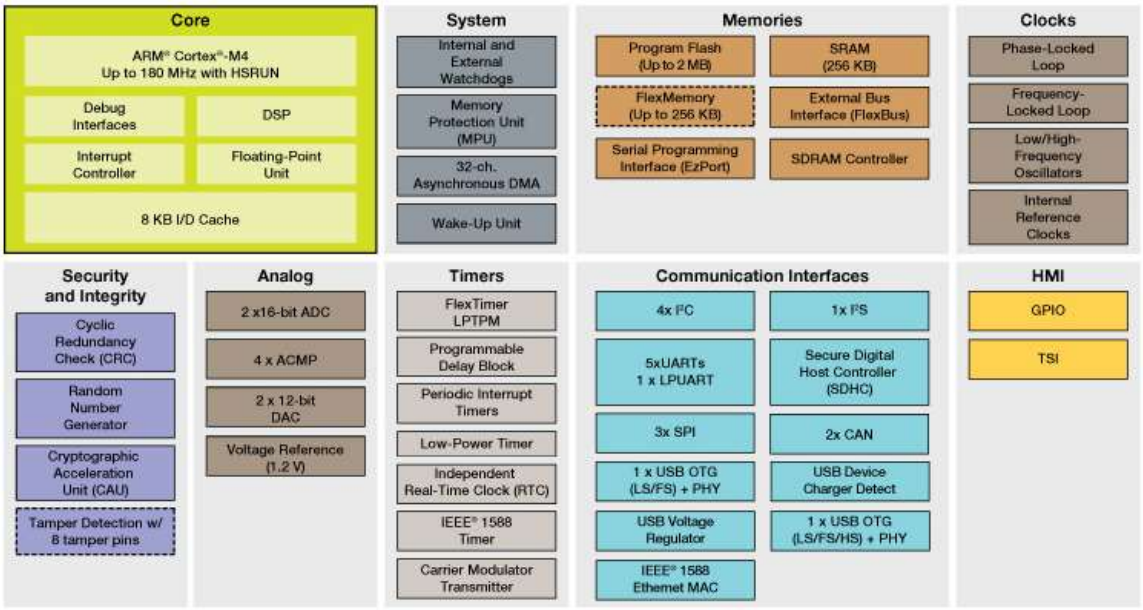


Figure 6 - Kinetis K66 block design

4.1.2 Visual camera

Figure 7 – Arducam V2 Mini camera module

The main issue when using a visual camera with the Teensy microcontroller is, that it does not have a digital camera interface (DCMI). Therefore, a standard interface has to be utilized to transfer the large amount of image data from the camera sensor to the MCU. Unfortunately, most standard camera modules rely on the slow UART protocol. Even at the maximum baud rate of 115200 Kbit/s, the transfer is very slow. The DIY-Thermocam V1 used a VC0706 UART VGA camera with a maximum resolution of 640x480. Because of the long transfer times resulting in a slow refresh rate and a noisy image quality in dark ambient light, a better solution had to be found.

Compared to UART, SPI offers a much faster transmission, that's why a camera using it was searched. There are not many solutions available on the market, and some from China have a very bad image quality. After testing several modules, finally a good solution was found. The result of the search is the Arducam V2 Mini camera module shown in Figure 7. It uses an OV2640 image sensor from OmniVision, with a maximum resolution of 2MP (1600x1200) and has a very compact form factor with no big lens attached. The field-of-view of the lens is with 60 degrees similar to the FOV of the LWIR sensor (56 degrees), which is important for creating combined images.

The image sensor can be configured over the I2C protocol. This allows for example to set the image type (RAW or JPEG) and the resolution (160x120 to 640x480). The image sensor communicates with a FPGA on the backside of the module, which has an internal frame buffer to store the image data. That JPEG byte stream can then be transferred over the SPI interface to the Teensy microcontroller with a clock frequency of up to 8 MHz. Figure 8 shows a block diagram of the Arducam system architecture.

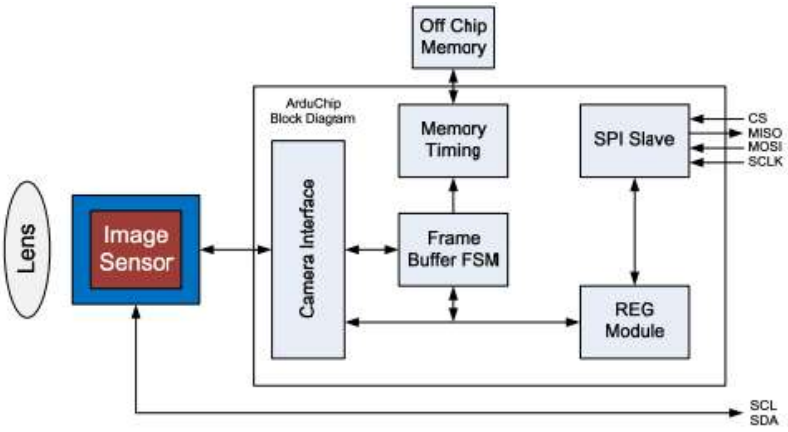


Figure 8- Arducam block diagram

4.1.3 Long-wave infrared sensor

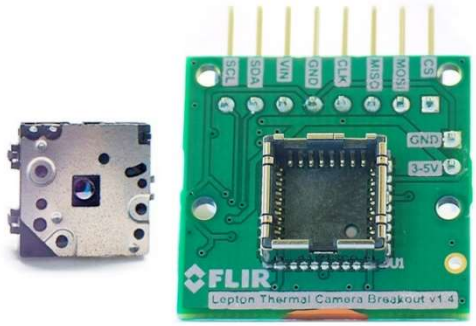


Figure 9 – FLIR Lepton3 module and sensor

The DIY-Thermocam V2 uses an upgraded version of the FLIR Lepton sensor. This third generation hardware has a thermal raw resolution of 160x120 pixels, compared to only 80x60 on the previous, second generation of the long wave infrared sensor. The Lepton3 is not available on the market as a standalone sensor so far, so it has to be unassembled from purchased FLIR One Smartphone thermal imaging camera attachments. The removal is easy and can be done in under five minutes, the sensor has a shutter already attached. Figure 9 shows the FLIR Lepton sensor left to the breakout board, which is used to communicate to the sensor and keep it in a safe place.

Similar to the visual camera module, the configuration of the Lepton is done over I2C, whereas the data is transferred over the SPI interface with a clock speed of up to 20 MHz. The shutter attached to the lens opens and closes automatically, when required. It provides a uniform surface, that can be used to recalibrate the active pixel array and remove noise from the image. It can be triggered manually by the user or automatically after a specific time period. The lens focuses the radiation towards the focal-plane array (FPA) and a system-on-chip (SOC) solution from provides the measured IR radiation of the single thermal pixels as 14-bit intensity value on the output. Figure 10 gives an overview about the FLIR Lepton3 system architecture. The FLIR Lepton2 and Lepton3 sensor have the same pinout and form factor, so only the software part is different, which needs to catch four times the amount of pixels in four different segments. That allows to upgrade previous versions of the DIY-Thermocam with the new sensor, too.

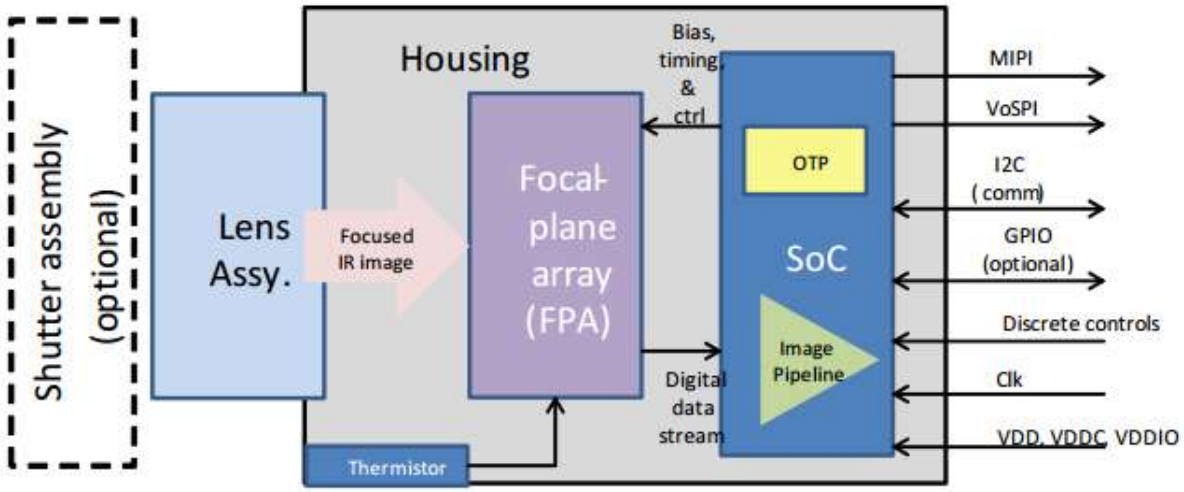


Figure 10 - FLIR Lepton3 system architecture

4.1.4 Storage and printed circuit board

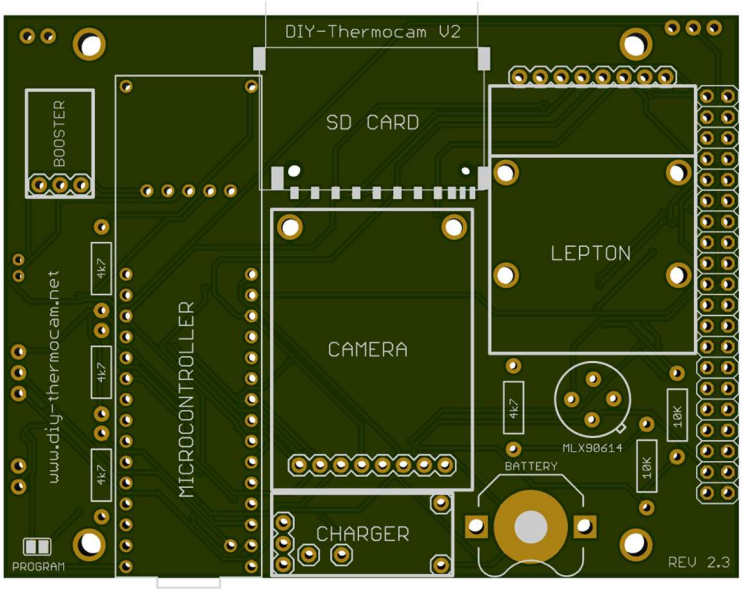


Figure 11 - Printed circuit board revision 2.3

A suggested improvement from many customers was to integrate an exchangeable internal storage into the device, as the DIY-Thermocam V1 only provided 2GB of space. At version 1, that microSD card was installed inside the device and was only accessible over a mass storage device, when connected to a computer. For the second version, a SD slot was placed on top of the printed circuit board, which allows to insert and remove a microSD adapter with up to 8GB of storage over a small slot in the enclosure. The SD card stores raw images / frames, bitmaps or JPEG images from the visual camera. Raw files can be converted to bitmaps later on-device or on the PC, and can also be loaded on the device or inside the thermal analysis software.

Figure 11 show the printed circuit board (PCB) designed for the DIY-Thermocam V2. The schematic is attached to this report in appendix A.3. The modular design of the device allows an easy assembly of the whole Thermocam in about 2-3 hours. The only required components are a soldering iron, some solder, a knife and a gripper. The subcomponents are the following: Microcontroller, visual camera, FLIR Lepton, spot sensor (MLX90614), display (connected over 40-pin header), 5V booster, lithium battery charger, coin cell battery, SD card and several resistors used as pull-ups for the I2C bus and voltage dividers for the lithium battery gauge.

During the work on the V2, three revisions of the PCB had to be created, ordered and tested, to come to this final result. The circuit was created with Cadsoft Eagle and the production of the circuit boards are done in China, because of the low production costs compared to Germany.

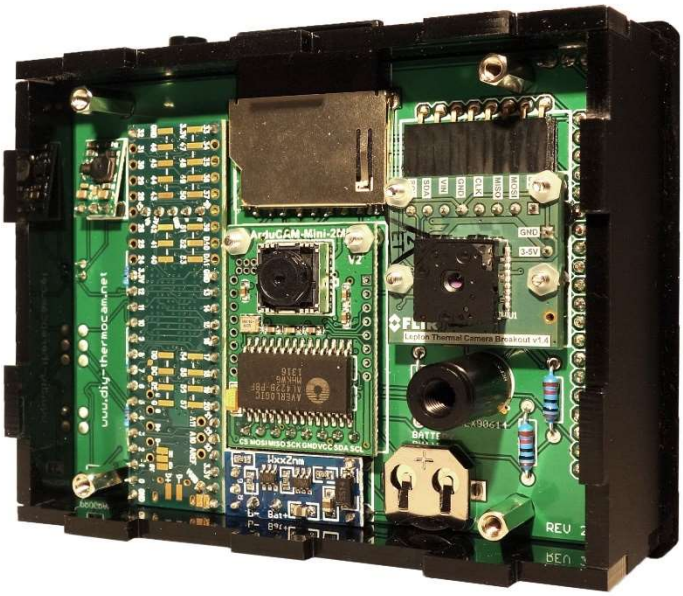


Figure 12 – Inside view device

Figure 12 gives an inside view into the DIY-Thermocam V2 with all the different components and modules soldered into the PCB and the enclosure assembled. It is important, that the LWIR sensor and the spot sensor are located close to each other, because otherwise an accurate calibration would not be possible. Furthermore, the visual camera has to be installed with a slight inclination towards the LWIR sensor, in order to improve the parallax issue. Of course, this does not work for every distance to the object, but the alignment of the visual and thermal image can be adjusted over the device firmware or on the computer at a later point.

4.1.5 Enclosure

Figure 13 - Top and back view device

Creating an enclosure for a small production device is not the simplest thing to do. Injection moulding is expensive, as the mould has to be created in a costly procedure first. 3D printing

has made many progresses in the last years, but is still slow and requires a lot of post processing. For this special kind of do-it-yourself device, it is also important to make the assembly as easy as possible, resulting in a stable construction that does not need to be glued.

The solution was to design a laser cut enclosure out of acryl plastic, shown in Figure 13. It consists out of six panels with 3mm width each, that are fixed by screws and nuts. The production is done by a German company called “Formulor”. One complete enclosure does cost about 10€. A finishing process is not required at any time; the production quality is very good. The outside of the material has a mate surface, so it cannot get scratched too fast.

4.1.6 Video output module

Figure 14 - Video output module

A microcontroller like the Teensy does not offer any video output capabilities like for example a Raspberry Pi does. Many customers want to attach the Thermocam on a drone and transmit the video signal to their ground station or show the thermal live images on a big screen, so a solution had to be found.

The result is the video output module from Figure 14. A Raspberry Pi Zero is used to create a HDMI or analogue video signal out of the USB serial connection to the DIY-Thermocam. For more information about the serial protocol, check out appendix section A.1. A python script running on top of the Linux operating system receives the data from the camera and creates a thermal image stream with a resolution of 640x480 pixels. Unlike the thermal live viewer covered in the software part, the thermal image is not rendered on the Pi Zero itself. Instead, the display buffer from the Thermocam is transferred as whole, what increases the refresh rate.

4.2 Software improvements

4.2.1 Device firmware

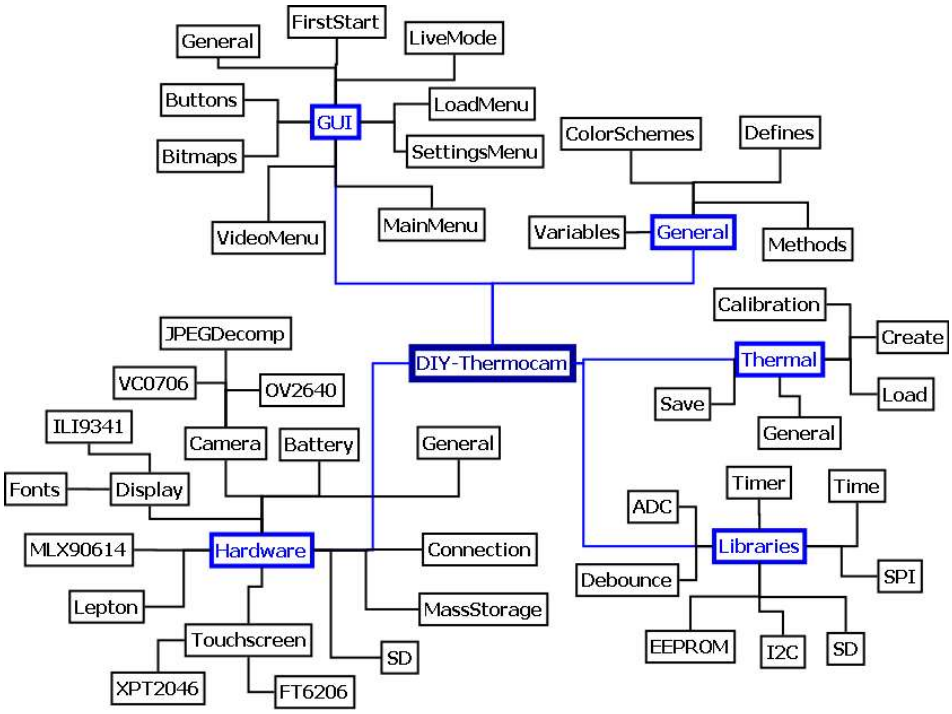


Figure 15 – Class diagram of the device firmware

The firmware of the DIY-Thermocam is written in the programming language C/C++ in an Arduino code compatible programming style. That allows developers familiar with the Arduino IDE to understand the code easily and make their own adjustments. The whole firmware is posted on the GitHub and is open-source under the GNU General Public License v3.0 license. Visual Studio 2015 is used as IDE, together with an add-on called “Visual Micro” and the Arduino / Teensyduino software. Figure 15 gives an overview about the structure of the firmware, the used hardware and libraries.

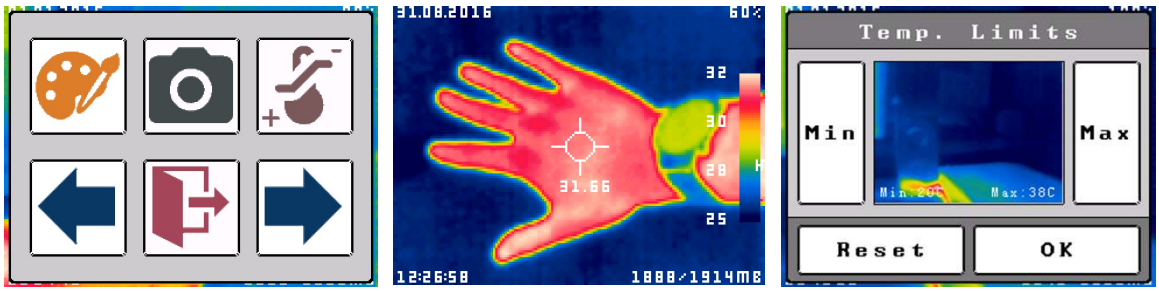


Figure 16 – Device firmware screenshots 1

The existing code base of the DIY-Thermocam V1 was extended, to support the new hardware of the V2. Pre-processor defines detect, for which hardware generation the code was compiled. That decision can be made with just one click inside the Visual Studio IDE. Backwards compatibility to the first version of the Thermocam was an important aspect when designing the adaptations, so previous customers can also profit from new software releases and improvements inside the firmware.

Figure 16 and Figure 17 show screenshots of the graphical user interface (GUI) of the device firmware. The interface is displayed on a 3.2 inch color display with a resolution of 320x240 pixel. The user input is done over a resistive touch. A capacitive touch panel was also tested, but not used in the final V2 because of issues with pressure towards the touch IC installed on bottom of the display module.

The various firmware functions are described in appendix A.4. After the device is turned on, it boots into the live mode. There are three different modes possible: Thermal, visual or combined. Combined is a transparent fusion of the thermal and visual image, similar to what FLIR uses with their proprietary MSX technology (MSX applies edge detection). A push button on top of the device allows it to capture images or videos. When the user touches the screen, the main menu is opened. The main menu also contains a menu to load the stored images and videos.

An additional spot sensor, the MLX90614 single-point IR sensor from Melexis is used to convert the raw values from the FLIR Lepton3 to absolute temperatures. The relationship between raw values and the temperatures can be linearized around a specific operating point. The user can trigger a calibration over the main menu and then point the device to different hot and cold objects in the surrounding area. One-hundred raw values from the Lepton and temperatures from the spot sensor are sampled. A least-square fit algorithm is applied to create a calibration formula. The result is a calibration slope and offset stored in the EEPROM for future usage. The offset is compensated constantly with the spot sensor and the ambient temperature.

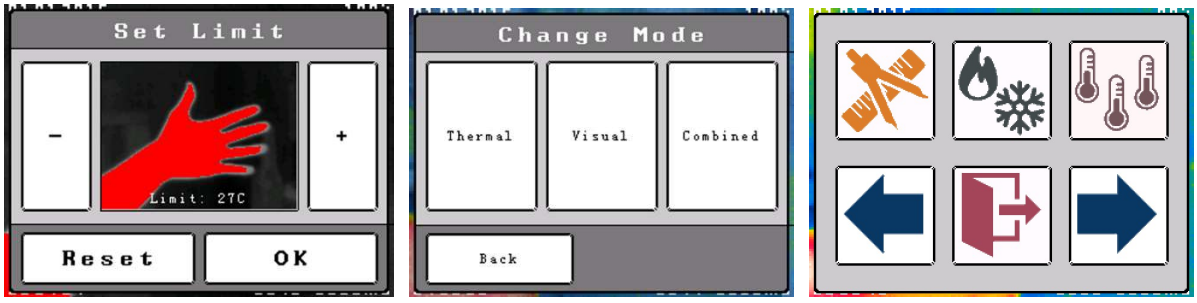


Figure 17 – Device firmware screenshots 2

4.2.2 Thermal live viewer, video output module and firmware updater

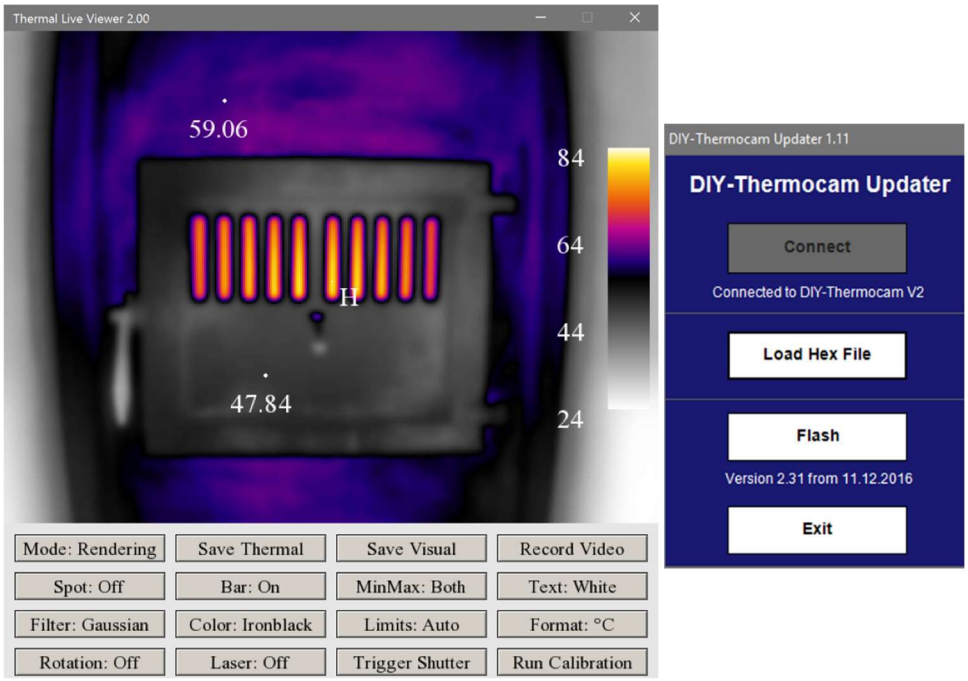


Figure 18 - Thermal live viewer and firmware updater

The thermal live viewer implements a connection over the USB serial protocol to the Thermocam in Python and therefore can be executed on any common operating system with Python installed. For Windows, there is standalone executable available that has been compiled with an application called “Py2Exe”. A simple code base for the thermal live viewer already existed under the V1, but it did not offer any options to control the device or adjust settings.

Version 2 of the application offers many new possibilities to influence the creation of the thermal image, the additional information and hardware features on the device itself. Figure 18 shows a screenshot of the software in action, capturing a furnace. The program implements two different transmission options: rendering and streaming. The second one is what the video output actually uses, so just the display frame buffer is transmitted and displayed. The rendering mode (default) creates higher quality thermal images by rendering them out of the raw data from the device itself. As the CPU of a modern PC offers a much better performance than a microcontroller does, the resolution and quality of the thermal image can be increased this way.

It is possible to save thermal and visual images to a bitmap file right inside the application, either by clicking on the button or using the push button on the device together with a longer

USB transmission cable. The software can also record .AVI videos and add or remove temperature points with a mouse click inside the image. If someone wants to build an integrated solution of the DIY-Thermocam without the usage of a display, all functionalities can be controlled by the thermal live viewer alone. The device detects the connected hardware modules on start-up and automatically switches to the USB transmission mode, if no display is attached.

New firmware updates can be flashed to the device easily with the standalone firmware updater, programmed in C# under Visual Studio 2015. The updater was extended with detection capabilities from the USB serial protocol, like determining the hardware model, the battery status and the current installed firmware version. The hex file from the download section of the website can be loaded and flashed with one click, if all requirements are fulfilled. In case the flash procedure fails for unknown reasons, the device can always be set into program mode with a simple jumper on the PCB.

4.2.3 Thermal data viewer and thermal analysis software

Figure 19 - Thermal data viewer

The thermal data viewer displayed in Figure 19 is a computer software, that allows it to view and edit the raw data files, created by the DIY-Thermocam. For more information about the structure of the raw files, check out appendix section A.2. The software was created in cooperation with Laszlo Lovass, a customer from the DIY-Thermocam V1. It offers many adjustments to the thermal image, like applying different filters, palettes, adding/removing measurement

points, changing the limits, etc. The pictures can be exported to various image formats, and it's also possible to convert a whole series of raw data files, for example from an image or time-lapse capture to an export format.

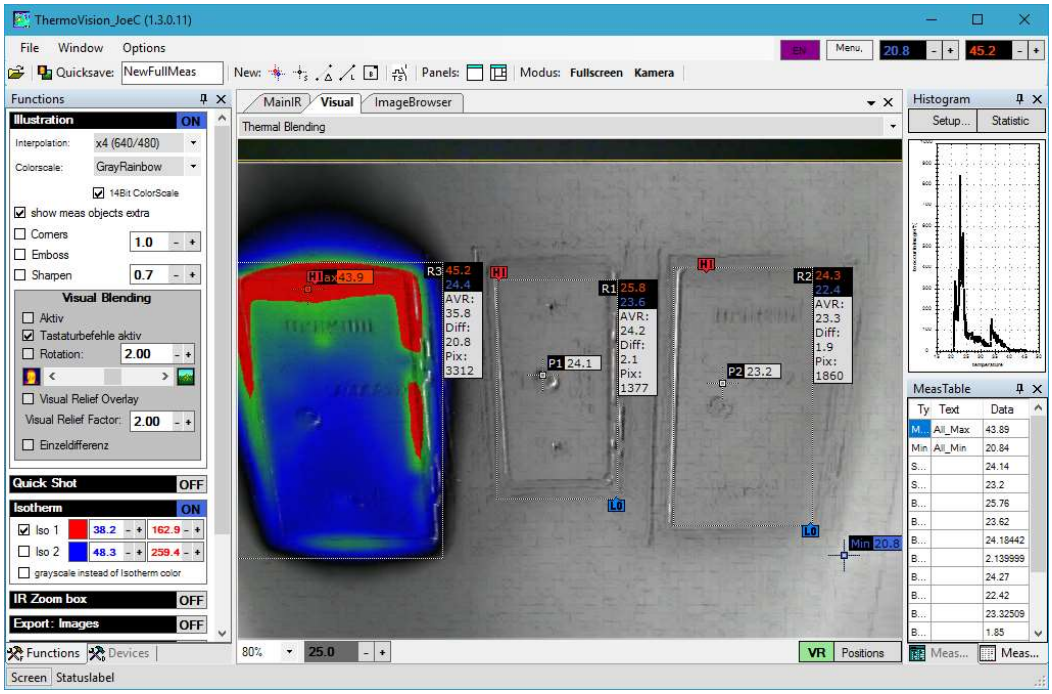


Figure 20 - Thermal analysis software

The thermal analysis software is the preferred solution for editing thermal raw images on the computer, as it features a rich set of functionalities, which are normally only available in proprietary solutions from big companies like FLIR or FLUKE.

This open-source software is created by Johann Henkel from Berlin, and has been adopted to support the DIY-Thermocam. The raw data files can be imported and edited or analysed. The software implements the serial protocol, so a live connection to the Thermocam is possible, too. It also supports adding visual images and offers the possibility to create combined images.

4.3 Performance optimizations

The Teensy 3.6 was overclocked to a maximum core frequency of 240 MHz, which proved to be stable, also under constant load and high ambient temperatures. An updated version of the GCC toolchain in version 5.4 (previous was 4.8) is used together with compiler optimizations (O3 and link time optimization) to speed up the existing code. Whenever possible, the code is tweaked to profit from the increased amount of available internal memory.

On the new visual camera module, the SPI Burst Transfer is implemented to transfer the JPEG byte stream as a whole from the frame buffer of the FPGA to the RAM of the Teensy. Otherwise, the image data would have to be transmitted in packages, whereby each package has to be requested by a special command over I2C. The JPEG data is then handed over to a software JPEG decompressor, that converts it to a RGB565 raw image. This image can be displayed directly on the screen, or combined with a thermal image. In visual only live mode, around 7 FPS can be achieved, in combined mode around 5 FPS.

```
byte gaussianKernel[3][3] = {
    { 1, 2, 1 },
    { 2, 4, 2 },
    { 1, 2, 1 }
};
long sum;

for (int y = 1; y < 119; y++) {
    for (int x = 1; x < 159; x++) {
        sum = 0;
        for (int k = -1; k <= 1; k++) {
            for (int j = -1; j <= 1; j++) {
                sum += gaussianKernel[j + 1][k + 1] * smallBuffer[(y - j) * 160 + (x - k)];
            }
        }
        smallBuffer[(y * 160) + x] = (unsigned short)(sum / 16.0);
    }
}
```

Figure 21 - Gaussian filter

The DIY-Thermocam V1 used a rather complex algorithm to filter the thermal image. As the resolution has increased by a factor of four times to 320x240 pixel, this filtering method would take too long. To speed it up, the simplest possible filter algorithm was implemented, using 3x3 kernels. This provides good enough filtering on the small screen and can be done with one iteration only. Figure 21 shows the Gaussian filter algorithm. There is also a box filter available, with all kernel values set to one.

When using the standard display library for the ILI9341 SPI display controller, all additional information like text, symbols, numbers, etc. are written in a separate write cycle. For displaying many additional info on one screen, this does not only increase the refresh time, it also leads to unwanted pop up effects. In order to avoid this problem, all additional display data is written into the framebuffer first. The display library rewritten to a new, custom implementation. After all information are embedded into the framebuffer containing the thermal, visual or combined image, the whole buffer is send to the display and displayed in only one refresh cycle.

4.4 Field of applications

4.4.1 Building analysis

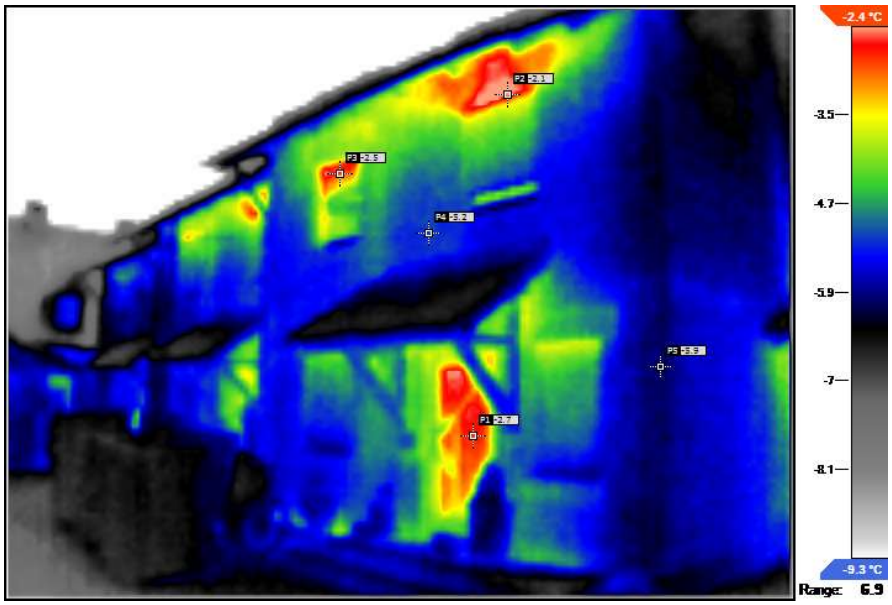


Figure 22 - Building from outside in winter

Especially in winter, a lot of energy can be wasted by bad insulation, resulting in high fuel bills. Figure 22 shows a house front in winter. It can be observed, that the isolation of the door and the upper two windows are not good. This information can help to improve the energy efficiency by either switching the door / windows or by adding an extra insulation to the edges.

Figure 23 - Air leak in the attic

Another scenario is the detection of air leaks. Those are air entrances or exits through the frontage, that are caused by different pressure levels. For this application, the coldest color scheme of the DIY-Thermocam can be very helpful, or the settings of a blue isotherm above a grayscale thermal or combined image, as shown in Figure 23. A third possible field of application is the discovery of mould behind walls or blankets. The heat signature of moisture differs from a standard wall and therefore it can also be detected by a thermal imager.

4.4.2 Electrical and mechanical applications

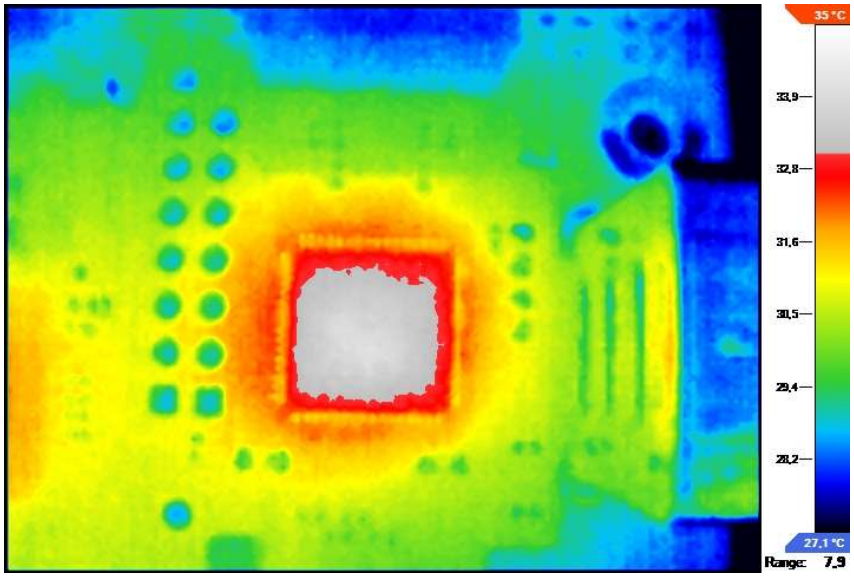


Figure 24 - Thermal surveillance of a CPU

Thermal imaging systems are often used for the inspection of electrical facilities or devices, because they can detect defects or potential disks contactless in almost no time. For example, the DIY-Thermocam could be used as a monitoring device for the CPU temperature of a prototype. The video mode or interval capture is ideal to monitor that situation over a long period of time. Alternatively, the thermal analysis software could be used in live mode together with a programmable temperature threshold action, like triggering an alarm or opening a specific program, when a temperature threshold has been crossed. Figure 24 shows a thermal image of such a thermal surveillance of a CPU.

Another possible application together with electrical devices is the detection of hot spots. They can lead to defects or wrong constructed electric components and help engineering to evaluate their circuit in the real world during the testing phase.

Mechanical components can also be analysed by the DIY-Thermocam. Figure 25 shows the thermal image of a car engine after a ride on the speedway. In this case, it needs to considered, that the maximum distinguishable temperature of the Lepton3 is at about 250 degrees and some of the parts may be even hotter. A radiometric version of the Lepton, the so called Lepton3.5 will be released later in 2017, which allows to measure higher temperatures by settings the sensor's measurement range to low gain.

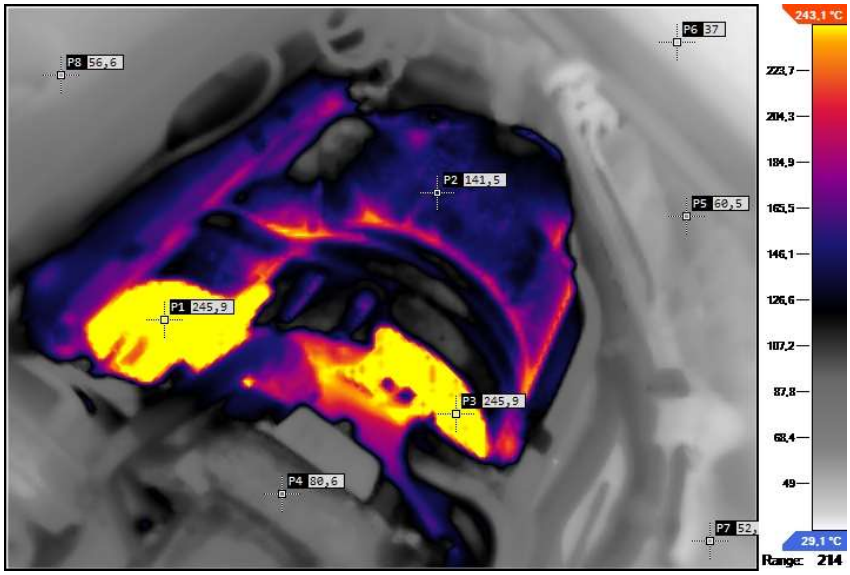


Figure 25 - Car engine

4.4.3 Rescue, firefighter and police services

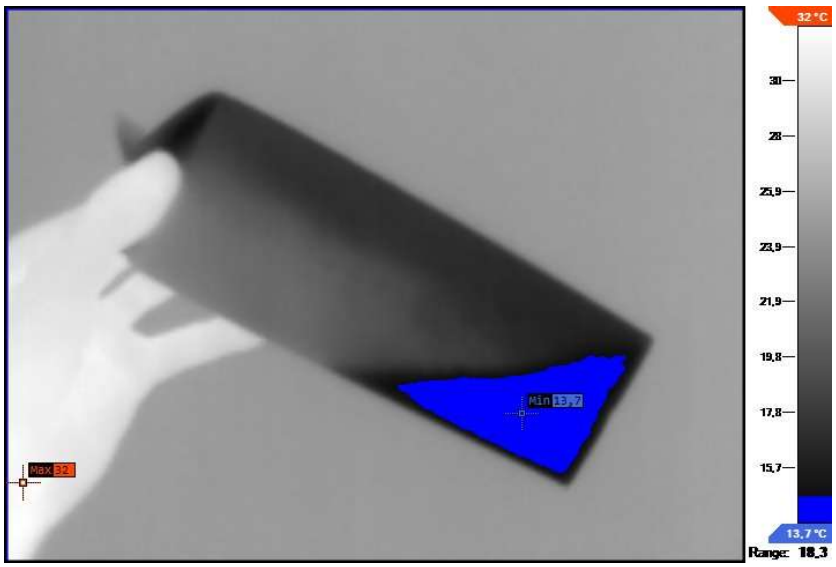


Figure 26 – Filling level control

Another possible area of application is the operation at rescue, firefighter and police services. Especially authorities in small cities do often not have the budget to invest in costly devices. Rescue teams and the police could use the DIY-Thermocam to detect persons in unclear territory like forests or mountains. The color scheme hottest or the isotherm feature with a settable temperature threshold at the body temperature would be ideal to spot those persons.

For firefighters, the device cannot be used inside burning buildings filled with smoke, because the enclosure cannot resist the temperatures in such an environment. But it is possible to spot fire sources from the outside or with some distance to the fire. Another imaginable usage is the filling level control dangerous goods without opening them. Figure 26 shows this principle on the basis of milk inside a bag.

4.4.4 Medicine

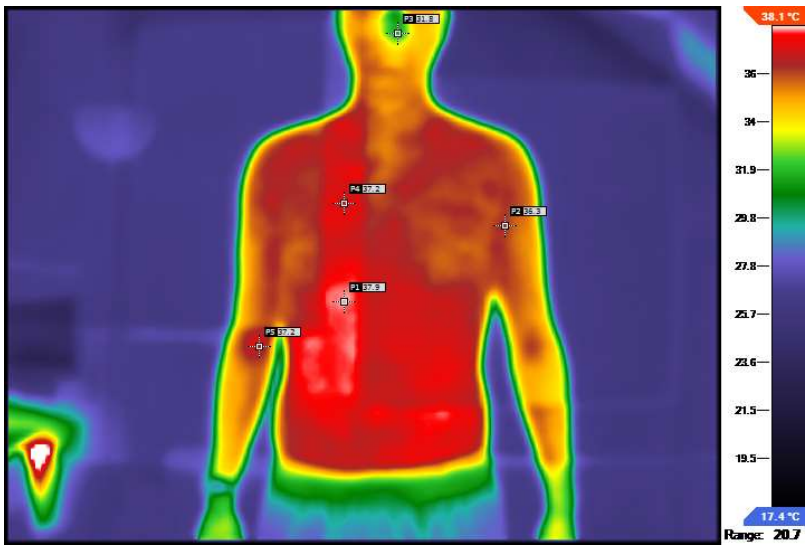


Figure 27 - Human with naked upper body

In medicine, thermography is commonly used to detect temperature differences on the skin surface, that can be an indication for specific kind of diseases. For example, the device could be used to detect breast cancer at women in an early stage. Figure 27 shows the thermogram of a human with a naked upper body in the medical color scheme.

Another field of application is the detection of fever in between a big crowd of people. When the swine flu spread to Europe in 2009, thermal imagers were utilized at airports or train stations to identify people with fever arriving from other countries. The isotherm mode can be set to the fever temperature of human skin, so only affected persons will be marked in red.

5 Conclusion

Figure 28 - Final device

In a time period of only four months, the DIY-Thermocam received a major hardware upgrade to version 2, included many new software features and improvements. Figure 28 shows the final device attached to a small tripod.

To evaluate the result on a public basis, a small production series of twenty pre-assembled & tested devices is delivered to interested people worldwide. The feedback of those customers will be valuable to enhance the hardware furthermore and to find remaining bugs in the device and computer software. All software and hardware sources are published on Github[1], together with a complete part list with possible distributors.

Because of its open-source concept, the DIY-Thermocam has become popular for research institutes worldwide. The firmware, the serial protocol, the raw data files and the python implementation provides a great starting point for many research topics. Companies delving on the topic of thermal imaging also value this solution. For example, the DIY-Thermocam was the basis for a surveillance company developing a system to detect housebreaker based on their heat signature.

[1] https://github.com/maxritter/DIY-Thermocam

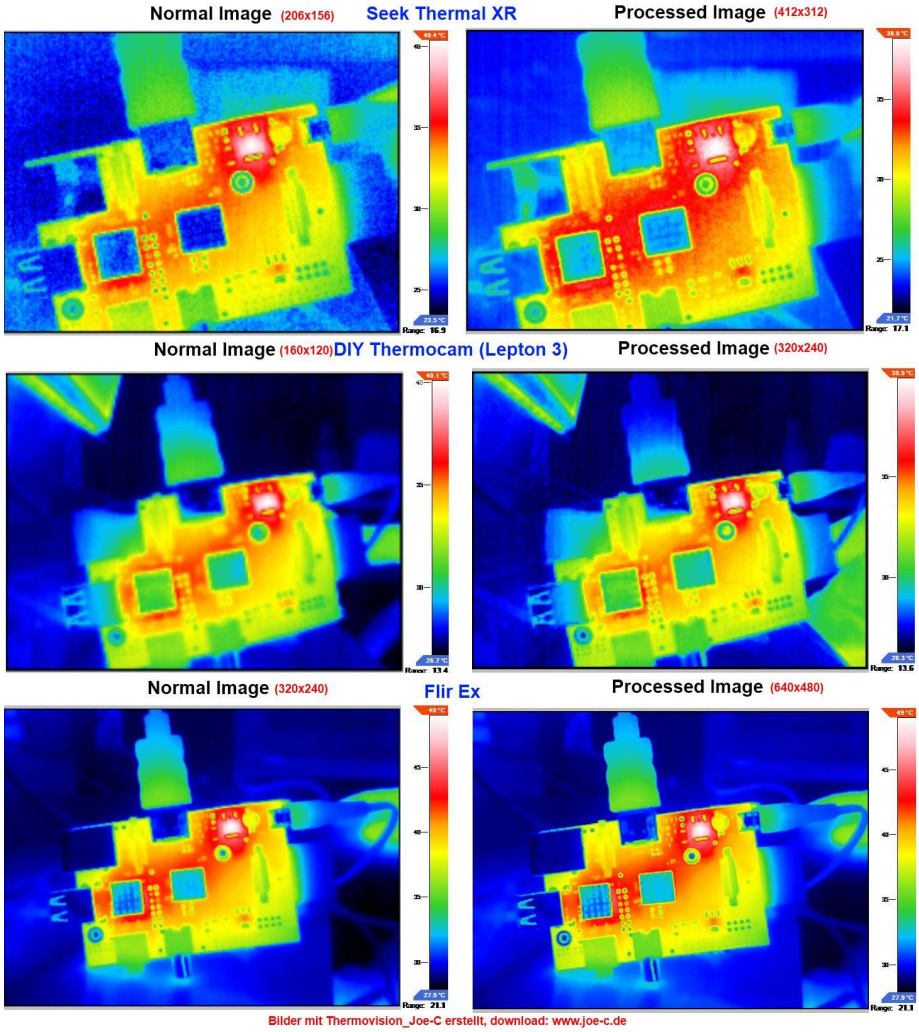


Figure 29 - Thermal image comparison

The image quality of DIY-Thermocam V2 should be compared to two other commercial devices, the Seek Thermal XR (300€, smartphone add-on) and the Flir E8 (around 5000€, standalone device). Figure 29 shows the thermal image of a Raspberry Pi from those three devices, unprocessed and processed with a filter. Of course, the Thermocam with a total hardware costs of only 300€ cannot beat the much more expensive FLIR camera. But in comparison with the Seek Thermal XR, it delivers solid proof results.

To conclude this report, it can be stated that the DIY-Thermocam V2 is a reasonable advancement to its predecessor. The on-device rendering quality has been improved, the device is easier to assemble, the visual camera delivers better and faster images and the firmware and PC software solutions have been further developed. Hopefully, this project will be helpful for other researchers and all people interested in a low-cost and open-source thermal imager.

Appendix

A.1 Serial protocol

This protocol describes how to communicate with the DIY-Thermocam over the USB serial connection

All commands are transmitted as decimal bytes. The number of bytes returned by each command is listed in the return part of each command. Whenever a command does not send data back, it will send its own command ID as acknowledge (ACK) if successful and zero (0) if not successful. In case data needs to be transmitted after the command itself, it is marked as payload in the description, followed by the number of bytes that need to be transmitted and their description.

- **100**: **CMD_START**
  - **Description:** Set the device into serial mode and waits for the next command from the list
  - **Returns** (1 byte): ACK or 0

- **200**: **CMD_END**
  - **Description:** Exit serial mode and set the device back to live mode
  - **Returns** (1 byte): ACK or 0

- **110**: **CMD_GET_RAWLIMITS**
  - **Description:** Get the lepton raw limits (14-bit) of the current frame
  - **Returns** (4 byte): minTemp MSB, minTemp LSB, maxTemp MSB, maxTemp LSB

- **111**: **CMD_GET_RAWDATA**
  - **Description:** Get the lepton raw values, 9600 byte (Lepton2) or 38400 byte (Lepton3)
  - **Explanation:** 14-bit raw values (0 … 16383) for each pixel, equivalent to infrared intensity
  - **Returns** (9600 / 38400 byte): 4800 / 19200 * (rawData[i] MSB, rawData[i] LSB)

- **112**: **CMD_GET_CONFIGDATA**
  - **Description:** Get the configuration data, that does not change if not adjusted manually
  - **Returns** (10 byte):
    - leptonVersion (1 Byte): 0 = Lepton2 shutter, 1 = Lepton3 shutter, 2 = Lepton2 no shutter

- rotationEnabled (1 Byte): 0 = Normal orientation, 1 = Device rotated by 180 degree
- colorScheme (1 Byte): Number definitions can be looked up in the file "GlobalDefines.h"
- tempFormat (1 Byte): 0 = Celsius, 1 = Fahrenheit
- spotEnabled (1 Byte): 0 = Disabled, 1 = Spot temperature displayed
- colorbarEnabled (1 Byte): 0 = Disabled, 1 = Colorbar displayed
- minMaxEnabled (1 Byte): 0 = Disabled, 1 = Min, 2 = Max, 3 = Both
- textColor (1 byte): 0 = White, 1 = Black, 2 = Red, 3 = Green, 4 = Blue
- filtertype (1 byte): 0 = Disabled, 1 = Gaussian blur, 2 = Box blur
- adjustLimits (1 Byte): 0 = Manual mode, limits fixed, 1 = Auto mode, limits adjust

- **113**: **CMD_GET_CALSTATUS**
  - **Description:** Get the calibration status from the device and seconds left for warmup
  - **Returns** (2 byte): calStatus (0 = warmup, 1 = standard, 2 = manual), seconds left for warmup

- **114**: **CMD_GET_CALIBDATA**
  - **Description:** Get the calibration data, offset first, then slope
  - **Explanation:** Required for raw-to-absolute temperature conversion with the following formula: *absTemp = rawValue * calibrationSlope + calibrationOffset*
  - **Returns** (8 byte): two float, split to four bytes each, LSB first, MSB at the end

- **115**: **CMD_GET_SPOTTEMP**
  - **Description:** Get the spot temperature from the MLX90614 sensor, in Fahrenheit or Celcius
  - **Returns** (4 byte): Float split into four bytes, LSB first, then MSB

- **116**: **CMD_SET_TIME**
  - **Description:** Set the time on the device, used to sync with PC time
  - **Explanation:** The temperature format is *"%Y-%m-%d %H:%M:%S\n"*, with %Y = year, %m = month, %d = day, %H = hour, %M = month and %S = second
  - **Returns** (1 byte): ACK or 0

- **117**: **CMD_GET_TEMPPOINTS**
  - **Description:** Get the temperature points that are shown on the screen
  - **Explanation:** First two bytes are index value (1 = (0,0) … 19200 = (159, 119)), second two bytes are the raw value. If the index is 0, the point is not set, otherwise evaluate the raw value

    - **Returns** (384 byte): 96 points (96 x 4), 2 bytes index, 2 bytes raw value, MSB first, then LSB

- **118**: **CMD_SET_LASER**
    - **Description:** Toggles the laser marker on the device (DIY-Thermocam V1 only)
    - **Returns** (1 byte): ACK or 0

- **119**: **CMD_GET_LASER**
    - **Description:** Get the current laser state (DIY-Thermocam V1 only)
    - **Returns** (1 byte): 0 = Disabled, 1 = Enabled

- **120**: **CMD_SET_SHUTTERRUN**
    - **Description:** Run the shutter flat-field-correction (FFC) on the FLIR Lepton
    - **Returns** (1 byte): ACK or 0

- **121**: **CMD_SET_SHUTTERMODE**
    - **Description:** Set the Lepton shutter FFC mode to automatic or manual
    - **Payload** (1 byte): 0 = Manual, 1 = Automatic
    - **Returns** (1 byte): ACK or 0

- **122**: **CMD_SET_FILTERTYPE**
    - **Description:** Set the type of filter used for the lepton raw values
    - **Payload** (1 byte): 0 = Disabled, 1 = Gaussian blur, 2 = Box blur
    - **Returns** (1 byte): ACK or 0

- **123**: **CMD_GET_SHUTTERMODE**
    - **Description:** Get the current shutter FFC mode from the FLIR Lepton
    - **Returns** (1 byte): 0 = Manual, 1 = Auto, 2 = None (no shutter attached)

- **124**: **CMD_GET_BATTERYSTATUS**
    - **Description:** Get the current battery status in percentage
    - **Returns** (1 byte): 0 = 0% (empty), 100 = 100% (fully charged)

- **125**: **CMD_SET_CALSLOPE**
    - **Description:** Set the calibration slope
    - **Payload** (4 byte): Float split into four bytes, LSB first, MSB at the end
    - **Returns** (1 byte): ACK or 0

- **126**: **CMD_SET_CALOFFSET**
    - **Description:** Set the calibration offset
    - **Payload** (4 byte): Float split into four bytes, LSB first, MSB at the end

- **Returns** (1 byte): ACK or 0

- **127**: **CMD_GET_DIAGNOSTIC**
  - **Description:** Get the diagnostic information containing the hardware status
  - **Explanation:** The following components are checked: Spot Sensor (0), Display (1), Visual Camera (2), Touch Screen (3), SD Card (4), Battery Gauge (5), Lepton I2C (6), Lepton SPI (7)
  - **Returns** (1 byte): Diagnostic byte, False when component not working: (diagnostic >> device) & 1

- **128**: **CMD_GET_VISUALIMG**
  - **Description:** Sends the visual image with 320x240 or 640x480 resolution as JPEG byte stream
  - **Payload** (1 byte): 0 = Low resolution (320x240), for streaming, 1 = High resolution (640x480), for saving
  - **Returns** (Variable bytes, as defined in length):
    - The first two bytes are the length, MSB first, then LSB
    - JPEG data starts with two bytes 0xFF & 0xD8 and ends with two bytes 0xFF & 0xD9
    - If the display rotation is enabled, 100 bytes of EXIF information are inserted, starting at offset 20 dec (0xFF & 0xE9) and ending at offset 119 dec, the picture information follows afterwards

- **129**: **CMD_GET_FWVERSION**
  - **Description:** Get the current firmware version installed on the device
  - **Returns** (1 byte): 100 equals 1.00, 255 equals 2.55

- **130**: **CMD_SET_LIMITS**
  - **Description:** Set adjust limits to lock limit or automatic mode
  - **Explanation:** When using lock limits, the raw value limits from the last frame are locked
  - **Payload** (1 byte): 0 = Lock limits, 1 = Auto mode, limits adjust
  - **Returns** (1 byte): ACK or 0

- **131**: **CMD_SET_TEXTCOLOR**
  - **Description:** Set the text color used to display the elements in live mode
  - **Payload** (1 byte): 0 = White, 1 = Black, 2 = Red, 3 = Green, 4 = Blue
  - **Returns** (1 byte): ACK or 0

- **132**: **CMD_SET_COLORSCHEME**
  - **Description:** Set current color scheme used for the conversion, see page 1 for numbers
  - **Payload** (1 byte): color scheme number

- **Returns** (1 byte): ACK or 0

- **133**: **CMD_SET_TEMPFORMAT**
  - **Description:** Set the temperature format to Celsius or Fahrenheit
  - **Payload** (1 byte): 0 = Celsius, 1 = Fahrenheit
  - **Returns** (1 byte): ACK or 0

- **134**: **CMD_SET_SHOWSPOT**
  - **Description:** Show spot temperature information in the image or not
  - **Payload** (1 byte): 0 = Disabled, 1 = Enabled
  - **Returns** (1 byte): ACK or 0

- **135**: **CMD_SET_SHOWCOLORBAR**
  - **Description:** Show color bar information in the image or not
  - **Payload** (1 byte): 0 = Disabled, 1 = Enabled
  - **Returns** (1 byte): ACK or 0

- **136**: **CMD_SET_SHOWMINMAX**
  - **Description:** Show the hottest, coldest or both points in the image or not
  - **Payload** (1 byte): 0 = Disabled, 1 = Min, 2 = Max, 3 = Both
  - **Returns** (1 byte): ACK or 0

- **137**: **CMD_SET_TEMPPOINTS**
  - **Description:** Set the temperature points to be shown on the screen
  - **Explanation:** First two bytes are index value (0 = (0,0) … 19200 = (159, 119)), second two bytes are the raw value. If the index is 0, the point is not set, otherwise evaluate the raw value
  - **Payload** (384 byte): 96 points (96 x 4), 2 byte index, 2 bytes raw value, MSB first, then LSB
  - **Returns** (1 byte): ACK or 0

- **138**: **CMD_GET_HWVERSION**
  - **Description:** Get the hardware version of the device
  - **Returns** (1 byte): 0 = DIY-Thermocam V1, 1 = DIY-Thermocam V2

- **139**: **CMD_SET_ROTATION**
  - **Description:** Sets the display rotation on the device
  - **Payload** (1 byte): 0 = normal orientation, 1 = 180° rotated
  - **Returns** (1 byte): ACK or 0

- **140**: **CMD_SET_CALIBRATION**
  - **Description:** Run the calibration method to calculate slope and offset

- **Returns** (102 byte): Status in percentage (0…100) and ACK when finished

- **141**: **CMD_GET_HQRESOLUTION**
  - **Description:** Get the HQ resolution status for the DIY-Thermocam V2
  - **Explanation:** The normal rendering resolution is 160 x 120, the HQ rendering resolution is 320 x 240
  - **Returns** (1): 0 = Normal rendering, 1 = HQ resolution rendering

- **150**: **CMD_FRAME_RAW**
  - Get a raw frame containing the raw data and additional information
  - First byte is the command ID:
    - 180 = push button was pressed short; user wants to save a thermal image
    - 181 = touch screen was pressed short; user wants to save a visual image
    - 182 = push button was pressed long; user wants to start/stop recording a video
    - 183 = no button pressed: normal frame, followed by the byte stream
    - If the command ID is 183, the following bytes are appended as payload: Raw data (9600 or 38400 byte), Raw limits (4 byte), Spot temperature (4 byte), Calibration data (8 byte)

- **151**: **CMD_FRAME_COLOR**
  - Command IDs equal to the normal raw-frame, the lepton values are converted to RGB565 colors
  - The raw-data to color conversion is done according to the current selected color scheme

- **152**: **CMD_FRAME_DISPLAY**
  - Command IDs equal to the normal raw-frame, but the payload is the display framebuffer only
  - On the DIY-Thermocam V2 with HQ resolution enabled, the size is 320 x 240 pixel (153600 byte), otherwise it is 160 x 120 pixel (38400 byte). Each pixel is represented by a RGB565 color (16-bit)

A.2 Raw data files structure

The structure of the raw data files (*.DAT) is defined as following:

- **Lepton raw values** - 9600 byte (Lepton2) or 38400 byte (Lepton3)
  - 4800 / 19200 raw values for the Lepton2 / Lepton3, split to two bytes, MSB first, then LSB
  - Raw value is 14-bit (0 … 16383) for each pixel, equivalent to infrared intensity
  - Difference can be made by the total file size (10005 byte Lepton2 vs. 38805 Lepton3)

- **Minimum temp** - 2 byte
  - 14-bit minimum raw value to apply the color scheme, could be fixed if in manual mode
  - Split to two bytes, MSB first, then LSB. Reconstruction analog to the lepton raw values
- **Maximum temp -** 2 byte
  - 14-bit maximum raw value, same format than the minimum temperature value

- **Spot sensor temp** - 4 byte
  - Object temperature measured by the spot sensor; Float, split to four bytes, LSB first
  - Format is Celsius or Fahrenheit, depending on the set temperature format

- **Color scheme** - 1 byte
  - Number of the color scheme, the image was created with
  - Number definitions can be looked up in the file “General/GlobalDefines.h”

- **Temperature format** - 1 byte
  - The temperature format is applied to all absolute temperatures
  - Celsius (0) or Fahrenheit (1)

- **Show spot** - 1 byte
  - Display the MLX90614 spot temperature information in an image or not
  - Disabled (0) or Enabled (1)

- **Show color bar** - 1 byte
  - Display the radiometric color bar in an image or not
  - Disabled (0) or Enabled (1), Disabled when in warmup (first 60 seconds)

- **Show minimum / maximum point** - 1 byte
  - Show the minimum and / or maximum temperature point(s) in an image or not
  - Disabled (0), Min only (1), Max only (2), Min & Max (3)

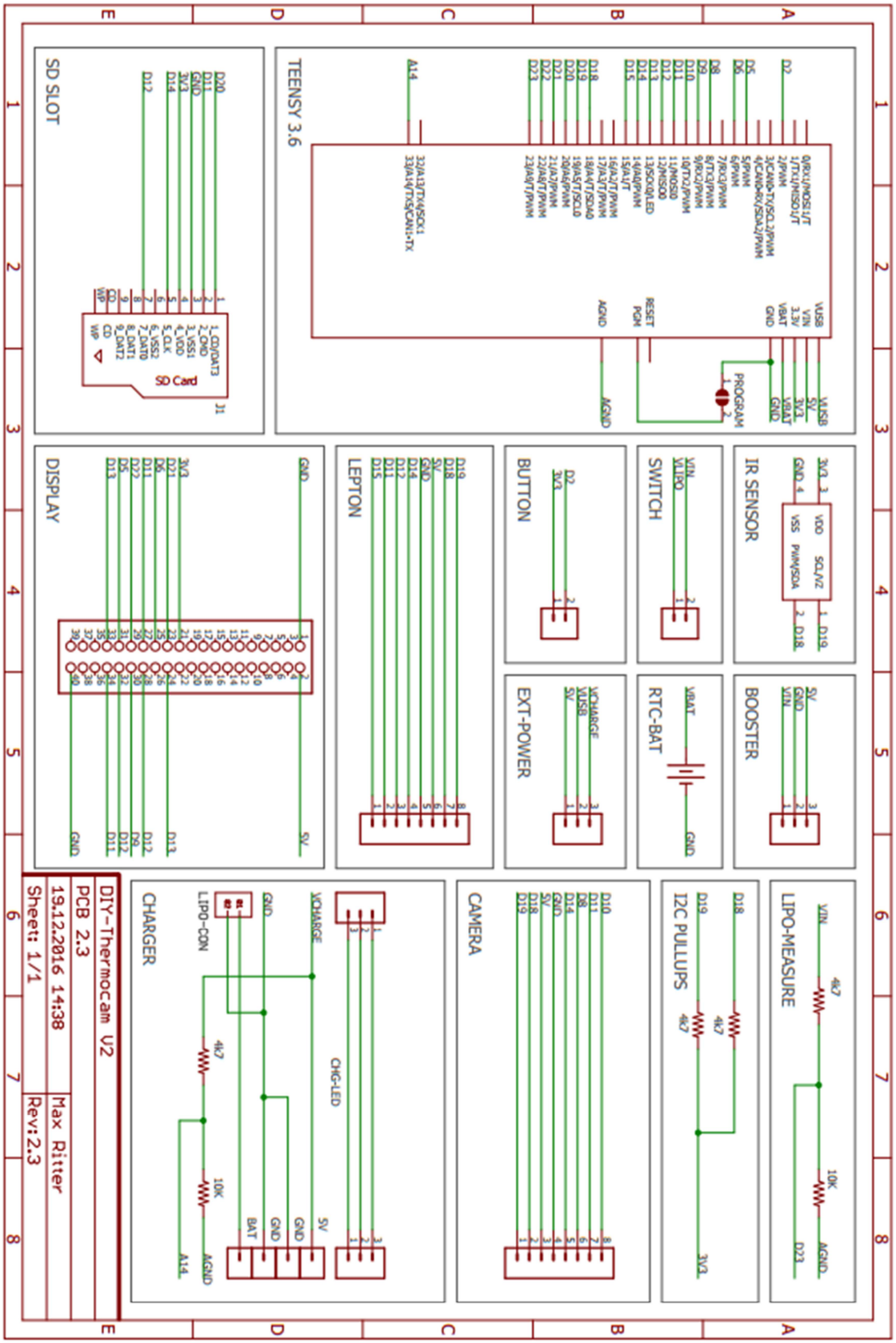
- **Calibration offset** - 4 byte
    - Offset for the raw-to-abs temperature conversion
    - Float, split to four bytes, LSB first. Reconstruction analog to the spot sensor temp

- **Calibration slope -** 4 byte
    - Slope for the raw-to-abs temperature conversion
    - Float, split to four bytes, LSB first. Reconstruction analog to the spot sensor temp

The absolute temperature can be calculated out of each raw value by using the following formula:
*absTemp = rawValue * calibrationSlope + calibrationOffset*

- **Temperature points** - 384 byte
    - 96 different temperature points, containing the index position and lepton raw value
    - They are manually defined by the user and could be skipped if not utilized
    - The first two bytes are the index position, second two bytes are the lepton raw value
    - Index position: (0,0) = 1; (159, 119) = 19200; 0 = not set, otherwise set; MSB first, then LSB

A.3 Schematic



A.4 Firmware functionality

 Change color: Switch between the different color scales applied to the thermal image.

 Change mode: There are three different operating modes: Thermal (only), visual (only) and combined.

 Change limits: You can choose between two modes: ‘Auto’ automatically finds the hottest and coldest value in every image, ‘Manual’ lets you choose the minimum and maximum temperature, between which the color scheme is applied.

 Load menu: Here you can view all images and videos stored on the SD card.

 Trigger shutter: Manually close and open the shutter on top of the FLIR Lepton sensor, in order to remove noise from the raw data and make the image uniform again.

 Settings menu: It allows you to adjust the basic device settings.

 Live display options: You can change what information is shown in live mode.

 HQ resolution: Choose between normal (160x120) or HQ resolution (320x240) for rendering.

 Display: Turn the display off in order to save battery, while preserving all settings as well as the calibration data. Touch the screen again to reactivate it.

 Calibration: Improves the accuracy of the color to absolute temperature conversion and should be performed after each start. During the calibration process, point the device to different hot and cold objects to ensure a good calibration. Repeat the calibration for different surrounding areas.

 Adjust combined: Only available in visual or combined mode. Move the visual image on top of the thermal image to get a better alignment. Can store up to three presets and allows the change alpha level.

 Hot / cold mode: Only available in thermal mode. Mark the hottest or coldest temperatures with an adjustable color. You can also select the level, above or under the marking should be done.

 Temperature points: Add or remove live temperature reading points for every position inside the image.

List of literature

[1] FLIR, “History of Thermal Imaging,” [Online]. Available: http://www.securitysales.com/files/Advertise/SS7dumies.pdf. [Accessed 28 12 2016].

[2] Optris, “Grundlagen der berührungslosen Temperaturmessung,” 2001. [Online]. Available: http://www.optris.de/downloads-infrarotkameras?file=tl_files/pdf/Downloads/Zubehoer/IR-Grundlagen.pdf. [Accessed 28 12 2016].

[3] S. Krösche, Physikalische Grundlagen und Anwendungen der InfrarotThermografie, Viscardi-Gymnasium Fürstenfeldbruck, 2010.

[4] FLIR, “Wärmebildtechnik - Ratgeber für industrielle Anwendungen,” 2011. [Online]. Available: http://www.flirmedia.com/MMC/THG/Brochures/T820264/T820264_DE.pdf. [Accessed 28 12 2016].

[5] FLUKE, “http://www.iv-krause.de/media/pdf/Kataloge/Fluke/2014_01_01_Einf-hrung-in-die-Thermogra.pdf,” [Online]. Available: Einführung zu den Thermografie-Prinzipien. [Accessed 28 12 2016].

[6] C. Linde, Wärmebildkamera: Physikalische Grundlagen - Sensoren, 2013.

[7] FLUKE, “Einführung zu den Thermografie-Prinzipien,” [Online]. Available: http://www.iv-krause.de/media/pdf/Kataloge/Fluke/2014_01_01_Einf-hrung-in-die-Thermogra.pdf. [Accessed 29 12 2016].

[8] Testo, “Pocket-Guide Thermografie,” 2013. [Online]. Available: http://www.produktinfo.conrad.com/datenblaetter/1200000-1299999/001234885-an-01-de-POCKET_GUIDE_TESTO_WAERMEBILDKAMERA_AKTI.pdf. [Accessed 29 12 2016].

List of illustrations