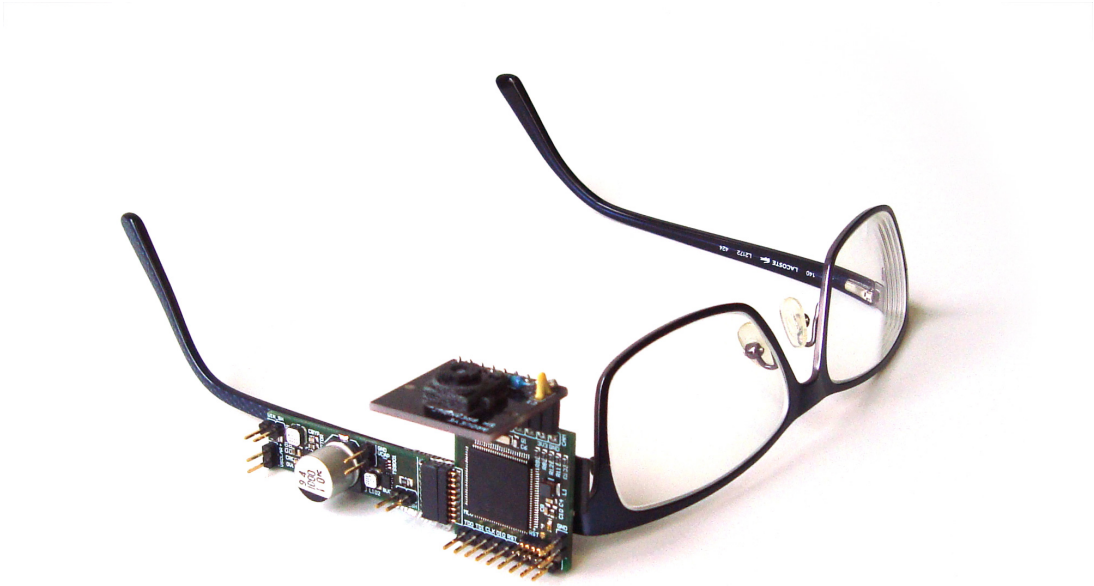
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Institut für
Technische Informatik und
Kommunikationsnetze

Master Thesis

Transiently Powered and Wearable Device for Ultra-Low-Power Vision Sensing

Thomas Schalch



March 2016 to August 2016

MA-2016-43

Tutors: Andres Gomez, Lukas Sigrist

Computer Engineering Group (TEC), ETH Zurich
Prof. Dr. Lothar Thiele

Abstract

In a traditional energy harvesting approach, large batteries are required to guarantee the continuous operation of the system. In contrast, transiently powered systems harvest and buffer at most the energy required for one task completion. The system operation thus highly depends on the currently available input power. The lack of a battery allows the design of small-sized, low-cost and efficient energy harvesting nodes, that can run without the need for maintenance at all. This thesis demonstrates the realization of computationally intensive applications in battery-less devices. Namely, a battery-less and wearable vision sensor for estimating the walking speed of a person is designed and implemented. The vision sensor harvests energy from a solar panel using the concept of transiently powered systems. The hardware and firmware is highly optimized for ultra-low power consumption and high performance. For further enhancing the performance of the transiently powered system, a non-volatile memory hierarchy (NVMH) consisting of an FRAM buffer and a microSD Card is presented. The NVMH increases the performance of the prototype by factor 1.8. The final velocity sensor acquires more than 10 velocity estimations per minute at an ultra-low input power of only 1 mW and stores the estimations as well as the captured images on a microSD Card. At an input power of 40 mW, up to 8 sensor evaluations per second can be achieved.

Contents

List of Figures

Chapter 1

Introduction

Innovative energy harvesting concepts are currently in great demand to satisfy the need for efficiently powering remote nodes in the Internet of Things (IoT) or small wearable sensor nodes. In a traditional energy harvesting approach, the system is continuously powered by a battery, whereas the battery is recharged by harvesting energy (e.g. solar, kinetic or thermal energy) whenever excess energy is available. The system runs continuously, as far as the energy source is powerful enough to provide in average more power than the load consumes and the battery is large enough to bridge time intervals in which too little input power is available. Unfortunately, meeting those specifications often results in excessively large batteries and energy harvesting sources. When building such systems, the battery is a critical element in terms of size, losses, costs and environmental sustainability.

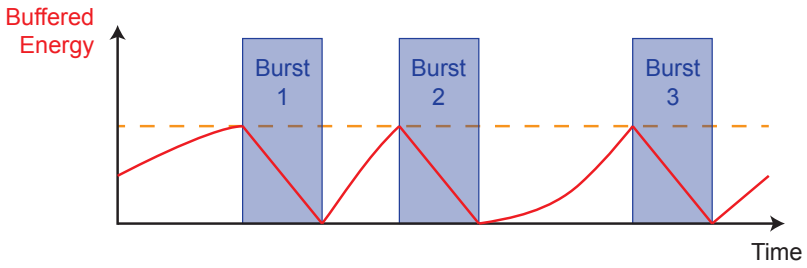


Figure 1.1: Transiently powered systems buffer harvested energy until enough energy is available for the execution of one single burst.

Typical sensor nodes operate with a low duty cycle, meaning that they acquire some data and then sleep for a long time until the next data acquisition is scheduled. Making the schedule of such an energy harvesting node highly dependent on the currently available input power is the idea of *Transiently Powered Systems*. Even at very low input power levels, such systems can still perform sporadic computations. Transiently powered systems harvest and buffer energy until enough energy is available for executing one single burst. After the burst execution, the system is shut down until enough energy for the next burst is harvested (see Figure 1.1). A so-called Energy Management Unit (EMU) [13] controls the energy harvesting and the burst generation (see Figure 1.2). In such systems, the energy buffer capacity can be reduced to the amount of energy needed for the execution of one single burst. The energy buffer is by orders smaller and its function is completely different from that of a battery in a traditional battery-powered device. That is why transiently powered systems are also referred to as *Battery-less Systems*.

The concept of transiently powered systems is very promising for powering small remote nodes. Other than in battery-based devices, the energy harvesting part can be scaled independently of the load part of the system, because the minimal size of the energy harvesting part is not limited by the active load power consumption. It is typical for transiently powered systems, that they operate with input powers that are significantly lower than the active load power. This allows the development of small-sized, low-cost and efficient energy harvesting nodes, that can run without needing maintenance at all.

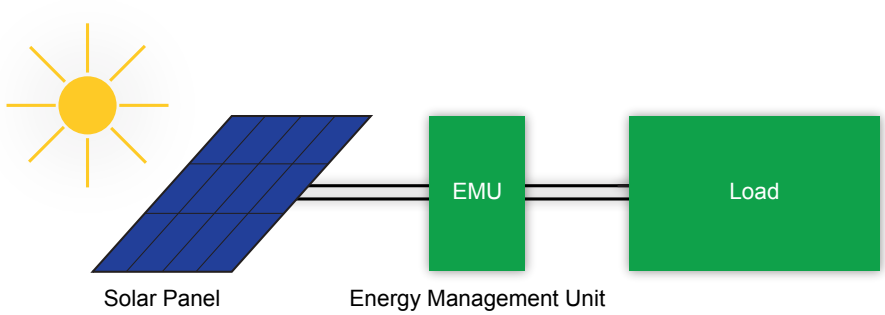


Figure 1.2: Transiently powered systems use an Energy Management Unit (EMU) instead of a battery for providing energy to the load.

The goal of this master thesis is to show that using the concept of transiently powered systems, a computationally intensive application can be executed in a small, battery-less sensor node. In particular, a wearable, transiently powered vision sensor to estimate a person's walking speed is developed. The final system supports both continuously powered operation with a battery and transiently powered operation using a solar panel. The vision sensor is able to acquire images, estimate the displacement between the images and log the images as well as the results of the image processing in a long-term storage device. The prototype of the vision sensor is small enough to be attached to someone's glasses. Operating the vision sensor continuously with a battery allows to track the covered distance of the walking person. As a transiently powered system, the vision sensor can also deal with the much lower, highly volatile input power of a solar panel. It then still provides sporadic estimations for the walking speed that can be used e.g. for context recognition.
If the wearable vision sensor is combined with other transiently powered sensor nodes that harvest for example kinetic energy from doing sports or thermal energy from human body heat, a human body sensor network could be formed. This could be used for example to extract useful information for context recognition or health monitoring. Each sensor node would then only contribute to the final result if it provides useful information. For example if too little light is available, the camera of the vision sensor could not capture any useful images anyway.
The development of the vision sensor prototype is documented in three chapters. Chapter 2 focuses on the concept of transiently powered systems, introduces the novel concept of a non-volatile memory hierarchy (NVMH) and specifies the design of the transiently powered velocity estimation sensor. Chapter 3 documents the implementation of the wearable vision sensor prototype and describes the chosen hardware, the firmware implementation and the PCB design. Chapter 4 discusses the motion estimation algorithm and its implementation. Finally, the vision sensor prototype is evaluated in Chapter 5.

Chapter 2

Transiently Powered Vision Sensor

A system is called transiently powered if its operation is a function of the currently available input power. Transiently powered systems are driven by energy bursts which power the system only for a limited amount of time. Such systems accumulate harvested energy, until enough energy is available for the execution of one single burst (see Figure 1.1). Afterwards, the system is shut down completely again, until enough energy is accumulated for the next burst execution. The length of the time interval between two bursts depends on the currently available input power. This type of operation directly leads to two challenges for the design of transiently powered systems:

Constraint (1): There is no control over the length of the time interval between two bursts, since this only depends on the currently available input power. The application needs therefore to be split into *separate bursts with no temporal dependencies*.

Constraint (2): Between two bursts, the system is completely shut down and all peripherals are powered off. Therefore, there is a need for using *non-volatile memory* technologies for retaining the state of the system and for long-term data logging.

A main feature of transiently powered systems is the need for only a very small energy buffer compared to battery-based systems. To guarantee the proper operation of transiently powered systems, the energy buffer needs to be able to store at least the energy required for one burst completion. Therefore, the burst with the highest energy requirement in the application determines the size of the energy buffer.
In [13], Gomez et al. introduce some fundamental concepts of transiently powered systems and the new concept of Dynamic Energy Burst Scaling (DEBS). A short overview over DEBS is given in Chapter 2.1 in order to motivate the use of this concept for the transiently powered vision sensor prototype. Chapter 2.2 introduces different non-volatile memory technologies to account for *Constraint (2)*. The concept of a Non-Volatile Memory Hierarchy (NVMH) to enhance the performance of transiently powered systems is introduced in Chapter 2.3. In Chapter 2.4, the application for estimating velocities using a transiently powered vision sensor is presented.

2.1 Dynamic Energy Burst Scaling (DEBS)

Typical cyber-physical systems consist of a microcontroller with its peripherals like memories, sensors or transceivers. Microcontrollers can usually operate in a wide voltage range, but they are most efficient at lower supply voltages. The peripherals can have completely different voltage requirements. Often, the highest minimum voltage is then taken for the whole application design to avoid converter losses for multiple voltage domains. If the application is divided into several tasks in which only some of the peripherals are used, each task has its own optimal supply voltage for lowest energy consumption. The idea of *Dynamic Energy Burst Scaling (DEBS)* is to supply each task with its optimal supply voltage, or in other words to track the load's optimal power point in order to minimize its energy consumption [13].
The advantage of DEBS is that the power point of the load and the source are completely decoupled. This allows harvesting the maximal possible power from the energy harvesting source, whereas each task is supplied with the lowest possible supply voltage to minimize the energy consumption for each task.

2.2 Non-Volatile Memory Technologies

According to *Challange (2)*, the memories in transiently powered systems are only supplied with power during an energy burst. Any data saved in volatile memories get lost after completion of the burst. For a transiently powered vision sensor, long-term logging of large amounts of data is requested. On the other hand, there is also a need for a fast, energy efficient memory for retaining the state of the system between two energy bursts.

Flash Memory: SD Cards

A very common non-volatile memory technology is flash. It offers a very high storage density and memories with huge storage capacities are available. Its integration into SD Cards makes flash to a widespread storage technology for storing large amounts of data in embedded systems. Using SD Cards, data can be transferred to a PC in an easy and fast way. However, the flash storage technology has some drawbacks: The power consumption is high and flash has a comparably low endurance of $10^4 - 10^5$ write cycles for each block [29]. Furthermore, SD Cards require an expensive initialization procedure after powering up, causing a large and unpredictable overhead in time and energy, before the actual memory operations can be performed. Unfortunately, this initialization procedure has to be repeated often in a transiently powered system, because the SD Card is powered off after each burst. Therefore, storing small amounts of data on SD Cards is very costly in transiently powered systems.

FRAM Technology

A promising non-volatile memory technology is Ferroelectric Random Access Memory (FRAM), which features a high endurance of 10^{14} read/write cycles and a very low power consumption (300 µA active current at 1 MHz SPI frequency [8]), which is in the order of 100 times lower than the power consumption of SD Cards. Unlike SD Cards, FRAM chips do not need an initialization procedure, have a fast power-up time and the minimal transfer unit for read/write operations is one single byte. Therefore, also small amounts of data can be stored efficiently.

The drawback of FRAM technology is the low storage density. Today, the largest commercially available FRAM SPI chip from Cypress has a storage capacity of 2 Mbits. This is clearly too small for long-term logging in data-intensive applications. However, the ultra-low power consumption and the minimal initialization overhead make FRAM technology ideal for storing small amounts of data in transiently powered systems that require frequent fast and deterministic read/write operations.

	SD Card (flash)	FRAM
storage capacity	$\sim$ GiB	$\leqslant$ 256 KiB
power consumption	$\sim$ 100 mW	$\sim$ 1 mW
endurance	$10^4 - 10^5$ cycles	10^{14} cycles
initialization overhead	large	minimal

Table 2.1: Comparison of the two non-volatile memory technologies flash (in SD Cards) and FRAM.

2.3 Non-Volatile Memory Hierarchy (NVMH)

To account for both requirements - a fast, ultra-low-power memory for state retention and a large storage device for long-term data logging - a memory hierarchy is one solution. The fast and energy efficient FRAM is an optimal candidate for buffering small amounts of data between the bursts, whereas an SD Card can provide a sufficiently large storage capacity for long-term data logging. Because of the high initialization costs of SD Cards, its use should be kept at a minimum. The idea of the NVMH is to use FRAM as a small data cache before the SD Card. To illustrate the operation of the non-volatile memory hierarchy in transiently powered systems, the two cases of using only an SD Card as non-volatile memory, and using a NVMH consisting of an SD Card an an FRAM chip are compared for a typical sensor node application.

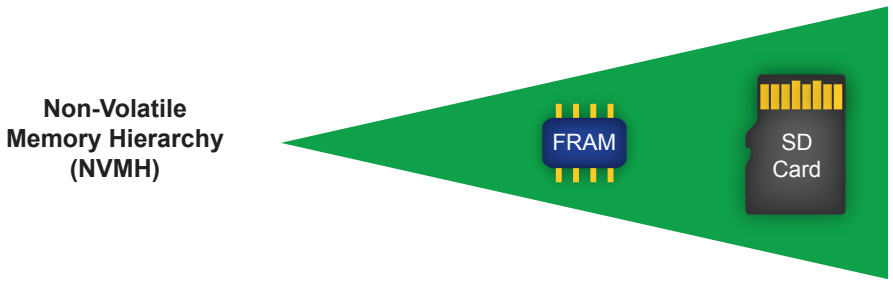


Figure 2.1: The concept of a Non-Volatile Memory Hierarchy (NVMH) consisting of an FRAM buffer and an SD Card can significantly improve the performance of a transiently powered system.

SD Card only

If only an SD Card is used as a non-volatile memory in transiently powered systems, a typical sensor node application could work as follows: The sensor acquires some data, processes it, and then stores the data onto the non-volatile memory, which is in this case the SD Card. According to *Constraint (1)*, this needs to be done in one single burst (see

Figure 2.2). The costly initialization procedure for the SD Card has to be repeated for every sensor reading.

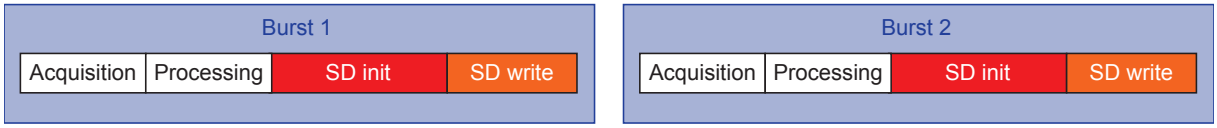


Figure 2.2: Using only an SD Card as a non-volatile memory, the costly initialization procedure has to be repeated for every sensor reading.

Non-Volatile Memory Hierarchy: SD Card and FRAM

The NVMH consists of an FRAM buffer and an SD Card. Using this memory hierarchy, a typical sensor node application acquires some data, processes it and stores it in the FRAM buffer (see Figure 2.3). This type of burst can be repeated n times, depending on the size of the FRAM buffer. If the FRAM buffer is full after n sensor readings, a special burst is introduced, that initializes the SD Card and flushes the content of the FRAM buffer onto the SD Card (see Figure 2.3). The expensive initialization costs can thus be spread over many sensor readings, depending on the size of the FRAM buffer. This reduces the average amount of energy per sensor reading and can improve the overall performance of transiently powered systems significantly.

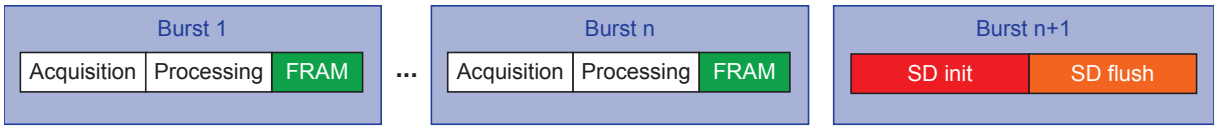


Figure 2.3: Using a NVMH consisting of an FRAM buffer and an SD Card can spread the large SD Card initialization costs over many sensor readings.

The energy overhead caused by the additional FRAM activities is minimal due to the ultra-low power consumption of the FRAM chip. The size of the FRAM buffer is currently limited by the available storage capacities of FRAM chips. However, the size of the FRAM buffer should not be arbitrarily increased anyway. The burst for flushing the FRAM buffer onto the SD Card will likely be the most energy-consuming burst in a sensor application. The energy consumption of this burst increases with the size of the FRAM buffer and so does the size of the needed energy buffer in the transiently powered system. A bigger energy buffer can lead to significantly longer cold-start times and a larger form-factor of the device. So in typical applications, choosing an appropriate size of the FRAM buffer will be a trade-off between an increase of the performance and the size of the energy buffer. The performance enhancement using a NVMH is evaluated in Chapter 5.2 for the transiently powered vision sensor.

2.4 Transiently Powered Visual Velocity Estimation Sensor

In this project, a transiently powered vision sensor shall be designed that can be attached to someone's glasses to estimate the walking speed of the person. The previously introduced concepts DEBS and NVMH shall be used for an enhanced efficiency of the transiently powered system.

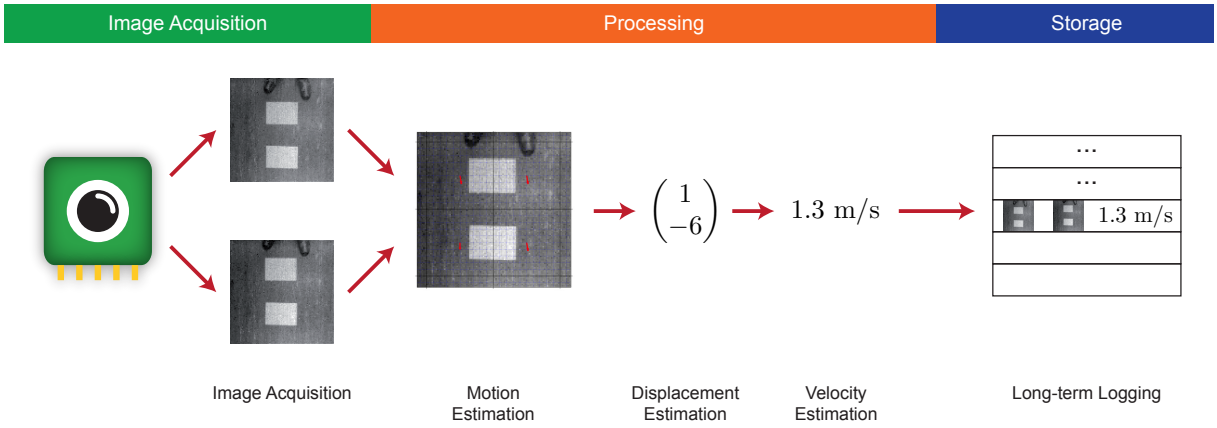


Figure 2.4: Generic function of the visual velocity estimation sensor.

Figure 2.4 shows the generic function of a visual velocity estimation sensor, which can be divided into three parts: *Image Acquisition*, *Processing* and *Storage*.

Image Acquisition

If the vision sensor is attached to someone's glasses, a camera needs to face down onto the floor to capture the area in front of the person. According to *Constraint (1)*, it is not possible to acquire a continuous sequence of images. But the least amount of images that needs to be acquired in a sequence for being able to recover a displacement is two. Therefore, two images with a well-known, constant time difference need to be acquired within the same burst to meet the temporal condition formulated in *Constraint (1)*.

Processing: Motion Estimation

Using the two acquired image frames, a velocity estimation needs to be calculated. This is done using a motion estimation algorithm as discussed in detail in Chapter 4. The result of the motion estimation algorithm is a so-called optical flow field, a vector field in which every vector indicates the displacement of the corresponding part of the image between the two frames. This optical flow field is then reduced to one single displacement vector indicating the displacement between the two frames. This displacement can be scaled to a final velocity estimation (see Chapter 4.3).

Storage

The final velocity estimation as well as the images shall be logged on a long-term storage device, namely on a microSD Card. To reduce the energy costs per velocity estimation, a non-volatile memory hierarchy (NVMH) is used, consisting of a microSD Card and an FRAM chip according to Chapter 2.3.

Chapter 3

Wearable Device for Ultra-Low-Power Vision Sensing

Figure 3.1 shows an overview of the hardware implementation of the wearable vision sensor. The hardware can be divided into two parts according to their functionalities. First, an Energy Management Unit (EMU) is responsible for energy harvesting and power management. The EMU accumulates the harvested energy from a solar panel in an energy buffer and controls the energy burst generation for the transiently powered vision sensor. The second part is an Image Processing Unit (IPU), which can acquire images, process the captured images and store data in different memories.

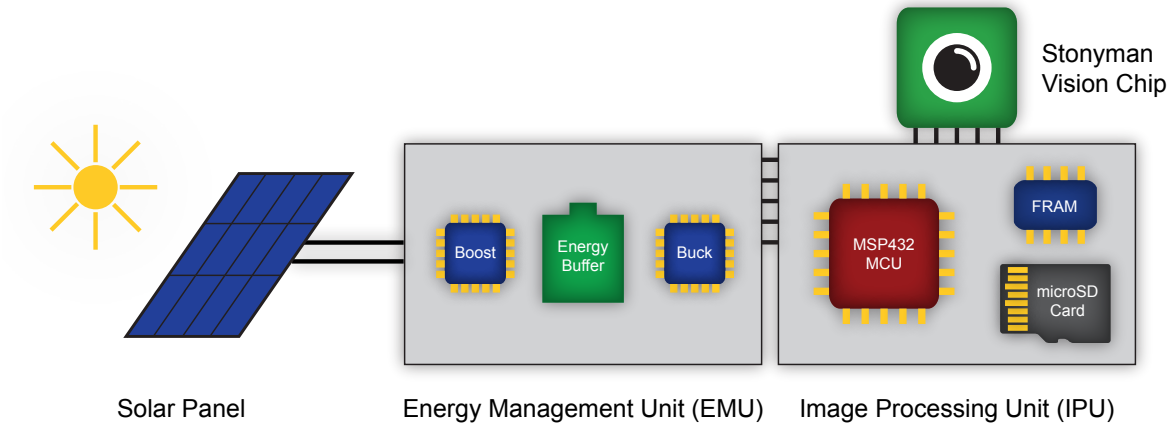


Figure 3.1: Overview over the hardware implementation of the wearable vision sensor.

The IPU is specifically designed as part of this master thesis and will be discussed in Chapter 3.1. The EMU has been proposed in [13] and only a few adoptions are made for integrating it in a small, wearable device. The EMU is described in Chapter 3.2. Chapter 3.3 covers the implementation of the vision sensor tasks in the IPU and the interface from the IPU to the EMU. In Chapter 3.4, a first prototype is presented, that is used for evaluation, development and debugging. The final wearable vision sensor is described in Chapter 3.5.

3.1 Image Processing Unit (IPU)

3.1.1 Microcontroller

MSP is a family of microcontrollers (MCUs) from Texas Instruments (TI) designed for ultra-low-power applications [19]. A main feature of those MCUs is their flexibility in choosing different clock sources, oscillators and voltage regulators, allowing to achieve an optimal trade-off between power consumption and performance. The MSP MCUs permit a fast transition between various low-power modes (LPMs). Those LPMs help optimizing the power consumption by enabling only the required peripherals and clocks.
In this project, two different MCUs from the MSP family are considered: the *MSP430FR5969* (further referenced as MSP430) [17] and the *MSP432P401R* (MSP432) [18]. The MSP430 has a built-in non-volatile FRAM instead of the commonly used volatile SRAM. This is a big advantage in transiently powered systems, as there is no need for external state retention even when powering the device off. In terms of internal peripherals, both MCUs are similar. However, the MSP432 features an ARM Cortex-M4F core instead of the custom architecture from Texas Instruments in the MSP430. The MSP432 can be clocked up to 48 MHz and is optimized for both, high performance and low power consumption, which results in a significantly higher performance than the MSP430. The MSP432 is currently the lowest-power MCU featuring the ARM Cortex-M4F architecture [19].
The ultra-low power consumption of the MSP432 despite its high performance makes the MSP432 ideal for the transiently powered vision sensor application.

3.1.2 Centeye Stonyman Vision Chip

The image sensor used in this project is a Stonyman vision chip from Centeye, Inc [7]. Its ultra-low power consumption (around 2mW at 3V supply voltage [27]) makes it unique amongst comparable commercially available image sensors and makes it a popular choice for ultra-low-power vision sensing projects. Examples using the Stonyman vision chip are iShadow (a wearable, real-time mobile gaze tracker [22]), KinetiSee (a wearable camera acquisition system with a kinetic harvester [27]) or a vision-based space landing control for a miniature tailed robot [30]. The Stonyman vision chip was also sold under the name ArduEye as an extension for the famous Arduino boards and is in this context used for many robot projects [11] [25].

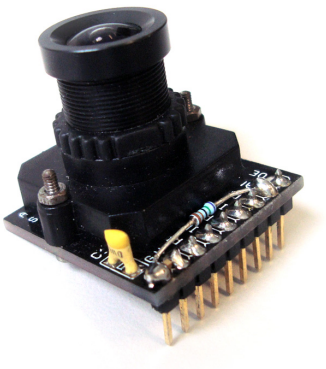


Figure 3.2: Stonyman vision chip from Centeye, Inc. with a mounted lens.

Hardware and Interface

The Stonyman vision chip consists of a square pixel array with a size of 112×112 pixels. The interface has five digital lines going from the MCU to the vision chip, plus one analog signal containing the pixel signal which needs to be digitized by the MCU's analog-to-digital converter (ADC). Using the five digital lines, the MCU can send commands to the vision chip and select one specific pixel. A voltage that is a logarithmic function of the brightness striking that pixel is applied to the analog output of the vision chip. This allows a random access to each pixel.

Two of the five digital lines allow to select an internal register of the vision sensor: *RESP* (Reset Pointer) and *INCP* (Increment Pointer). A pulse on the corresponding line resets the register pointer to zero or increments it by one. Two other digital lines are used to reset or increment the value in the currently selected register: *RESV* (Reset Value) and *INCV* (Increment Value). Amongst other registers for various settings, there is one register for choosing the currently selected column in the pixel array (*COLSEL*) and one register for choosing the currently selected row in the pixel array (*ROWSEL*). The signal of the currently selected pixel is applied to the analog output. After a settling time that needs to be determined empirically, the analog pixel value can be sampled using the ADC of the MCU. The fifth digital line (*INPHI*) operates an internal amplifier that is not used in this project, since it adds too much noise. An Output Enable pin (*OE*) exists, that can optionally be used for accessing multiple vision chips in the same circuit.

The vision chip can therefore be connected to an MCU using only five digital lines, one analog line, plus the power supply. Only a few additional components are necessary, namely the use of bypass capacitors to suppress noise in the supply voltage. Any fluctuations in the supply voltage are translated to the analog output signal and will cause noise in the image. A 1nF and a 0.1µF capacitor are recommended in the data sheet. In this project, an additional capacitor with a value of 4.7µF has significantly improved the image quality. Furthermore, as recommended in the data sheet, a resistance of 7.5kΩ is used between the analog output and ground to decrease the output impedance of the vision chip.

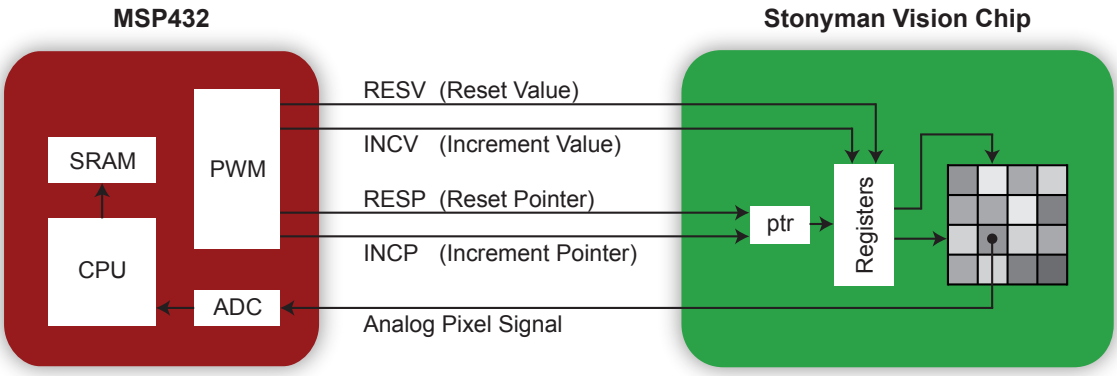


Figure 3.3: Interface of the Stonyman vision chip to the MSP432 MCU.

Firmware for the MSP432

The Stonyman vision chip is operated using the MSP432 MCU. For that, an existing firmware for the MSP430 was adapted for the MSP432. The pulses on the digital lines for selecting the current pixel are periodic: After reading a pixel, a pulse on the *INCV* line increases the value in the column select register to select the next pixel. After a complete column is read, the column register value needs to be reset to zero (using *RESV*) and the row select register is increased by 1. Using the timers of the MSP432 in PWM mode, the pulses for reading all 12544 pixels of the vision chip are generated without any further interaction of the CPU. The ADC runs in parallel and continuously samples and digitizes the analog output signal for each pixel. Unfortunately, the DMA of the MSP432 can only transfer data in chunks of at most 1024 items before a reinitialization is necessary. So it is not possible to transfer the whole image within one DMA transfer. Because a reinitialization would require to stop the timers and the ADC, the time overhead for using the DMA would be big. Therefore, the CPU is used to copy the digitized pixel values from the ADC memory to the SRAM, using an interrupt that is triggered every time an ADC cycle is completed.

After a pixel is selected, the analog output signal of the vision chip needs some time to settle down. According to the data sheet, a time delay of a couple of microseconds is recommended before sampling the analog output signal [7]. Some experiments showed that time delays below 2µs lead to noticeably more noise in the image and a reduced dynamic range. If one pixel is acquired every 2µs, the acquisition of the whole image ($112 \times 112 = 12544$ pixels) lasts approximately 25.1ms, which corresponds to a frame rate of 40 frames per second (fps). So the maximal achievable frame rate is limited by the Stonyman vision chip and should not exceed 40 fps.

For achieving such a high frame rate, one ADC cycle on the MSP432 needs to be completed every 2µs. The length of an ADC cycle is determined by the ADC clock frequency and the ADC resolution. An ADC resolution of x bits results in a quantization of the analog signal in steps of $V_{ref}/2^x$, where V_{ref} is an internal reference voltage. In the initial firmware for the MSP430, the highest possible resolution of 12 bits was used and every pixel has to be stored in an unsigned integer variable with a length of 16 bits. However, because the image is relatively noisy anyway, the resolution can be decreased. This was done in the firmware for the MSP432. The internal reference voltage V_{ref} is set to the lowest possible value of 1.2V and an ADC resolution of 10 bits is used. Because the voltage level of the analog output signal is between 0V and 0.3V for indoor light conditions, the two most significant bits of the sampled pixel values can be truncated. This leads to a 8-bit resolution of the pixel values between 0V and 0.3V and has the advantage that every pixel value can be stored in an unsigned integer variable with a length of 8 bits. The memory requirement for storing one image is therefore reduced by factor two, compared to the initial firmware implementation. Furthermore, the further image processing is much faster for 8-bit integer values than for 16-bit integer values (see the implementation of the motion estimation algorithm in Chapter 4.3). One ADC cycle with a resolution of 10 bits lasts 17 ADC clock cycles. An ADC clock frequency of 8 MHz is needed to achieve a frame rate of 37.5 fps. After each ADC conversion, an interrupt is triggered to copy the current pixel value from the ADC memory to the SRAM. For a proper operation of this copy process, the CPU has to run three times faster than the ADC, namely with 24 MHz.

Fixed Pattern Noise

Illuminating the whole vision chip with an uniform intensity should theoretically produce the same analog output voltage for all pixels. However, this is not the case. Due to process variations, each pixel exhibits an individual, temporally constant offset in the output voltage. This kind of image sensor noise is called **Fixed Pattern Noise (FPN)** [7]. The offset of each pixel is a function of the intensity level, the supply voltage and the configuration of the Stonyman vision chip. The supply voltage is kept constant using a reliable voltage regulator and bypass capacitors and the configuration is always the same in this project. So the FPN is constant over time for a given illumination level (e.g. indoor conditions). Removing the FPN from a captured image is therefore relatively easy. A mask I_{mask} is acquired while illuminating the vision chip uniformly, e.g. by covering the vision chip with a white sheet of paper. This mask is acquired once and then stored in memory. After each acquisition of a raw image I_{raw}, the FPN compensated image I_{comp} is computed as

$$I_{comp} = I_{mask} - I_{raw} \tag{3.1}$$

Figure 3.4: Example for the FPN compensation of an acquired image.

Figure 3.4 shows an example of an acquired raw image, a mask and an FPN compensated image. The computation is done on the MCU using a previously measured mask that is stored in the memory.

FPN Compensation on the MSP432

The fixed pattern noise can be removed from a raw image I_{raw} after the acquisition, using the formula in Equation 3.1 and the mask I_{mask} stored in the internal memory of the MSP432. This obviously causes an additional time overhead. Because the CPU is active during acquisition anyway, the question comes up, if the FPN compensation can be done on-line right during acquisition. This is indeed possible, but the CPU frequency has to be increased to 48 MHz instead of 24 MHz for maintaining the same ADC clock frequency of 8 MHz and a frame rate of 37.5 fps. The on-line FPN compensation is faster and causes no additional overhead in time. However, the evaluations in Chapter 5.1 show that acquiring one image with on-line FPN compensation consumes more energy than performing the acquisition and the FPN compensation in a sequence with a lower CPU frequency. This can be explained by the significantly higher power consumption of the MCU for higher CPU frequencies. A detailed evaluation of the two cases is given in Chapter 5.1 and in Table 5.1.

3.1.3 Memories

Non-volatile memories play a key role in transiently powered systems. As explained in Chapter 2.3, two different non-volatile memory technologies are considered in this project: flash and ferroelectric random access memory (FRAM). For that, a microSD Card and an FRAM chip need to be introduced as peripherals for the MSP432.

Flash: microSD Card

SD Cards are based on flash memory technology and are commonly used for transferring large amounts of data from an embedded system to a PC. SD Cards are usually operated with a file system (e.g. FAT), but it is also possible to read/write raw data in blocks of 512 bytes without any file system. SD Cards are specified by the so-called SD Group in [3]. The access to SD Cards happens either over a special SD bus protocol or the common SPI bus protocol. The memory is organized in blocks of 512 bytes, which is the basic data transfer unit. Devices like video cameras or PCs use the SD bus protocol to achieve the maximal data transfer rates. However, since lots of MCUs support the access to an SPI bus with built-in peripherals, the access to SD Cards using the SPI bus protocol is often much easier in embedded systems.
The firmware for accessing SD Cards and microSD Cards with the MSP432 using the SPI bus is based on a generic FAT file system module for embedded systems called FatFs [1]. This is a platform-independent file system module that implements all common functions for accessing files using the FAT file system on any type of storage device. It is completely separated from the low-level disk I/O layer. The low-level access to the devices needs to be implemented specifically for the MSP432 and SD Cards. For the low-level drivers, some generic examples in the FatFs module [1] and an initial implementation for the MSP432 [2] is used. However, this initial implementation did not work reliably at all and many improvements were necessary for a proper operation.
The low-level disk driver is developed according to the specifications from the SD Group [3]. For accessing an SD Card using the SPI bus protocol, a specific initialization procedure has to be conducted. After the initialization, a command has to be sent to announce the read/write operation of one or multiple blocks, followed by the data in blocks of 512 bytes. The final firmware for the MSP432 allows the access to SD Cards or microSD Cards with a FAT file system using the FatFs system module, or without any file system by reading/writing blocks of 512 bytes using the own implementation of the low-level disk I/O driver.

FRAM Chip

FRAM chips are ideal for non-volatile memory applications that require frequent and fast read/write operations of small amounts of data. The FRAM chip FM25V10 with 1 MBits storage capacity is used in this project [8]. The memory chip can be accessed over SPI with clock frequencies of up to 40 MHz.
The firmware implementation for the MSP432 is relatively easy, since the FRAM chip does not need any initialization after power-up and the commands are simple. For a read or write operation, the MCU sends the corresponding command(s), followed by the memory address of the data. The data is then transferred from the FRAM chip to the MCU or from the MCU to the FRAM chip respectively. There is no need for specifying the amount of data to read/write, because the chip simply reads/writes the amount of data following the corresponding commands until the Chip Select (CS) line is disabled by the MCU. The

FRAM chip supports SPI clock frequencies of up to 40 MHz, but the MSP432 only up to 16 MHz, so the maximal data transfer rate is limited by the MCU.

3.2 Energy Management Unit (EMU)

The Energy Management Unit (EMU) harvests energy from a source (e.g. a solar panel), stores it in an energy buffer and controls the energy burst generation for the vision sensor. The EMU used in this project was presented by A. Gomez, L. Sigrist et al in [13]. A short overview over the operation of the EMU is given in this section, since the EMU is also integrated in the PCB of the final prototype.
The heart of the EMU is a BQ25505 energy harvesting chip from Texas Instruments. This chips features a built-in boost converter that brings the input voltage of the source to a higher level for accumulating the harvested energy in a capacitor. The input impedance of the chip is continuously adjusted to track the optimal power point of the source. The load is supplied with the desired voltage by a buck converter (TPS62740) with digitally adjustable output voltage. Hence, the power points of the source and the load are completely decoupled.
The EMU is designed for the use in transiently powered systems. The generation of energy bursts is implemented as follows. The load (in this case the IPU) needs to configure the size of the next energy burst, as well as the current optimal output voltage level. The output voltage level can directly be adjusted digitally by the load using the interface at the buck converter. Between two energy bursts, the output voltage is kept at the minimal operating voltage of the load MCU in deep sleep. If the MCU is woken up to execute the next task, the minimal voltage level for the current task is set by the MCU. The EMU needs to know the size of the next energy burst for triggering the wake-up of the load MCU and therefore the execution of the next task. For that, the *battery OK* signal of the energy harvesting chip (BQ25505) is used. It changes to high as soon as a configurable voltage threshold at the energy buffer is reached. The load MCU can configure this voltage threshold according to the requested size of the next energy burst. The *battery OK* signal of the energy harvesting chip is then used to wake up the load MCU by a GPIO interrupt. The voltage level V_{cap} at the energy buffer capacitor must stay within 2 V and 5.5 V, limited by the energy harvesting chip and the allowed input voltage for the buck converter. The current amount of stored energy E_{cap} can easily be calculated using the formula for the stored energy in a capacitor of size C:

$$E_{cap} = \frac{1}{2} C V_{cap}^2 \tag{3.2}$$

Using this formula, the size of the energy buffer capacitor is determined according to the burst with the highest energy requirement.

3.3 Task Implementation for Transiently Powered System

The vision sensor shall be powered transiently, driven by energy bursts from the EMU. Between two bursts, all peripherals must be shut down and the microcontroller must go into the deepest sleep mode to minimize the losses between energy bursts. The Stonyman vision chip supports a software shut-down and can be connected directly to the supply voltage. The FRAM chip and the microSD Card are connected to the supply voltage via a load switch (TPS22960) controlled by GPIO pins of the MSP432. This allows the MCU to activate the memories only if they are needed.
The MSP432 is optimized for low-power applications and features various low-power modes (LPMs) [19]. In the deepest low-power mode (LPM4.5), all peripherals including the CPU and the SRAM bank are disabled. The device can only be woken up by toggling a appropriately configured GPIO pin or using the reset pin. The current consumption in LPM4.5 is only 25 nA [18]. Therefore, this LPM is ideal for the use in a transiently powered system. Because also the volatile internal SRAM banks are disabled in LPM4.5, the system state has to be restored after wakeup. Using the transiently powered system with a NVMH, the system state is stored in the FRAM chip. Appropriate firmware functions are developed for automatically storing the necessary data depending on the executed tasks in the FRAM before entering the LPM and restoring the data after wakeup. The firmware is organized as a state machine whose operation depends on the configured schedule of the application.
For waking the MSP432 up when it is in LPM4.5, the EMU toggles a *battery OK* signal, causing a GPIO interrupt on the MCU. Four digital lines are used for configuring the buck converter in the EMU. Voltages from 1.8 V to 3.3 V in steps of 100 mV can be requested. Two additional digital lines are used to configure the next burst size. The EMU uses this information for setting the energy threshold for sending the *battery OK* signal. All these six digital lines are driven by GPIO pins of the MSP432.

3.4 MSP432 Demoboard

For the development of systems based on the MSP432 microcontroller unit, Texas Instruments provides a so-called MSP432 LaunchPad. The launchpad is a PCB, featuring an MSP432 MCU with the necessary external components and connection headers for a simple access to all necessary pins of the MCU. Various BoosterPacks can be plugged onto the board for the evaluation of peripherals. The launchpad has also a built-in debug board that allows to program the MSP432 over USB using the provided software on the computer.
The MSP432 LaunchPad makes it possible to easily attach peripheral devices with wires. However, plugging many different peripherals onto the launchpad is impractical, especially when testing a portable application and if there are noise-sensitive analog signals involved. For evaluating the different peripherals used during this project, a custom MSP432 Demoboard was designed. With different on-board memories and a connection header for the Stonyman vision chip, the demoboard is specified for the evaluation of portable vision sensor applications.

Features

The heart of the demoboard is a MSP432P401R microcontroller unit with the necessary external components. The demoboard contains a connector for the Stonyman vision

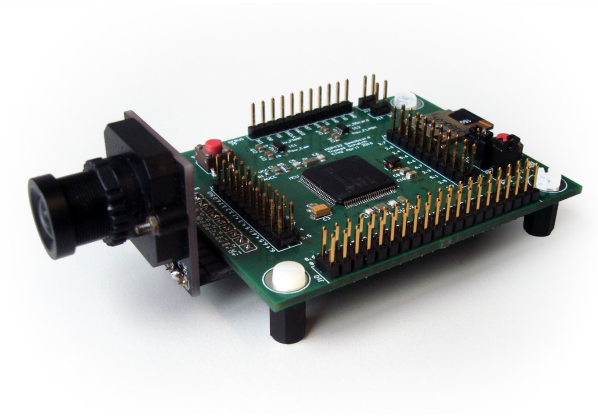


Figure 3.5: Image of the custom MSP432 Demoboard with attached Stonyman vision chip.

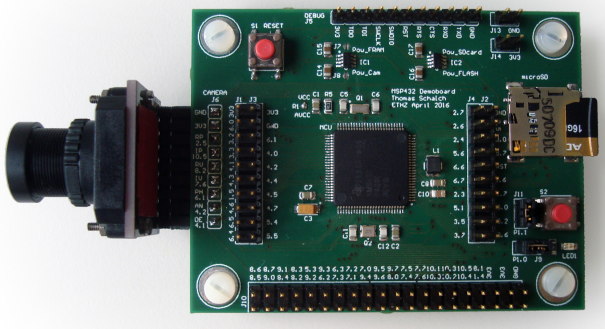


Figure 3.6: Image of the custom MSP432 Demoboard with attached Stonyman vision sensor and inserted microSD Card.

chip and the BoosterPack connectors as available on the MSP432 LaunchPad, such that the BoosterPacks can also be used with the demoboard. It further features a microSD Card slot, an FRAM chip and a Flash memory chip. Since the two memory chips have exactly the same footprint and pin-out, one can easily replace the chips with a chip of another technology or storage capacity. This simplifies the evaluation of different storage technologies. Four controllable load switches allow switching the power for the camera, the microSD Card and the two memory chips. Furthermore an LED and a user button can be used for debugging purposes. Figure 3.7 shows, how the peripheral devices are connected to the MCU. All pins of the MCU are also accessible through connection headers, which facilitates the debugging and allows to attach further peripherals to the demoboard. A separate debugging connector is exposed for programming and debugging the MCU through the Serial Wire Debug (SWD) technology or JTAG. The easiest way to program the demoboard is to attach it to the built-in debugger of the MSP432 LaunchPad. The demoboard then behaves like a LaunchPad and the same tools can be used on the PC.

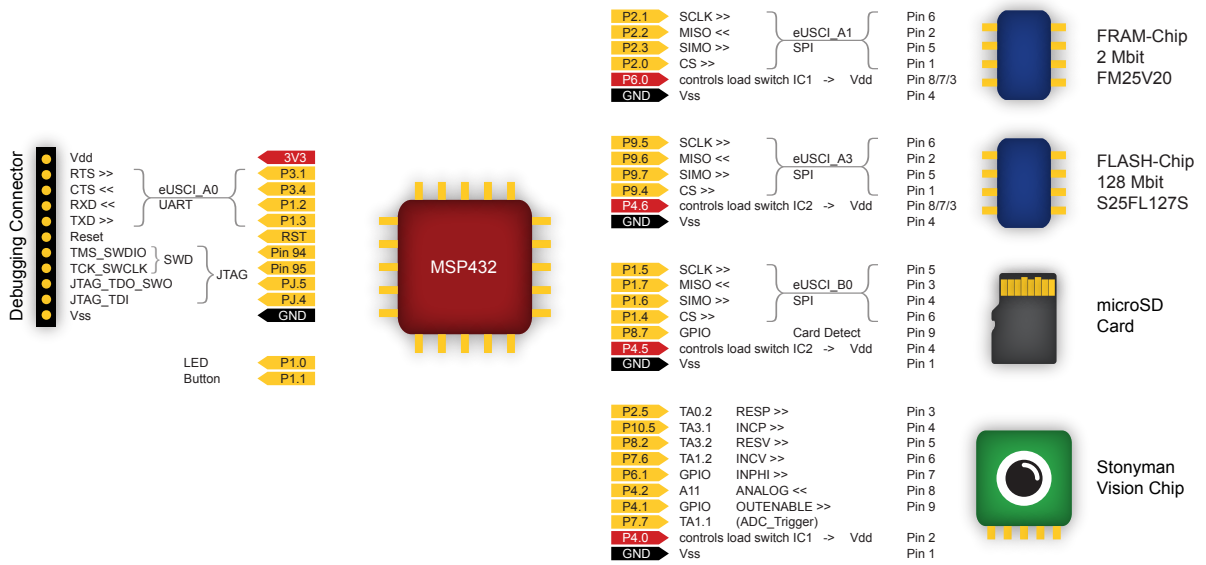


Figure 3.7: The schema of the MSP432 Demoboard shows how the peripherals are connected to the MCU.

Printed Circuit Board (PCB) Design

The printed circuit board (PCB) for the MSP432 Demoboard is designed in the PCB Design Course at ETHZ using Altium Designer. The schematics base on the schematics of the MSP432 LaunchPad, complemented with the additional peripheral components. For the resistors and capacitors, the package size 0805 (i.e. 2mm×1.25mm) is chosen, since those components can easily be soldered by hand but are still small enough to save space. Two dual load switches TPS22960 from *Texas Instruments* are used to switch the power supply to the two memory chips, the microSD Card and the camera. The PCB consists of four copper layers, of which two are used as signal layers and two as power supply planes. The final size of the board is 5.6 × 7.3 cm.

3.5 Vision Sensor Prototype

A wearable prototype of the velocity estimation sensor is designed, that is optimized for a small form factor. The prototype consists of an Image Processing Unit (IPU) board, an Energy Management Unit (EMU) board for powering the IPU transiently and an additional battery board for powering the IPU continuously with a battery pack. The IPU can either be connected to the EMU for transiently powered operation or the battery board for continuous operation. The PCBs of all boards consist of four copper layers, of which two are used as signal layers and two as power planes. The PCBs are designed in Altium Designer. Figure 3.8 and 3.11 show the assembled vision sensor prototype consisting of the IPU, the EMU and the attached Stonyman vision chip. Figure 3.10 shows an overview over all three boards of the prototype.

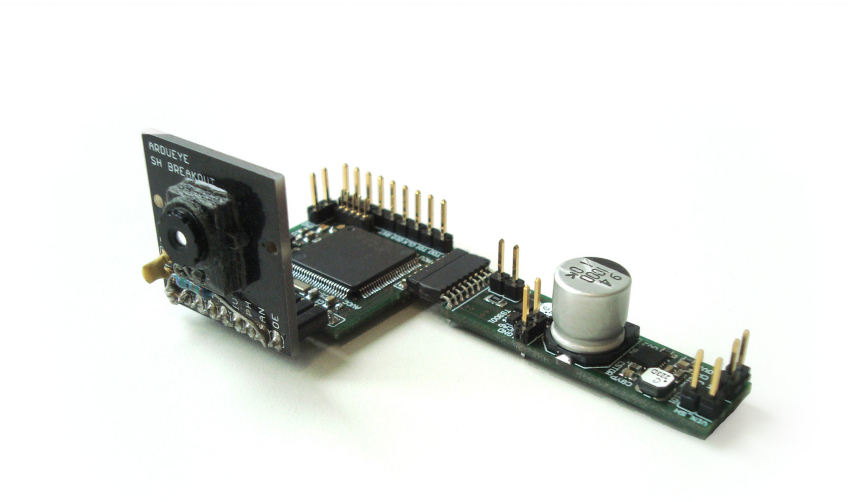


Figure 3.8: Image of the vision sensor prototype consisting of the Image Processing Unit (IPU) with attached Stonyman vision chip and Energy Management Unit (EMU).

Image Processing Unit (IPU) Board

The IPU board has similar features as the MSP432 demoboard, but its size is significantly reduced and all redundant components and connection headers are removed. The IPU contains a MSP432P401R microcontroller unit with the necessary components, an FRAM chip, a microSD Card slot, a connection header for the Stonyman vision chip, a connection header for plugging in the EMU board or the battery board and some connection headers for programming and debugging. The organization of all peripherals is shown in Figure 3.9. The size of the IPU board is 2.6×2.8 cm.

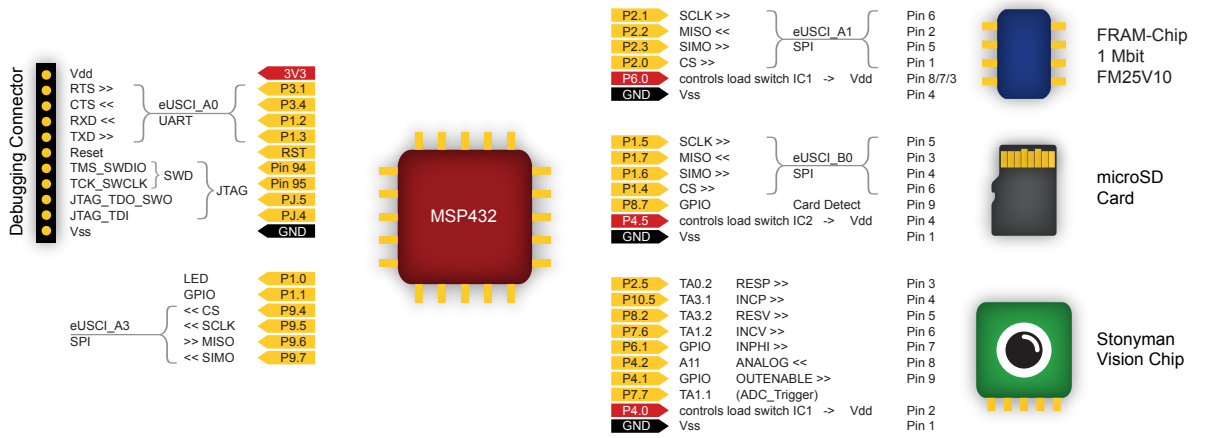


Figure 3.9: The schema of the IPU of the vision sensor prototype shows how the peripherals are connected to the MCU.

Energy Management Unit (EMU) Board

The EMU board is designed according to the description in Chapter 3.2. Two shunt resistances at the input and output of the EMU are integrated in the board, such that all interesting currents can be determined. The board contains headers for connecting an energy harvesting source (e.g. a solar panel) and for easily measuring all necessary voltages. It can be plugged to the IPU over a connector. The size of the EMU board is 4.2×1.2 cm.

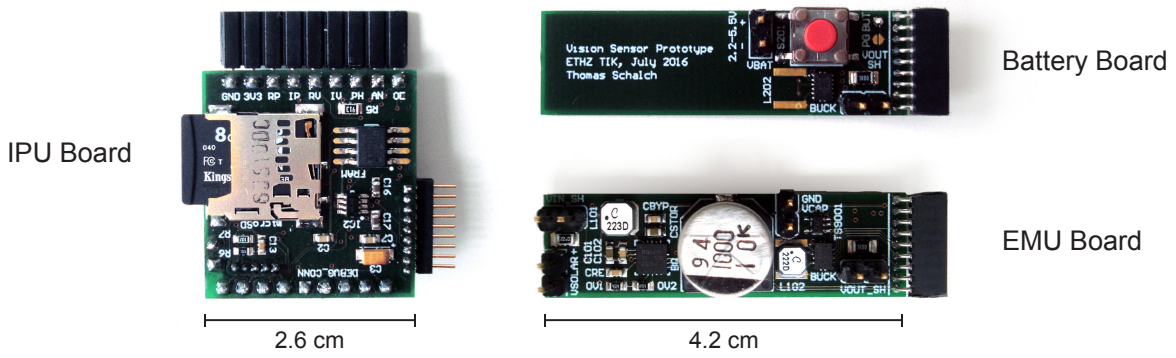


Figure 3.10: All three PCBs of the vision sensor prototype.

Battery Board

The battery board has the same size as the EMU board and can be used for debugging purposes or for powering the IPU continuously using a battery. The battery board features the same buck converter as on the EMU. If plugged to the IPU, the MCU can choose the supply voltage over the same interface as with the EMU. The battery board features a user button that is connected to the wakeup signal on the MCU. This allows a manual triggering of energy bursts for debugging purposes when testing the transiently powered system operation.

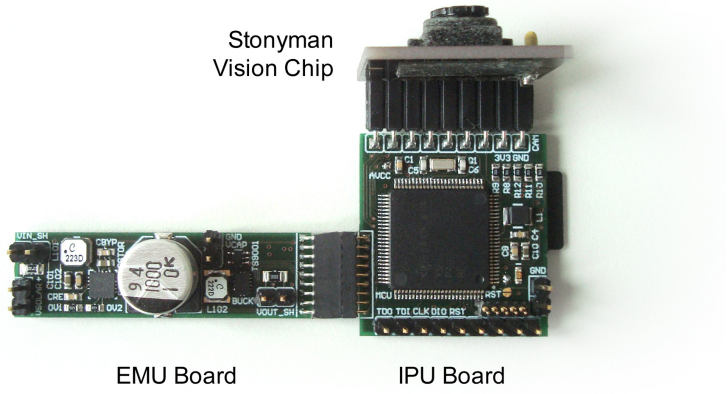


Figure 3.11: Image of the vision sensor prototype consisting of the Image Processing Unit (IPU) with attached Stonyman vision chip (top) and Energy Management Unit (left).

Chapter 4

Motion Estimation

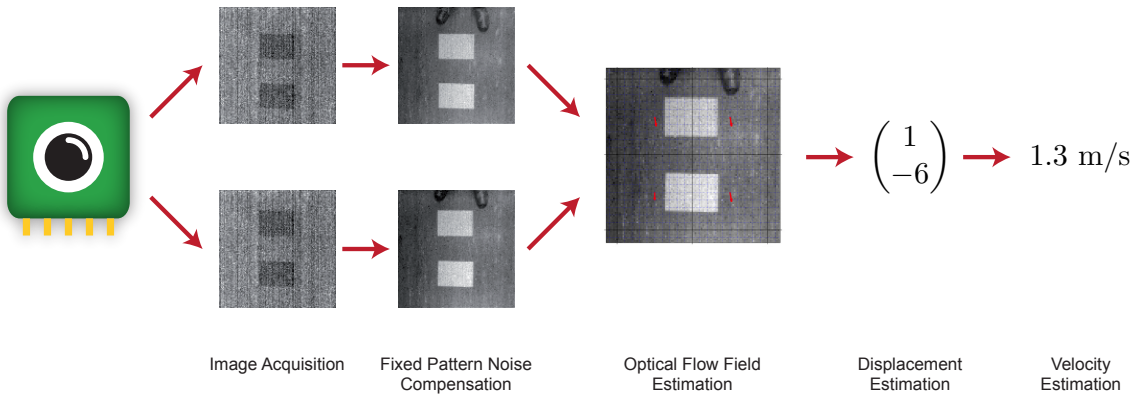


Figure 4.1: Function of the visual velocity estimation sensor.

In Chapter 2, the generic function of the transiently powered vision sensor is described, whereas in Chapter 3, the hardware implementation is discussed in detail. This chapter introduces different techniques for visual motion estimation and describes the final implementation of the velocity estimation algorithm. Figure 4.1 shows a detailed overview over the visual velocity estimation application. After acquisition of the two images, the fixed pattern noise (FPN) has to be removed (see Chapter 3.1.2). The two images are then processed for detecting the motions within the two images using a so-called optical flow algorithm. This motion estimation technique is introduced in Chapter 4.1. Using the final algorithm implementation (Chapter 4.3), the displacement between the two images and therefore the velocity can be estimated.

4.1 Optical Flow

Optical flow is a fundamental concept in visual perception firstly described by Gibson in 1950 [12]. The optical flow describes the apparent velocities of movements of brightness patterns in the perceived image [16]. Those movements of brightness patterns may be caused by relative motion between the captured objects and the observer or by motion of single objects within the scenery. Gibson formulated the concept of optical flow by considering human visual perception [12], but it is nowadays a fundamental concept in image processing and computer vision. Expressing the optical flow in an image sequence

in an appropriate mathematical way can be a very useful tool for many applications like flow measurements of fluids [5], motion segmentation [9] or video compression [4]. In this project, the concept of optical flow is used to extract information about the movement of the observer while capturing a static scenery.
This section firstly introduces a common mathematical interpretation of the optical flow and then classifies and evaluates different approaches for estimating the optical flow in an image sequence. Because there is a large variety of different approaches to estimate the optical flow, only algorithms that are interesting for this project are considered. Unless stated otherwise, the overview is based on the paper *The Computation of Optical Flow* from Beauchemin and Barron [5].

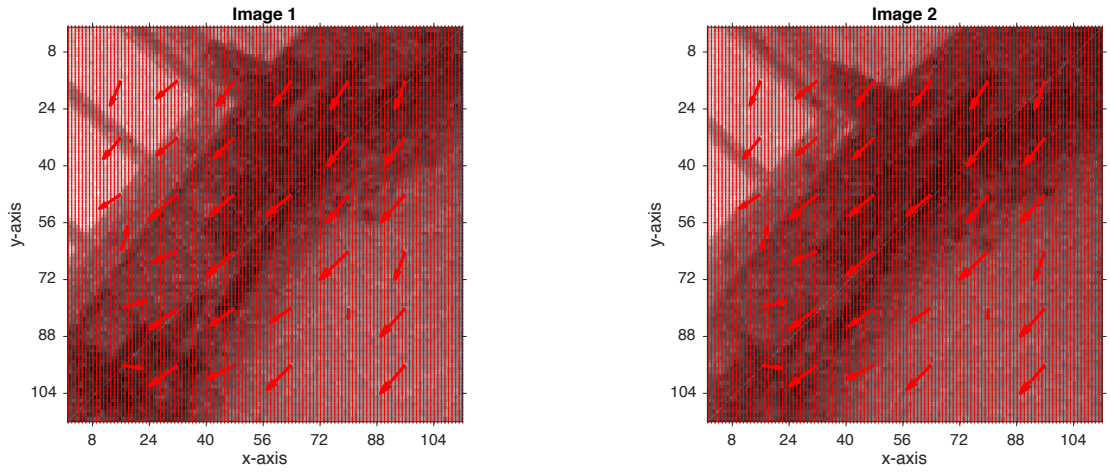


Figure 4.2: Optical flow between image 1 and image 2, mathematically expressed as a vector field (red) and estimated using the block-matching algorithm (see Section 4.1.3) with a blocksize of 16×16 pixels.

4.1.1 Mathematical Interpretation of the Optical Flow

For a sequence of images, where $I(x, y, t)$ describes the brightness of the pixel at position (x, y) at time t, the optical flow specifies how much each pixel of the image moves between subsequent images [25]. The computation of the optical flow is based on the hypothesis that within the time Δt, the pixel at position (x, y) moves to position $(x + \Delta x, y + \Delta y)$ without a change in intensity [5]. Expressed as an equation, this hypothesis means:

$$I(x, y, t) \approx I(x + \Delta x, y + \Delta y, t + \Delta t) \tag{4.1}$$

The right-hand side of equation (4.1) can be expanded as Taylor series:

$$I(x + \Delta x, y + \Delta y, t + \Delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + ... \tag{4.2}$$

Equations (4.1) and (4.2) can be combined and for small displacements Δx, Δy and a small time difference Δt, the higher-order terms of the Taylor series can be ignored:

$$\frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t = 0 \tag{4.3}$$

Dividing equation (4.3) by Δt and defining the velocities $v_x := \frac{\Delta x}{\Delta t}$ and $v_y := \frac{\Delta y}{\Delta t}$ gives:

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t} = \frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0 \tag{4.4}$$

Equation 4.4 is called **optical flow constraint equation**. It only defines a single constraint for two variables (v_x and v_y). This problem is known as the **aperture problem**. One consequence of this problem is, that if there is no brightness structure along one coordinate of the image (or the considered part of the image), the velocity along this coordinate can not be recovered.

4.1.2 Differential Methods

In differential methods, the spatial and temporal intensity derivatives ($\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$) are estimated from the image intensities and then directly used to compute the optical flow in an image by solving the optical flow constraint equation (Eq. 4.4) for each pixel. The result is a velocity vector (v_x, v_y) for each pixel. Because there exists only one constraint equation for two velocity components, additional constraints need to be defined.

Horn-Schunck Method

The Horn-Schunck method [16] assumes that neighbouring points of the same objects have similar velocities and the velocity field therefore varies smoothly over the whole image. This can be expressed in the additional mathematical constraint of limiting the squared difference between the velocity at one specific point and the average velocity over a neighbourhood around that point. Minimizing both the error of the optical flow constraint equation (Eq. 4.4) and the additional constraint, gives an estimate for the velocity vector (v_x, v_y) for each pixel. But analytically solving this optimization problem over the whole image with two equations for every pixel would be very costly. Therefore usually iterative methods are used. The idea is to make an estimate for the velocity vector of a certain pixel based on the previous estimation of the velocity vectors of this and his neighbouring pixels.
One advantage of the Horn-Schunck method is its robustness to additive noise in the image intensities, since the error of the optical flow constraint equation is also minimized with an adjustable weight. The Horn-Schunck method uses considerable computational effort to provide a good, smooth estimate for the whole optical flow field in an image. However, if a smooth optical flow field is not of relevance for an application, a method using less computational effort can be preferred.

Lucas-Kanade Method

The Lucas-Kanade method [21] defines local constraints around a small neighbourhood around the considered pixel. Namely, it assumes that the velocity is constant within a certain small window around the pixel. This results in one equation for each pixel (x_i, y_i) within the window:

$$\left.\frac{\partial I}{\partial x}\right|_{(x_i,y_i)} v_x + \left.\frac{\partial I}{\partial y}\right|_{(x_i,y_i)} v_y = -\left.\frac{\partial I}{\partial t}\right|_{(x_i,y_i)} \tag{4.5}$$

The squared error of this set of equations can be minimized using the method of least squares:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum_i \left(\frac{\partial I}{\partial x}\Big|_{(x_i,y_i)} \right)^2 & \sum_i \frac{\partial I}{\partial x}\Big|_{(x_i,y_i)} \cdot \frac{\partial I}{\partial y}\Big|_{(x_i,y_i)} \\ \sum_i \frac{\partial I}{\partial x}\Big|_{(x_i,y_i)} \cdot \frac{\partial I}{\partial y}\Big|_{(x_i,y_i)} & \sum_i \left(\frac{\partial I}{\partial y}\Big|_{(x_i,y_i)} \right)^2 \end{bmatrix}^{-1} \cdot \begin{bmatrix} \sum_i \frac{\partial I}{\partial x}\Big|_{(x_i,y_i)} \cdot \frac{\partial I}{\partial t}\Big|_{(x_i,y_i)} \\ \sum_i \frac{\partial I}{\partial y}\Big|_{(x_i,y_i)} \cdot \frac{\partial I}{\partial t}\Big|_{(x_i,y_i)} \end{bmatrix} \tag{4.6}$$

The index i runs over all pixel within the considered window. Note that the spatial and temporal gradients need only be computed once for all pixels. The computation of the matrix entries are then simple multiplications and additions. The velocities v_x and v_y for the pixel in the center of the considered window result then from calculating the inverse of a 2×2 matrix. So the implementation of this algorithm on a microcontroller is straightforward. However, the computational effort for calculating the velocity for each pixel is considerable, since a least squares problem needs to be posed and solved for each pixel. Furthermore, storing two subsequent images plus their spacial and temporal derivatives makes high demands on the memory capacity.
This method can be improved by using a weighted window where the pixels in the center of the window get a higher weight. This results in a weighted least squares problem.

Feature Selection

One drawback of local constraints like in the Lucas-Kanade method is, that the movement in areas with uniform intensities can not be recovered. It therefore makes sense to select a certain subset of pixels (e.g. corners) for which the estimation of the optical flow seems promising, instead of considering all pixels of the image. This is called *feature selection*.

Pyramidal Approach

The differential methods rely on the assumptions made during the derivation of the optical flow constraint equation (Eq. 4.4). Namely, the displacements Δx, Δy and the time difference Δt need to be small. Often, only displacements in the order of one pixel can be recovered. It is obvious that this makes high demands on the image sampling rate. To relax those limitations, the resolution of the images can be reduced. Once a coarse estimate for the velocities is established, the estimate can be iteratively refined using an image with a higher resolution until the maximal resolution is reached. Such a pyramidal implementation of the Lucas-Kanade method is proposed by Bouguet [6]. The main drawback of this method is, that small objects in the images are lost, because they can not be recognized in the low-resolution images. This method is therefore not appropriate if the resolution of the camera is poor anyway, or if only fine structures in the images are expected. Furthermore, poor estimates for coarse images propagate through all hierarchical levels and result in a poor estimate at the end.

4.1.3 Block-based Methods

A heuristic approach of estimating the motion between two images is to compare the images for all possible displacements and search for the displacement, where the images "fit best". Such a method only works, if there is a pure translation and only a negligibly small rotation between the images. To relax this condition, one could think of dividing the image into blocks and then search for each block the position in the next image where the block "fits best". The smaller the block size, the more robust the method is to rotations.

The result of the algorithm is an independent translation vector for each block, which represents an estimate for the optical flow of the pixels within the considered block. Block-based methods are not restricted to small displacements like the differential methods and are therefore able to recognize displacements of many pixels.

Correlation-based Methods

The correlation is one possible figure of merit for quantifying how well one block fits at a certain search position. In signal processing, the correlation between two functions $f(x)$ and $g(x)$ is defined as

$$R_{fg}(\tau) = \int_D f(x+\tau)g(x)dx \tag{4.7}$$

From a heuristic point of view, the correlation describes how well the two functions f and g fit each other under the displacement τ. For example, if f and g are identical functions, but displaced by two units, then the correlation function $R_{fg}(\tau)$ will have a maximum at $\tau = 2$ (where the maximal value is related to the energy of the function). For estimating the displacement of blocks between two subsequent images, one can calculate the two-dimensional correlation function. Its maximum value will be an estimate for the displacement of the block between the two images.
As the computation of the correlation between two large functions is computationally expensive, it is often beneficial to calculate the correlation in the frequency domain, where convolutions are transformed into multiplications. This is done by the *phase correlation method* [28], [10].

Block-Matching Algorithms

Instead of maximizing the correlation function, it is conceptually equivalent to minimize a certain cost function. This is done in block-matching algorithms. The displacement with the lowest cost function value is supposed to be the best displacement estimate. Possible cost functions are the sum of absolute differences (SAD) or the sum of squared differences (SSD) [4]:

$$SAD(\Delta x, \Delta y) = \sum_i |I(x_i, y_i, t_1) - I(x_i + \Delta x, y_i + \Delta y, t_2)| \tag{4.8}$$

$$SSD(\Delta x, \Delta y) = \sum_i \left(I(x_i, y_i, t_1) - I(x_i + \Delta x, y_i + \Delta y, t_2)\right)^2 \tag{4.9}$$

The index i runs over all pixels for which the cost function is evaluated, Δx and Δy are the image displacements in x- and y-direction and $I(x, y, t)$ is the intensity of the pixel at location (x, y) at time instant t. Minimizing the cost function over all possible Δx and Δy gives an estimate for the displacements Δx and Δy. Calculating the SSD is more expensive than calculating the SAD, as an additional multiplication has to be performed. Block-matching algorithms are widely used in video compression. A publication from Barjatya [4] provides a good overview over different block-matching algorithms and their use for video compression. Essential parameters of the block-matching algorithms are the block size and the size of the search area for which the cost function is evaluated.
The computational effort of the block-matching algorithm depends on the number of cost function evaluations, which is proportional to the number of pixels within the image and

the size of the search area. The number of pixels is given by the resolution of the camera. The search area should therefore be chosen as small as possible. However, the size of the search area is the size of displacements that need to be detected and is therefore also usually specified. The block size has no impact on the computational effort. A small block size is more robust to rotations between the two images, whereas a large block size is more robust against noise in the images.

4.2 Related Works

In recent years, there was a considerable research activity in the implementation of motion estimation algorithms in small, low-power systems. Often, the systems are used for optical navigation of unmanned aerial vehicles (UAV) or robots. A very interesting contribution originates from Dominik Honegger et al. Their PX4FLOW optical flow sensor provides velocity and position estimation at high update rates for mobile robot navigation [15]. The sensor system is based on an ARM Cortex-M4 architecture, like this project and the optical flow field is also calculated using a block-matching algorithm. The system not only works vision based, but also includes a gyroscope and an ultrasonic sensor. The power consumption of this system is specified as 575 mW. In our project, the power consumption for the hardware proposed in Chapter 3 is by orders lower, at the cost of a reduced update rate and accuracy of the final vision sensor. Dominik Honegger et al. also proposed an other visual optical flow sensor system for robot navigation based on an FPGA [14].
J. D. Jackson et al. built a low-cost optical navigation device based on a commercial optical mouse sensor [20]. Those mouse sensors work properly in outdoor conditions, if they are supplied with sufficient light, but they can usually only be used indoors together with a bright light source. V. More et al. proposed a visual navigation system for UAVs based on optical flow estimation using the Lucas-Kanade method and an ultrasonic sensor [23]. To account for the high computational effort for the Lucas-Kanade method, the system runs on a on-board linux computer (Odroid-U3). Computing the optical flow with the Lucas-Kanade method was performed on an Atmel ATmega2560 microcontroller by K. Schneider et al. [26]. They used the same Stonyman vision chip from Centeye as used in our project. However, the used microcontroller can only handle camera resolutions up to 28×28 pixels due to the high memory requirement of the Lucas-Kanade method.

4.3 Visual Velocity Estimation on the MSP432

Final Choice of the Motion Estimation Algorithm

The differential methods in Chapter 4.1.2 can only recover displacements in the order of one pixel. Since the maximal frame rate with the given hardware is 37.5 fps, displacements of many pixels between the two images are expected for walking speed. The proposed pyramidal approaches for differential methods are not satisfactory in this case because fine brightness structures in the low-resolution images can not be detected, but this is essential for a good performance of the walking speed estimation application. In contrast, the search area in the block-matching algorithm can be chosen arbitrarily and displacements of many pixels can be recovered. However, the block-matching algorithm has only a resolution of one pixel, whereas differential methods feature sub-pixel accuracy. In this project, the reliability of the algorithm has more importance than a fine accuracy.

Furthermore, the computational effort is huge for differential methods like the Lucas-Kanade method, since for each pixel a weighted least squares problem has to be posed and solved. The Lucas-Kanade method also has a high memory requirement, since beside the two images, three gradient fields of the same size as an image need to be stored during processing. Correlation-based methods that are most efficiently computed in the frequency domain are clearly not suited for implementation on a small MCU because of the immense computational effort. The computational effort of the block-matching algorithm is much lower and can be optimized by choosing a cost function whose evaluation is fast on the given architecture. The block-matching algorithm has almost no additional memory requirements during processing, since the cost function evaluation works directly with the brightness values of each pixel.
For these reasons, the block-matching algorithm is the most appropriate algorithm for implementation on the given hardware as proposed in Chapter 3.

Implementation of the Block-Matching Algorithm on the MSP432

The block-matching algorithm is first implemented in MATLAB and fed with previously acquired image datasets from the Stonyman vision chip to verify the proper operation for the velocity estimation application and to find optimal values for the block size (see evaluation in Chapter 4.4). The block-matching algorithm is then implemented in C for use on the MSP432.
The ARM Cortex-M4 architecture allows some very useful optimizations of the block-matching algorithm. The dedicated integer vector instructions *USAD8* (unsigned sum of absolute differences) and *USADA8* (unsigned sum of absolute differences and accumulate) can be used for calculating the sum of absolute differences (SAD) of four 8-bit values simultaneously. The cost function of the block-matching algorithm can therefore be evaluated for four pixels within one instruction cycle. Together with further optimizations, the execution time of the block-matching algorithm is reduced from around 2s for the initial implementation to 56ms in the final implementation, using the same parameters (CPU frequency: 48 MHz, search area: ±8px in y-direction and ±3px in x-direction).

Velocity Estimation

The block-matching algorithm computes an optical flow field. This field has to be reduced to one final displacement estimation vector. The Stonyman vision chip has strong fixed pattern noise (FPN, see Chapter 3.1.2), which is compensated after acquisition. However, the FPN can not be completely removed. The remaining weak FPN acts like a bias to zero for the block-matching algorithm. If one block contains no brightness structure within the whole search area, the block-matching algorithm will likely output the zero vector as displacement estimation for this block, because the FPN is identical in both images and will produce the minimal cost function for zero displacement. It therefore makes sense to assume that a block whose displacement is estimated to be zero, only contains noise, unless all blocks lead to a zero displacement estimation. For this application, the displacement is therefore estimated using the following two rules:

1. If all optical flow vectors are zero, the displacement estimation is zero.

2. If at least one vector is non-zero, the displacement estimation is the average over all non-zero vectors.

The final displacement estimation vector has the units of pixels. It can be scaled by a constant factor to get a velocity estimation. This factor depends on the characteristics of the lens, the height of the camera above ground and the time difference between two images. Since this factor depends on the application, it is not included in the firmware and can be added later on the computer, if needed.

4.4 Calibration of the Motion Estimation Algorithm

The two essential parameters of the block-matching algorithm are the size of the search area and the block size, as discussed in Chapter 4.1.3. The size of the search area has an impact on the computational effort and should be chosen according to the maximal displacements that the algorithm needs to recover. On the other hand, the block size has no impact on the computational effort and can therefore be freely chosen for optimal performance. Larger block sizes are more robust to noise in the image. On the other hand, smaller block sizes are more robust to rotations. The optimal choice of the block size therefore depends on many application-specific factors. The goal of this section is to find an optimal choice for the block size and evaluate the performance of the block-matching algorithm. For doing so, the position of the vision sensor is tracked in this experiment by integrating the velocity estimations over time. Such a visual position estimation method is called **Visual Odometry** [24]. This evaluation method is quite restrictive, as every wrong velocity estimation causes an erroneous offset for all further position estimations. For a poor velocity estimation algorithm, the estimated position can already completely drift away after a few estimations. However, being able to track the position of the vision sensor accurately is a sign for a robust velocity estimation sensor.

Data Set Acquisition

For a fair comparison of the different settings of the block-matching algorithm, all configurations need to be evaluated using the same image sequence as input dataset. Therefore, an image dataset is acquired for this experiment and is then fed into a MATLAB model of the different configurations of the block-matching algorithm. For a given image sequence, the MATLAB model performs exactly the same calculations as the motion estimation algorithm in the MSP432 implementation. The image dataset is acquired using the battery-powered vision sensor and a firmware that is optimized for speed, such that a frame rate of approximately 20 fps is achieved. However, the frame rate is not exactly constant, as the time for writing one image onto the microSD card is not predictable and can vary by some microseconds, leading to slight variations in the frame rate. In this evaluation, the displacements between two frames can directly be integrated to get a position estimate. The time difference between two frames does therefore not need to be known and slight variations in the frame rate have no influence on the final result.

For acquiring the image dataset, the MSP432 demoboard is attached to a trolley to guarantee a constant height of the lens above ground (62 cm). The trolley is then moved back and forth along the x-coordinate axis. White sheets of paper are put on the ground as position markers every 0.5 m. The acquired dataset has a length of 3024 frames and covers a travelled distance of 42 m. Figure 4.3 shows an excerpt of six frames of the dataset.

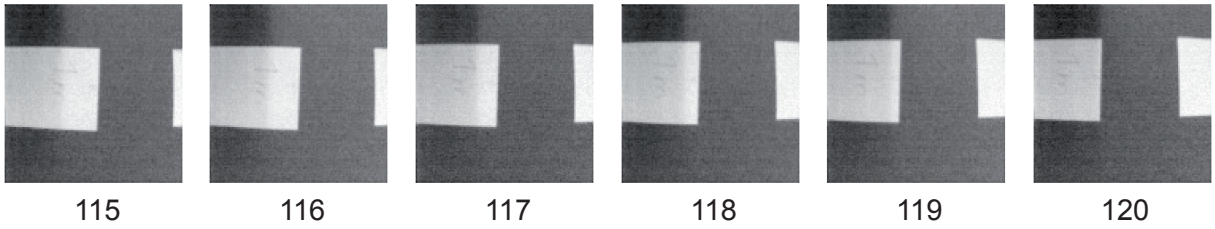


Figure 4.3: Frames 115 to 120 from the image dataset (total length: 3024 frames) used for the motion estimation evaluations.

Finding Optimal Block Size for Block-Matching Algorithm

To find an optimal value for the block size, the acquired image data set is fed to the MATLAB model of the block-matching algorithm using different block sizes between 96 pixels (using the whole image as one single block) and 16 pixels (dividing the whole image into 36 blocks). The size of the search area is set to ±8px in x- and y-direction. The final displacement estimation is the average over all non-zero optical flow vectors. To convert the unit of the displacement vector from pixels into meters, a scale factor is used that is experimentally determined by capturing an image of an object with well-known dimensions.
Figure 4.4 shows the estimated position of the vision sensor after each frame using the MATLAB implementation of the block-matching algorithm for different block sizes. The green crosses indicate the actual position of the sensor device. Those reference positions are labelled by hand by inspection of the position markers within the images.

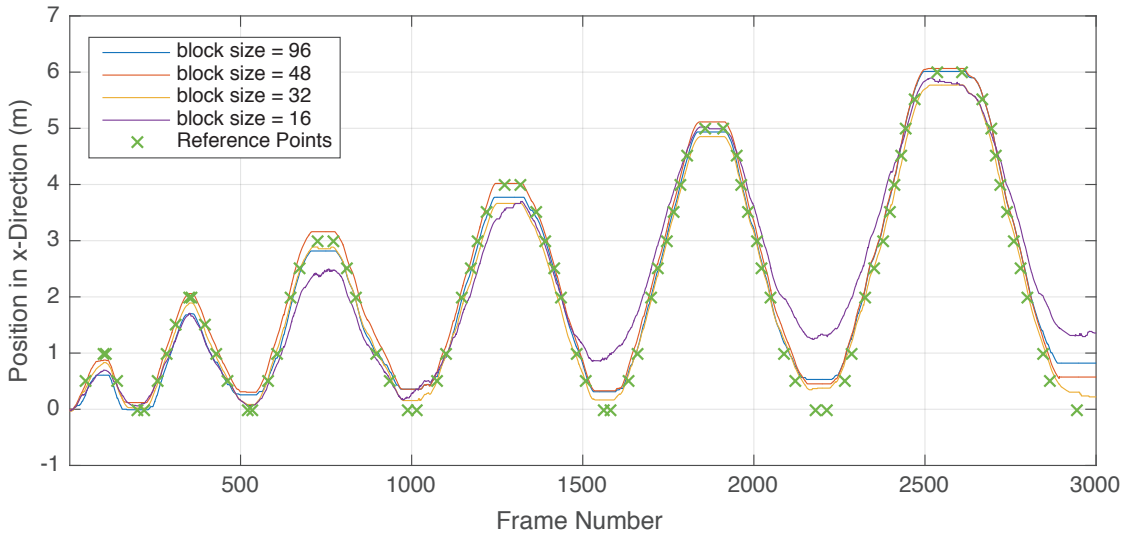


Figure 4.4: Estimated position of the sensor device in x-direction using different block sizes for the block-matching algorithm, simulated with the MATLAB model using real-world input data.

The results in Figure 4.4 show that in general, the block-matching algorithm manages to track the actual position of the sensor device quite well. However, there are some differences in accuracy for the different block sizes. The smallest block size of 16px (violet curve) shows the worst behaviour, since it can not always follow the reference positions and the curve shows a certain peaky behaviour. This can be explained by the lower robustness

of the algorithm to the noise in the images for lower block sizes. The three other block sizes show a similarly good behaviour in this plot. One can also observe, that the performance of the block-matching algorithm is worst near the turning points. The trolley is moved very slowly near the turning points such that the desired position of the trolley is exactly met. Therefore, the movements within the last centimeters before a turning point are spread over many frames. So the displacement between two frames is for those frames usually below one pixel. Because the block-matching algorithm only has a resolution of one pixel, such slow movements can not be tracked. However, this effect has less relevance for the final velocity estimation application, because even slow walking speeds are fast enough to cause displacements of many pixels within distinct frames.

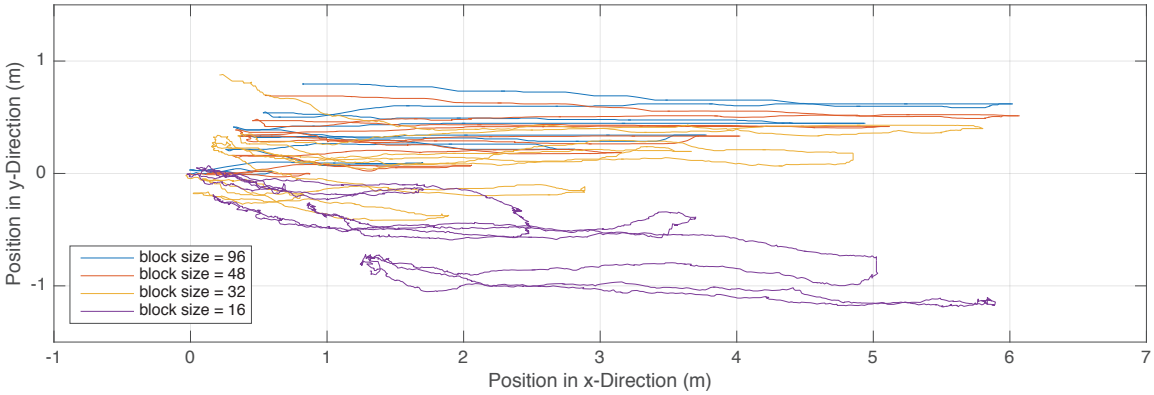


Figure 4.5: Estimated position of the sensor device in the x-y-plane parametrized by the frame number using different block sizes for the block-matching algorithm, simulated with the MATLAB model using real-world input data.

For a further inspection, the estimated position is shown in the x-y-plane in Figure 4.5. In this plot, it is even more obvious that the algorithm with a block size of 16px shows locally random behaviour, which can not be explained by the smooth translations of the trolley during the dataset acquisition. The algorithm with large block sizes of 96px and 48px show the best tracking behaviour of the translations of the trolley. Because using only one large block filling the whole image (block size = 96px) is very sensitive to rotations between two subsequent frames, this large block size should be avoided. A block size of 48px already shows a reduced sensitivity against rotations. Furthermore, the block size of 48px shows a good tracking behaviour in x-direction. After a travelled distance of 42 m, the estimated position deviates 57 cm from the reference value, which corresponds to only 1.4% of the travelled distance. The evaluation thus shows, that a block size of 48px for the block-matching algorithm is most reasonable for this application.

Chapter 5

Evaluation

The evaluation of the vision sensor is organized as follows: First, the energy consumption and the execution time of all tasks are measured (Chapter 5.1). Those measurements are used to optimize the tasks for lowest energy consumption and for configuring the Energy Management Unit. Next, the performance of the vision sensor as a transiently powered system is evaluated to investigate the benefits of using a non-volatile memory hierarchy (Chapter 5.2). Finally, the wearable velocity estimation sensor is evaluated in a real-world experiment (Chapter 5.3).

5.1 Task Characterization and System Optimization

In the task characterization, the energy consumption and the execution time of each task is determined experimentally. According to Chapter 2.4, the tasks of the visual velocity estimation sensor are: *image acquisition*, *processing* (motion estimation) and *storage* (see Figure 5.1). Each task has many variable parameters like the supply voltage, the CPU frequency of the MCU and further task-dependent parameters. The task characterization helps finding the optimal parameters to achieve a minimal energy consumption for each task. Furthermore, the energy consumptions of the tasks need to be known for configuring the Energy Management Unit (see Chapter 3.2).

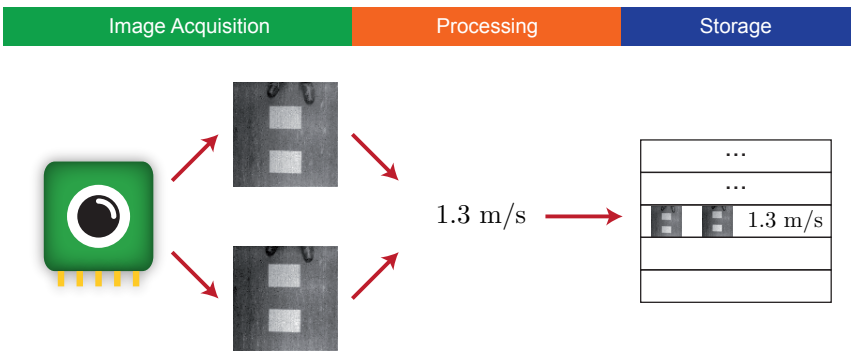


Figure 5.1: Tasks of the visual velocity estimation sensor.

Experimental Set-Up

All measurements are conducted using the IPU board of the final vision sensor prototype with attached Stonyman vision chip (see Chapter 3.5). For each measurement, the considered task is continuously repeated in a loop, while making sure that the task is fed with reasonable input data if necessary. GPIO pins of the MCU are used to indicate the start and the end of one task execution. The system is powered by a constant supply voltage V_{CC}. The current consumption I of the IPU board is determined by measuring the voltage V_{sh} over an appropriate high-side shunt resistor ($R_{sh} = 1...10\ \Omega$):

$$I = \frac{V_{sh}}{R_{sh}} \tag{5.1}$$

The supply voltage V_{CC}, the voltage V_{sh} over the shunt resistor, as well as the GPIO pin states are measured using a data acquisition device (NI USB-6216 from *National Instruments*). This measurement device features a 16-bit analog-to-digital converter and is configured to capture 1000 samples per second for each measurement channel. The gathered data is imported in MATLAB for further evaluation. The energy consumption E_{task} of one task execution is determined by integrating the instantaneous power consumption $P_{task}(t)$ over the execution time of the task, which starts at t_{start} and lasts until t_{stop}:

$$E_{task} = \int_{t_{start}}^{t_{stop}} P_{task}(t)\ dt = \int_{t_{start}}^{t_{stop}} V_{CC}(t) \cdot \frac{V_{sh}(t)}{R_{sh}}\ dt \tag{5.2}$$

V_{CC} and V_{sh} are measured values, R_{sh} is a known constant and t_{start} and t_{stop} are determined using the captured GPIO pin states. For the discrete-time evaluation in MATLAB, the integral in Equation 5.2 is replaced by a sum. The execution time of one task execution is given by:

$$t_{task} = t_{stop} - t_{start} \tag{5.3}$$

The energy consumption and the execution time is determined for all task executions in the experiment. The mean value, as well as the minimum and the maximum value over all task executions give some information about the expected energy consumption of the considered task, its variability and worst-case behaviour.
Repeating the measurements for all tasks and for different parameter settings helps optimizing the tasks for lowest energy consumption. A complete list of the results is given in Appendix C. In the following, some results are discussed in detail and the choice of the optimal parameters is explained.

5.1.1 Image Acquisition Task

The image acquisition task acquires two images in a sequence and compensates for the fixed pattern noise (FPN). For minimizing the computational effort of the motion estimation algorithm, the time difference between the two acquired images should be minimal. Therefore, the maximal frame rate of 37.5 frames per second is chosen for this application. For achieving this frame rate, the CPU frequency has to be at least 24 MHz. Using a CPU frequency of 48 MHz even allows to compensate for the FPN on-line during acquisition, while still achieving the same frame rate (see Chapter 3.1.2).
For the image acquisition task, two configurations are considered:

Configuration (1) uses a CPU frequency of 48 MHz and acquires two images with on-line FPN compensation.

Configuration (2) uses a CPU frequency of 24 MHz and acquires two images and compensates then separately for the FPN in the two images (in a sequence).

For both configurations, the supply voltage $V_{CC} = 3$ V is used, since this is the minimal supply voltage for the camera.

Configuration	CPU Freq.	Executions	E_{task} (mean)	t_{task} (mean)
on-line FPN compensation (1)	48 MHz	1152	930 µJ	53 ms
separate FPN compensation (2)	24 MHz	1829	537 µJ	61 ms

Table 5.1: Energy consumption and execution time of the image acquisition task averaged over the given number of task executions in the measurement, using two different configurations.

Table 5.1 reveals a trade-off between execution time and energy consumption for the two configurations. As expected, operating the image acquisition task at a CPU frequency of 24 MHz results in a slightly longer execution time, since the FPN compensation is done in sequence after the image acquisition. Note that the time for acquiring the two images (and therefore also the frame rate) are exactly the same for the two configurations. A CPU frequency of 48 MHz (with on-line FPN compensation) leads to a shorter task execution time, but increases the overall energy consumption significantly. This can be explained by the higher power consumption of the MSP432 for higher clock frequencies [18] and that in configuration (2), further optimizations on compiler-level are possible as the FPN compensation is done for both images simultaneously.
The evaluation reveals that configuration (2) with the lower CPU frequency is beneficial in a transiently powered system, since the overall energy consumption is 42% lower. However, for optimizing the task for speed in a continuously powered system, configuration (1) can lower the execution time by 13% at the cost of a higher energy consumption.

5.1.2 Processing Task

The processing task estimates the displacement between the two previously captured images by applying the block-matching algorithm to get an optical flow field, which is then further processed to estimate the final displacement value.
For the task characterization, the block size of the block-matching algorithm is set to 48px and the search area to ±3 and ±8 in x- and y-direction respectively (see Chapter 4.4). The power supply voltage should be chosen as low as possible, which is 2.2 V in this case, since this is the minimal supply voltage for the MSP432 when using high clock frequencies. The only remaining variable parameter for the processing task is thus the CPU frequency.

CPU Freq.	Executions	E_{task} (mean)	t_{task} (mean)	P_{task} (mean)
48 MHz	1079	757 µJ	56 ms	13.5 mW
24 MHz	1254	635 µJ	110 ms	5.8 mW
12 MHz	770	1371 µJ	388 ms	3.5 mW

Table 5.2: Energy consumption, execution time and power consumption of the processing task for different CPU frequencies, averaged over the given number of task executions in the measurement.

Table 5.2 compares the energy consumption and the execution time of the processing task for different CPU frequencies. As expected, higher CPU frequencies lead to shorter execution times. However, shorter execution times do not necessarily result in lower energy consumption, because the MSP432 MCU consumes much more power for higher clock

frequencies [18]. As the results in Table 5.2 show, the energy consumption of the processing task is lowest at a CPU frequency of 24 MHz.

5.1.3 Storage Task

For the storage task, the energy consumption of two scenarios is of interest: Writing one dataset from the internal SRAM of the MCU to the microSD Card, and flushing the whole FRAM buffer (i.e. five datasets) to the microSD Card. The MSP432 firmware supports both the usage of the microSD card with a FAT file system or without any file system by writing raw data onto the microSD Card. Both cases are evaluated to analyze the overhead caused by the FAT file system. Furthermore, the use of different CPU frequencies is evaluated.

The power supply voltage for microSD Cards needs to be at least 2.7 V [3]. Because high peak currents during the write process may cause significant voltage drops over the high-side shunt resistance ($R_{sh} = 1\ \Omega$), a supply voltage of $V_{CC} = 3$ V is used for the experiments to guarantee a supply voltage of at least 2.7 V for the microSD Card.

Task	File System	CPU Freq.	Executions	E_{task} (mean)	t_{task} (mean)
write 1 dataset	-	48 MHz	514	83162 µJ	607 ms
write 1 dataset	FAT	48 MHz	341	131801 µJ *	1037 ms *
flush 5 datasets	-	48 MHz	454	102677 µJ	875 ms
flush 5 datasets	-	24 MHz	367	110761 µJ	1084 ms
flush 5 datasets	FAT	48 MHz	126	343582 µJ *	3135 ms *

Table 5.3: Energy consumption and execution time of the microSD Card storage task for different amounts of data, different CPU frequencies and using FAT/no file system, averaged over the given number of task executions. The energy and time measurements using the FAT file system (*) show a significant dependence on the previous history of the microSD Card.

Table 5.3 presents the results of the task characterization for the different storage configurations. First of all, it is worth noting that the energy consumption of the microSD Card storage task is in the order of 100 times higher than for the image acquisition and processing tasks. The results for the configurations which use the FAT file system are marked with an asterisk (*) because the energy and time evaluations depend on the previous history of the microSD Card and show large differences between the individual executions. Beside the significant overhead that the FAT file system generates, the unpredictable energy consumption of the microSD Card task is a problem in transiently powered systems. The size of the energy buffer on the EMU needs to be specified according to the most energy-consuming task. Considering the unpredictable behaviour of the storage task when using a FAT file system, either a huge energy buffer is needed to guarantee the completion of the task or the task can not be completed in some cases. Both scenarios are unwanted in transiently powered systems and therefore no file system is used for this project.

Other than for the image acquisition and the processing tasks, both the energy consumption and the execution time is higher when using a CPU frequency of 24 MHz instead of 48 MHz. In this task, the power consumption of the microSD Card dominates the overall power consumption. Minimizing the active time of the microSD Card by using the highest possible CPU frequency of 48 MHz results in a minimal overall energy consumption.

SD Card Variability

For the experiments, a Kingston 8GB Class 10 microSD Card is used. The high power consumption of this microSD Card raises the question whether other microSD Cards consume less power. For that, the task characterization is repeated using four different microSD cards. Exactly the same MCU configuration is used for all experiments (CPU frequency: 48 MHz, supply voltage: 3 V, flush five datasets from the FRAM to the microSD Card). The results are shown in Table 5.4. Indeed, the differences between the distinct microSD Cards are astonishing. Executing the same task using a SanDisk 2GB mircoSD Card consumes in average only around 13% of the energy than using the Kingston 8GB microSD Card. Because the microSD Card storage task is the most energy-consuming task in the vision sensor application, the energy buffer of the EMU when using the SanDisk 2GB microSD Card can be almost eight times smaller than when using the Kingston 8GB microSD Card.

Task	microSD Card	Executions	E_{task} (mean)	E_{task} (max)	t_{task} (mean)
flush 5 datasets	Kingston 8GB	454	102677 µJ	165967 µJ	875 ms
flush 5 datasets	Kingston 2GB	841	36207 µJ	86898 µJ	471 ms
flush 5 datasets	SanDisk 2GB	1134	13580 µJ	28633 µJ	348 ms
flush 5 datasets	Samsung 8GB	422	85793 µJ	216139 µJ	940 ms

Table 5.4: Energy consumption and execution time of the microSD Card storage task for the same configuration (CPU frequency: 48 MHz, supply voltage: 3 V) but for different microSD Cards, averaged over the given number of task executions.

Remarkable are also the variations of the energy measurements when repeating the same task several times. The energy consumption of the Samsung 8GB microSD Card seems to be quite unpredictable, since the energy consumption is different for every experiment and the maximal measured energy consumption is huge. This behaviour is highly undesirable in a transiently powered system. The SanDisk 2GB microSD Card shows a relatively constant energy consumption behaviour with only a few outliers, which are still in a reasonable range.

5.2 Performance of the Transiently Powered Vision Sensor

The interesting parameters for evaluating the transiently powered vision sensor are the number of velocity estimations per time and the efficiency of the EMU in dependence on the available input power. In particular, it is interesting if using a non-volatile memory hierarchy (NVMH, see Chapter 2.3) can really increase the performance of the system.

Experimental Set-Up

For the evaluation, the custom vision sensor prototype is operated as a transiently powered system, using the Energy Management Unit (EMU) and the Image Processing Unit (IPU). The EMU harvests energy from a solar panel (flexible MP3-37 solar panel from PowerFilm with an area of 42 cm^2) that is exposed to constant illumination by a lamp. The experiment runs several times for different illumination levels in order to evaluate the performance of the transiently powered system for different constant input power levels ranging from 198 µW to 2376 µW. Each experiment lasts between 300 s (for high input powers) and 900 s (for low input powers) to ensure capturing a representative amount of burst executions.

In order to determine the source input power P_{in} and the load power P_{load}, the voltage drops ($V_{sh,in}$ and $V_{sh,load}$) over two shunt resistors are measured - one between the solar panel and the EMU ($R_{sh,in} = 22\ \Omega$) and one between the EMU and the IPU ($R_{sh,load} = 1\ \Omega$). Furthermore, the source input voltage V_{in}, the load supply voltage V_{load}, as well as the voltage over the energy buffer capacitor V_{cap} are measured. The state of a GPIO pin indicating the activity of the load is also captured to count the number of burst executions. The voltages are captured using the data acquisition device NI USB-6216 from *National Instruments*. The further processing of the data is done in MATLAB.

Evaluating the Non-Volatile Memory Hierarchy

The experiment is repeated two times for the two schedules mentioned in Chapter 2.3 and depicted in Figure 5.2 and Figure 5.3. In the first schedule, the only non-volatile memory is an SD Card. The schedule consists of one single burst (compare Figure 5.2):

Schedule (1): use only microSD Card as non-volatile memory

- Burst 1: acquisition of two images + FPN compensation + motion estimation + save dataset and system state to microSD Card

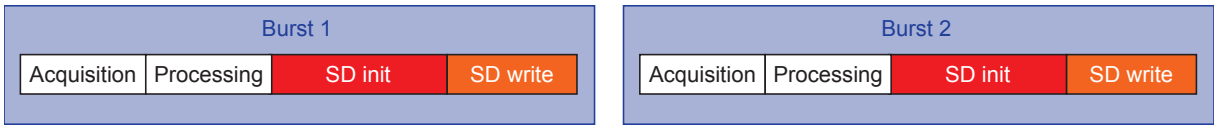


Figure 5.2: Using only a SD Card as a non-volatile memory, the SD Card initialization has to be repeated in every sensor reading.

In the second schedule, a non-volatile memory hierarchy consisting of an FRAM buffer and a microSD Card is used. The results are stored in the FRAM buffer until it is full. Then, a special burst is introduced, that flushes the content of the FRAM buffer onto the microSD Card. The schedule then looks as follows (compare Figure 5.3):

Schedule (2): use NVMH consisting of an FRAM buffer and a microSD Card

- Bursts 1 to n: acquisition of two images + FPN compensation + motion estimation + save dataset and system state in FRAM buffer
  (repeat n times, until FRAM buffer is full)
- Burst $n + 1$: flush the content of the FRAM buffer to the microSD Card

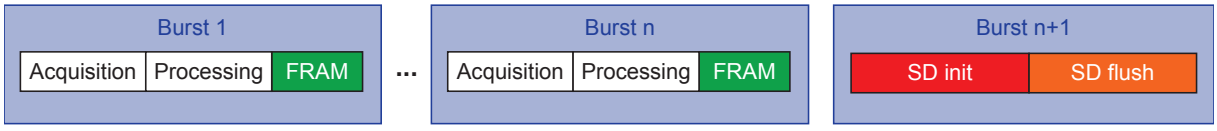


Figure 5.3: Using only a NVMH consisting of an FRAM buffer and an SD Card can spread the large SD Card initialization costs over many sensor readings.

The size of the energy buffer capacitor on the EMU is dimensioned using the results from the task characterization in Chapter 5.1. Each burst in Schedule (1) is expected

to consume 7856 µJ of energy. The most energy-consuming burst in Schedule (2) is Burst $n+1$, expected to consume 13580 µJ of energy. Those are the expected mean energy values, but the actual energy consumption can slightly vary, especially in the microSD Card tasks. It therefore makes sense to choose a slightly larger capacitance to guarantee the proper execution of most microSD Card write tasks. Using Equation 5.6, the necessary capacitance for storing the given amount of energy between 5.1 V and 3 V can be calculated. The capacitances $C_1 = 1000$ µF for Schedule (1) and $C_2 = 2200$ µF for Schedule (2) are chosen.

Number of Velocity Estimations per Time

For a given constant input power level, the number of velocity estimations per minute is plotted in Figure 5.4. The number of velocity estimations is proportional to the input power. The slope of the curve is determined by the average energy for one velocity estimation. Using a NVMH significantly increases the number of velocity estimations in a given time period. In this application, the NVMH can increase the performance by more than factor 1.8.

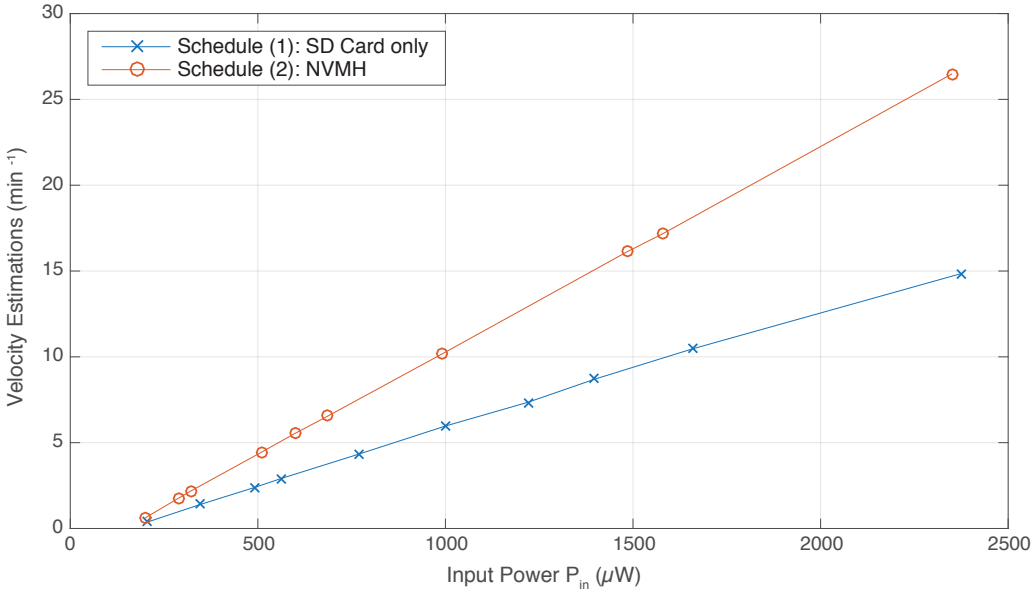


Figure 5.4: Number of velocity estimations per minute at different input power levels. The NVMH can significantly increase the performance of the system.

Efficiency of the EMU

The instantaneous input power and load power are calculated as follows:

$$P_{in}(t) = V_{in}(t) \cdot \frac{V_{sh,in}(t)}{R_{sh,in}} \tag{5.4}$$

$$P_{load}(t) = V_{load}(t) \cdot \frac{V_{sh,load}(t)}{R_{sh,load}} \tag{5.5}$$

By integrating the instantaneous input power $P_{in}(t)$ over the whole evaluation time interval, one gets the total input energy E_{in} harvested from the solar panel. The consumed energy

by the vision sensor E_{load} is calculated by integrating the load power $P_{load}(t)$ over the time periods when the load is active. This means that the small amount of energy that the vision sensor consumes between the bursts is added to the losses of the system. The efficiency η of the system is then given as $\eta = E_{load}/E_{in}$.

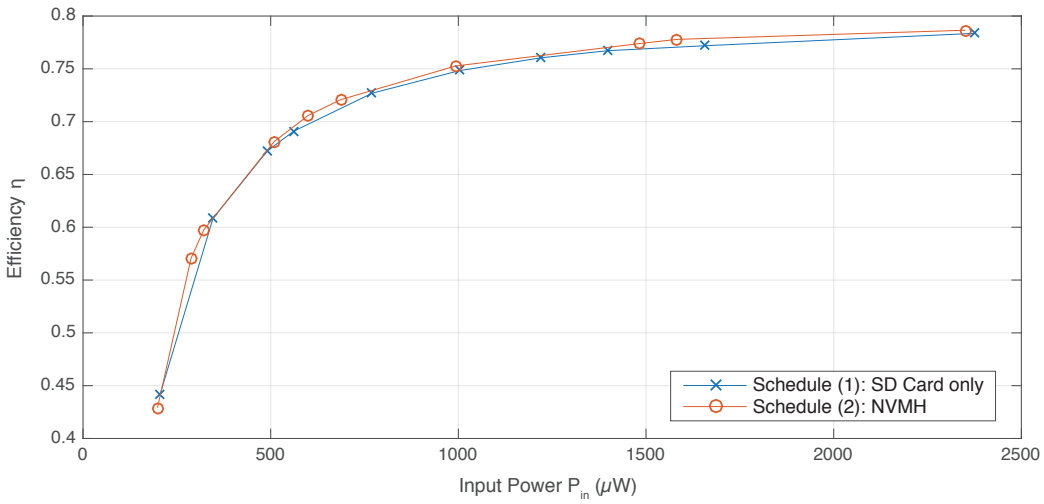


Figure 5.5: Efficiency $\eta = E_{load}/E_{in}$ of the transiently powered vision sensor at different input power levels.

The measured efficiency is plotted in Figure 5.5 for different input power levels. As expected, the efficiency curve is consistent for both schedules. The efficiency of the system increases for higher input power levels. The product of the efficiencies of the boost and the buck converters on the EMU limits the maximally achievable efficiency. The highest measured efficiency is 78.7%. For input power levels above 600 µW, the efficiency is higher than 70%. For input power levels below 330 µW, the efficiency drops sharply below 60%. The lowest input power level in the experiment is 198 µW and achieves an efficiency of 43%.

5.3 Real-World Experiment

The developed velocity estimation sensor is designed to estimate the speed of a walking person, if it is attached to the glasses. The real-world experiment shows the performance of the vision sensor prototype for this application. To get as many velocity estimations per time as possible, the velocity estimation sensor is continuously powered by a battery in this experiment. The estimations then happen at deterministic time instances, making it easier to assign the estimation to a certain activity of the person wearing the sensor. The behaviour of the transiently powered vision sensor is then simulated in MATLAB using the continuously acquired data.

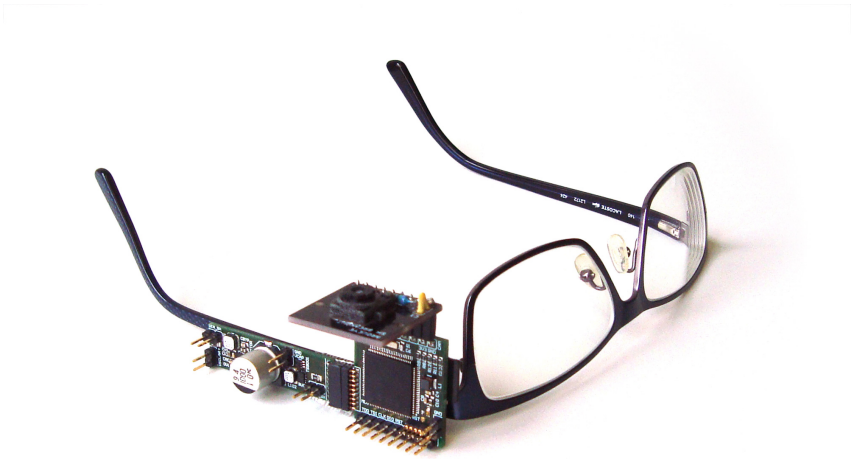


Figure 5.6: Velocity estimation sensor attached to glasses for performing the real-world experiment.

Experimental Set-Up

The IPU board and the battery board of the vision sensor prototype as described in Chapter 3.5 are used for the real-world experiment. The system is powered by a battery pack. The wearable vision sensor prototype is attached to the glasses (see Figure 5.6), such that the Stonyman vision chip faces downwards and captures the floor in front of the person.
The vision sensor executes a loop of acquiring two images, estimating the displacement between two images and writing the images and the results onto the microSD Card. Because changing the clock frequencies during runtime causes some overheads and for optimizing the application for speed, a CPU frequency of 48 MHz is used for the whole application. The search area of the block-matching algorithm is configured to ± 3 pixels in x-direction and ± 8 pixels in y-direction. The block size of the block-matching algorithm is set to 48px, which was found to be optimal according to the calibration in Chapter 4.4. Using this set-up, the sensor evaluation rate is measured to be approximately 8 velocity estimations per second. However, due to variations in the microSD Card write times, the evaluation rate is not exactly constant. According to the measurements from the task characterization (see Appendix C), the vision sensor consumes an average power of approximately 40 mW.

Context Recognition

In the first experiment, the person wearing the velocity estimation sensor stands, walks and stands again, where each phase lasts for some seconds. After conducting the experiment, the displacement values as well as the images are copied to the PC for evaluating and visualizing the experiment in MATLABThe experiment is conducted without any markers on a floor where sufficient brightness structure is expected. Figure 5.7 shows an excerpt of the captured images during the experiment. The image quality is good and the structure of the floor is clearly visible.

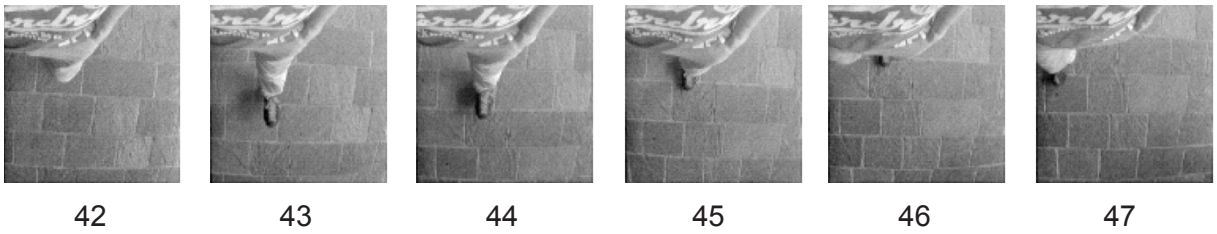


Figure 5.7: Excerpt of the images captured during the real-world experiment.

In Figure 5.8, the walking speed estimations calculated by the vision sensor prototype are depicted as blue crosses. The red line shows the low-pass filtered velocity estimation, post-processed in MATLAB using a moving average finite impulse response (FIR) filter. The different phases of the experiment (standing / walking / standing) can clearly be distinguished. This shows that the velocity estimation sensor is able to produce useful information for context recognition.

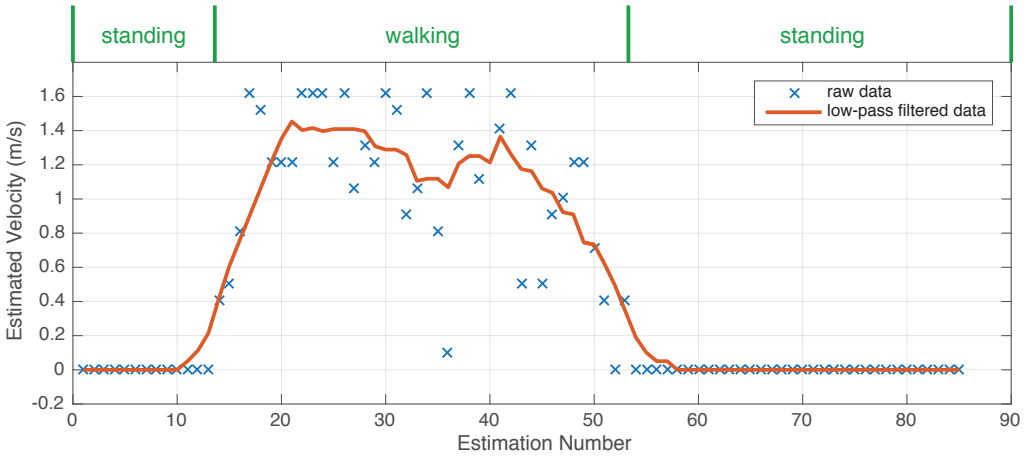


Figure 5.8: Velocity of the walking person estimated by the vision sensor prototype (blue crosses) and post-processed in MATLAB using a low-pass filter (red line).

Estimation of the Walking Speed

As the velocity estimation sensor produces useful results for context recognition, the question arises whether the velocity estimation sensor can distinguish between different walking speeds. For that, the person wearing the vision sensor repeats the previous experiment with different phases: stand, walk slowly, stand, walk rapidly and stand again. The experiment is conducted on the same floor and using the same set-up as the first experiment.

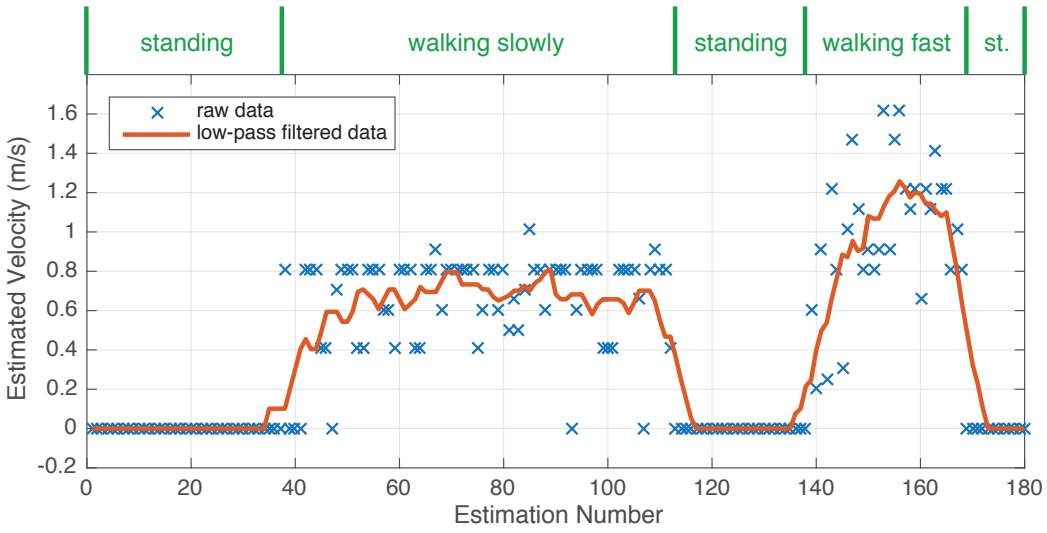


Figure 5.9: Velocity estimation of the vision sensor prototype (blue crosses) for a person walking at different speeds and post-processed in MATLAB using a low-pass filter (red line).

Figure 5.9 presents the walking speed estimations depicted as blue crosses, as well as the low-pass filtered data using a moving-average FIR filter in MATLAB (red line) . The different phases of the experiment can clearly be recognized. Furthermore, the velocity estimation sensor is really able to distinguish between different walking speeds.

Simulation: Transiently Powered Velocity Sensor

The previous experiments show that the visual velocity estimation sensor produces useful results if it is powered continuously by a battery. The behaviour of the transiently powered system at reduced input power can be simulated using the data from the experiment with the battery-powered system. The continuously powered vision sensor produces approximately 8 sensor evaluations per second and has a power consumption of approximately 40 mW. In Chapter 5.2, the velocity estimation rate of the transiently powered sensor is measured for different constant input power levels. Using this information, the real-world experiment can be simulated in MATLAB for lower input powers using the vision sensor in transiently powered operation.
Figure 5.10 shows the simulated behaviour of the transiently powered system in the real-world experiment (blue crosses), if it is powered at an input power level of 8 mW, which corresponds to 1/5 of the active load power consumption. The simulation is based on data of the real-world experiment using the continuously powered system (grey crosses). Even if the input power is reduced by factor 5, the transiently powered velocity estimation sensor does still produce useful measurements for context recognition.

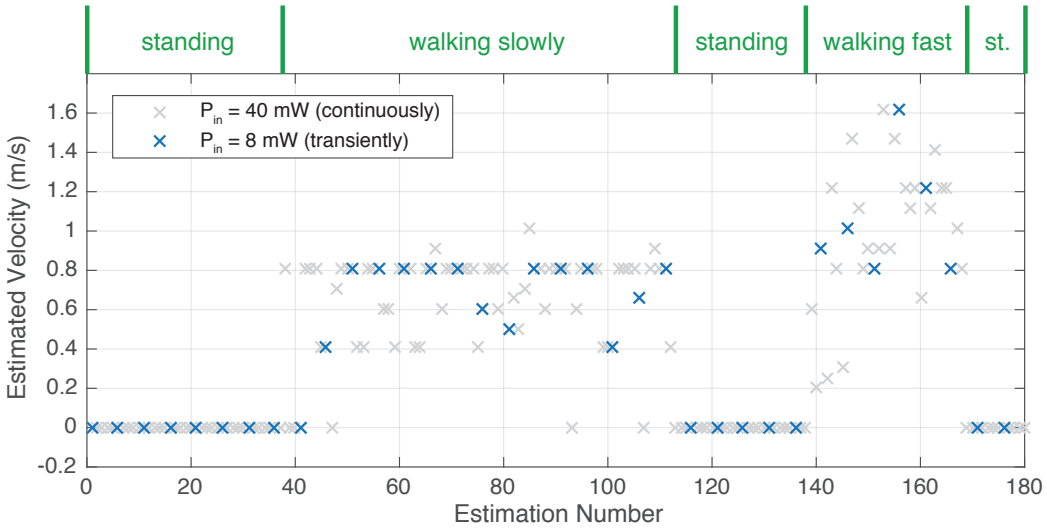


Figure 5.10: Simulated velocity estimations of the transiently powered vision sensor using an input power of 8 mW (blue crosses), based on the data from the real-world experiment (grey crosses).

The input power is even further reduced in Figure 5.11 and Figure 5.12 to 4 mW and 2 mW respectively. The simulation shows, that the transiently powered system can deal with input power levels that are significantly lower than the active load power consumption, but still correctly executes the computationally intensive velocity estimation application. Even though the input power is 20 times lower than the power consumption of the continuously powered load, the transiently powered vision sensor leads to useful, sporadic velocity estimations, that can be used e.g. for context recognition.

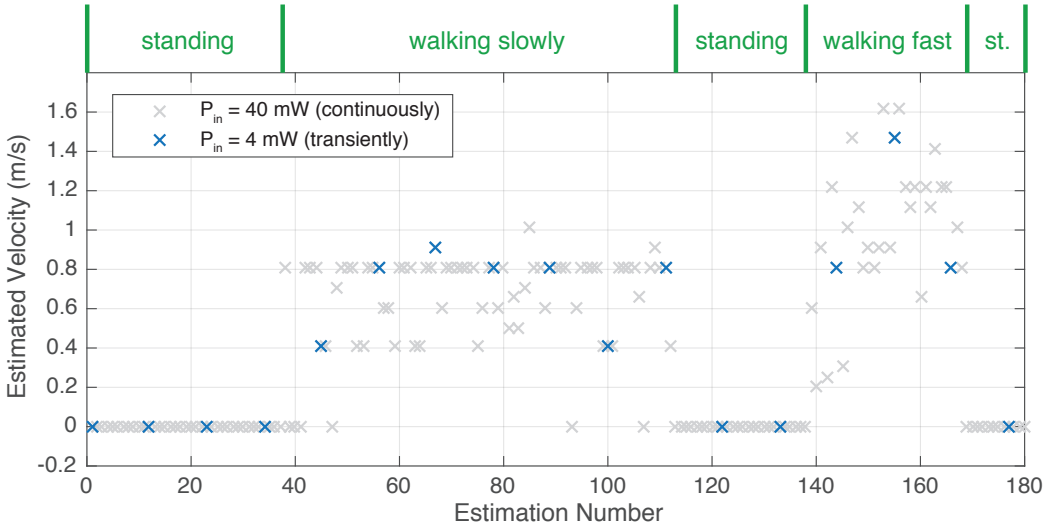


Figure 5.11: Simulated velocity estimations of the transiently powered vision sensor using an input power of 4 mW (blue crosses), based on the data from the real-world experiment (grey crosses).

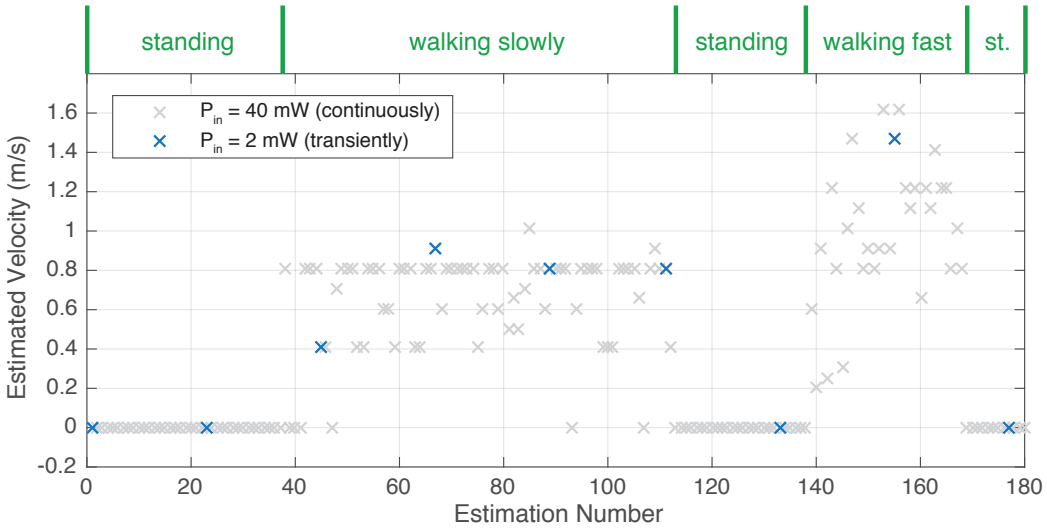


Figure 5.12: Simulated velocity estimations of the transiently powered vision sensor using an input power of 2 mW (blue crosses), based on the data from the real-world experiment (grey crosses).

5.4 Summary

Optimized Task Set

As part of the system design process, the task characterization measurements help finding the optimal parameters for a minimal energy consumption of the system. Furthermore, the task characterization provides the necessary data for configuring the EMU. The size of the energy buffer in the EMU has to be specified according to the most energy-consuming task in the application. In Table 5.5, the characterization of all tasks using the optimal parameters is summarized. The energy buffer capacity on the EMU can be calculated using the formula for the stored energy E in a buffer of capacitance C at a voltage level V:

$$E = \frac{1}{2}CV^2 \tag{5.6}$$

Task	CPU Freq.	V_{CC}	E_{task} (mean)	t_{task} (mean)
acquire 2 images + FPN comp.	24 MHz	3 V	537 µJ	61 ms
processing	24 MHz	2.2 V	635 µJ	110 ms
write 1 dataset to microSD Card	48 MHz	3 V	6684 µJ	226 ms
flush 5 datasets to microSD Card	48 MHz	3 V	13580 µJ	348 ms

Table 5.5: Measured energy consumption and execution time of all tasks using the optimal parameters.

Performance Enhancement using NVMH

The evaluation of the transiently powered system in Section 5.2 reveals, that using a NVMH can significantly improve the energy efficiency of the system. In this vision sensor application, the number of velocity estimations per time can be increased by factor 1.8 using a NVMH, instead of using only an SD Card as non-volatile memory. The efficiency

of the EMU reaches almost 80% for high input power levels. For input power levels below 330 µW, the efficiency drops sharply below 60%. The transiently powered system can therefore even operate with very low input power levels, that are much lower than the active load power consumption.

Performance in Real-World Experiment

The real-world experiment described in Section 5.3 shows, that the developed velocity estimation sensor can produce useful walking speed estimations if the vision sensor is attached to someone's glasses. The experiment is conducted on an ordinary floor with relatively poor brightness structure. Nevertheless, the velocity estimations allow to clearly distinguish between different activities and walking speeds.

Chapter 6

Conclusion and Outlook

In this master thesis, a wearable prototype of a visual velocity estimation sensor is developed, that is able to operate without any battery. The vision sensor can be attached to someone's glasses for estimating the walking speed of that person. It harvests energy from a solar panel to produce sporadic velocity estimations, whereas the sensor reading rate depends on the currently available input power. The hardware and firmware of the vision sensor is highly optimized for ultra-low power consumption and high performance (see Chapter 3). Different motion estimation algorithms are investigated for calculating a velocity estimation using visual information (see Chapter 4). The evaluation has shown that the final vision sensor prototype can produce reliable velocity estimations that can be used for context recognition (see Chapter 5).
The battery-less vision sensor is designed according to the concept of transiently powered systems (see Chapter 2). The operation of such systems is a function of the currently available input power. The successful design of the battery-less vision sensor in this master thesis has shown, that the idea of transiently powered systems can be put into practice and even computationally intensive applications can be implemented and correctly executed in battery-less devices.
For enhancing the energy-efficiency of the transiently powered vision sensor, a non-volatile memory hierarchy (NVMH) is presented in this thesis (see Chapter 2.3). The NVMH uses an FRAM chip as a cache for an SD Card, such that the expensive SD Card initialization costs are spread over many sensor readings. A NVMH can significantly reduce the average energy requirement per sensor reading and therefore increase the overall performance of the transiently powered system. In the vision sensor prototype, the performance is increased by factor 1.8 if using a NVMH, bringing down the average energy required per estimation to 4.3 mJ (see Chapter 5.2).
The concept of transiently powered systems is a promising approach to account for the current need for innovative energy harvesting concepts to power nodes in huge sensor networks as the Internet of Things. However, this concept is fairly new and many questions remain unsolved until today. The task scheduling and the energy management in transiently powered systems can be improved. In particular, more intelligence can be added to the energy management unit in order to increase the efficiency of the system, or to optimize the scheduling of tasks. Furthermore, the concept of transiently powered systems needs to be expanded to other sensor applications and other energy harvesting sources (e.g. harvesting kinetic or thermal energy) for exploring new use cases. Building a large sensor network consisting of transiently powered nodes will lead to interesting challenges for signal processing to extract desired information from sensor nodes that only operate sporadically and unpredictably.

Bibliography

[1] FatFS: Generic FAT File System Module. `http://elm-chan.org/fsw/ff/00index_e.html`. Accessed: 2016-02-26.

[2] SD Card low-level Disk I/O Control and FatFS Implementation for MSP432. `https://github.com/bluehash/MSP432Launchpad/tree/master/MSP432-Launchpad-FatFS-SDCard`. Accessed: 2016-07-22.

[3] SD Card Association. SD Specifications Part1 Physical Layer Simplified Specification, Version 4.10, 2013.

[4] Aroh Barjatya. Block matching algorithms for motion estimation. *IEEE Transactions Evolution Computation*, 8(3):225–239, 2004.

[5] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM computing surveys (CSUR)*, 27(3):433–466, 1995.

[6] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. *Intel Corporation*, 2001.

[7] Inc. Centeye. Stonyman and Hawksbill Vision Chips Silicon Documentation, 2013. Datasheet.

[8] Cypress Semiconductor Corporation. FM25V10: 1-Mbit Serial (SPI) F-RAM, 2015. Datasheet.

[9] Gunnar Farnebäck. Very high accuracy velocity estimation using orientation tensors, parametric motion, and simultaneous segmentation of the motion field. In *Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 171–177. IEEE, 2001.

[10] Hassan Foroosh, Josiane B Zerubia, and Marc Berthod. Extension of phase correlation to subpixel registration. *Image Processing, IEEE Transactions on*, 11(3):188–200, 2002.

[11] Max Frantz, Ivan Penskiy, and Sarah Bergbreiter. Reu: Examining vision sensor for cm-scale robots. 2013.

[12] James J. Gibson. The perception of the visual world. 1950.

[13] Andres Gomez, Lukas Sigrist, Michele Magno, Luca Benini, and Lothar Thiele. Dynamic energy burst scaling for transiently powered systems. In *2016 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 349–354. IEEE, 2016.

[14] Dominik Honegger, Pierre Greisen, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. Real-time velocity estimation based on optical flow and disparity matching. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5177–5182. IEEE, 2012.

[15] Dominik Honegger, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1736–1741. IEEE, 2013.

[16] Berthold K. Horn and Brian G. Schunck. Determining optical flow. In *1981 Technical symposium east*, pages 319–331. International Society for Optics and Photonics, 1981.

[17] Texas Instruments. MSP430FR59xx: Mixed-Signal Microcontrollers, 2015. Datasheet.

[18] Texas Instruments. MSP432P401R: Low-Power Mixed-Signal Microcontroller, 2015. Datasheet.

[19] Texas Instruments. MSP Low-Power Microcontrollers. `http://www.ti.com/lit/sg/slab034ad/slab034ad.pdf`, 2016. Accessed: 2016-07-22.

[20] Joshua D Jackson, Dale W Callahan, J Marstrander, Jim A Richardson, and Ihsan K Hakima. Low-cost precision tracking of vehicles using optical navigation technology. *WSEAS Signal processing, computational geometry and artificial vision*, 8:164–171, 2008.

[21] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

[22] Addison Mayberry, Pan Hu, Benjamin Marlin, Christopher Salthouse, and Deepak Ganesan. ishadow: Design of a wearable, real-time mobile gaze tracker. In *Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services*, MobiSys '14, pages 82–94. ACM, 2014.

[23] Vikrant More, Hitendra Kumar, Sarthak Kaingade, Pradeep Gaidhani, and Nitin Gupta. Visual odometry using optic flow for unmanned aerial vehicles. In *Cognitive Computing and Information Processing (CCIP), 2015 International Conference on*, pages 1–6. IEEE, 2015.

[24] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–652. IEEE, 2004.

[25] Kathryn Schneider, Joseph Conroy, and Wiliam Nothwang. Computing optic flow with ardueye vision sensor. Technical report, DTIC Document, 2013.

[26] Kathryn Schneider, Joseph Conroy, and Wiliam Nothwang. Computing optic flow with ardueye vision sensor. Technical report, DTIC Document, 2013.

[27] L. Spadaro, M. Magno, and L. Benini. Poster abstract: Kinetisee - a perpetual wearable camera acquisition system with a kinetic harvester. In *2016 15th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, pages 1–2, April 2016.

[28] Richard Szeliski. Computer Vision: Algorithms and Applications. 2011.

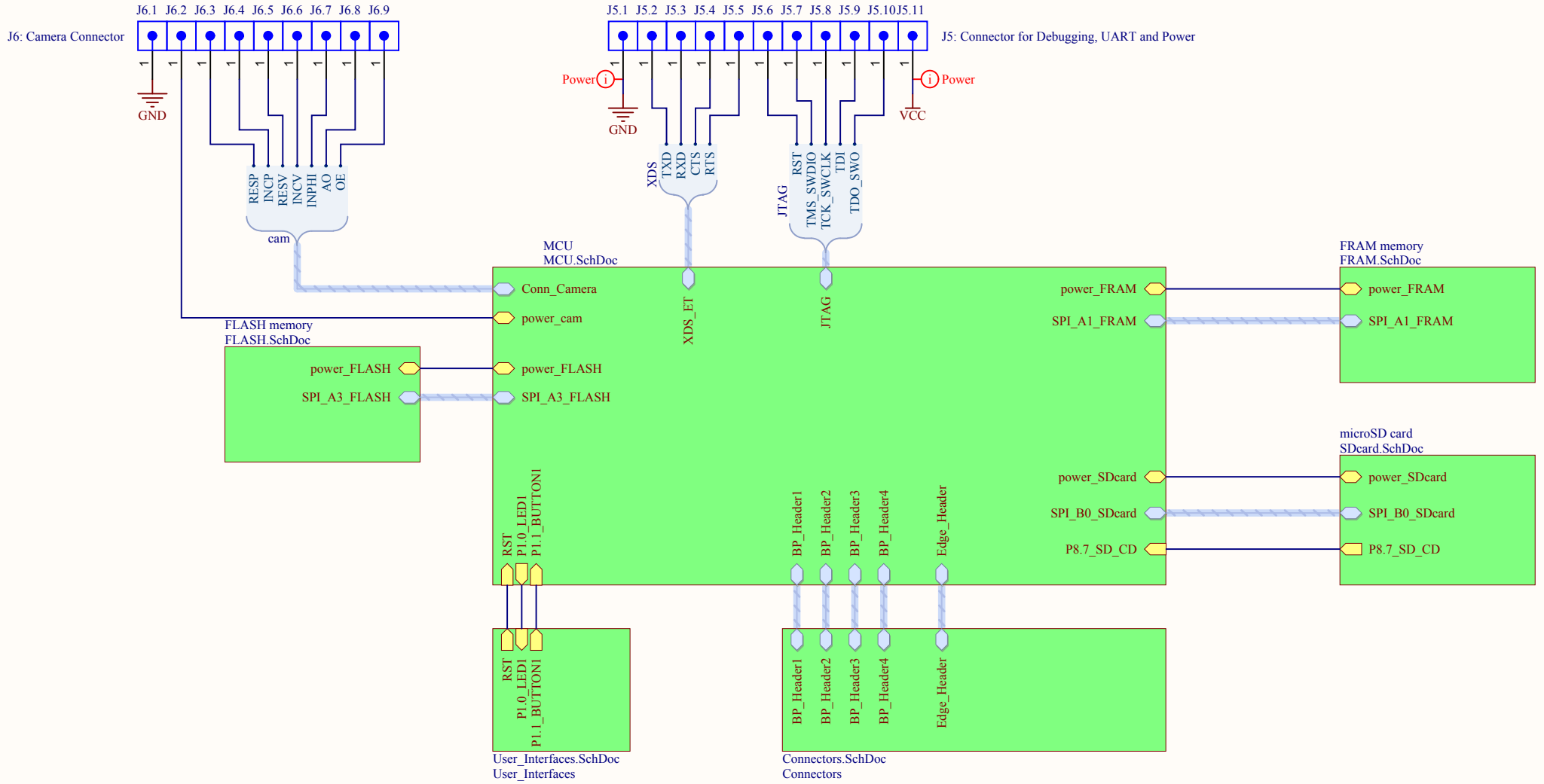
[29] L. Torres, R. M. Brum, L. V. Cargnini, and G. Sassatelli. Trends on the application of emerging nonvolatile memory to processors and programmable devices. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 101–104, May 2013.

[30] J. Zhao, Hongyi Shen, and N. Xi. Non-vector space landing control for a miniature tailed robot. In *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pages 2154–2159, Sept 2015.

Appendix A

MSP432 Demoboard

On the following pages, the schematics of the MSP432 Demoboard are shown. The board was designed with Altium Designer.



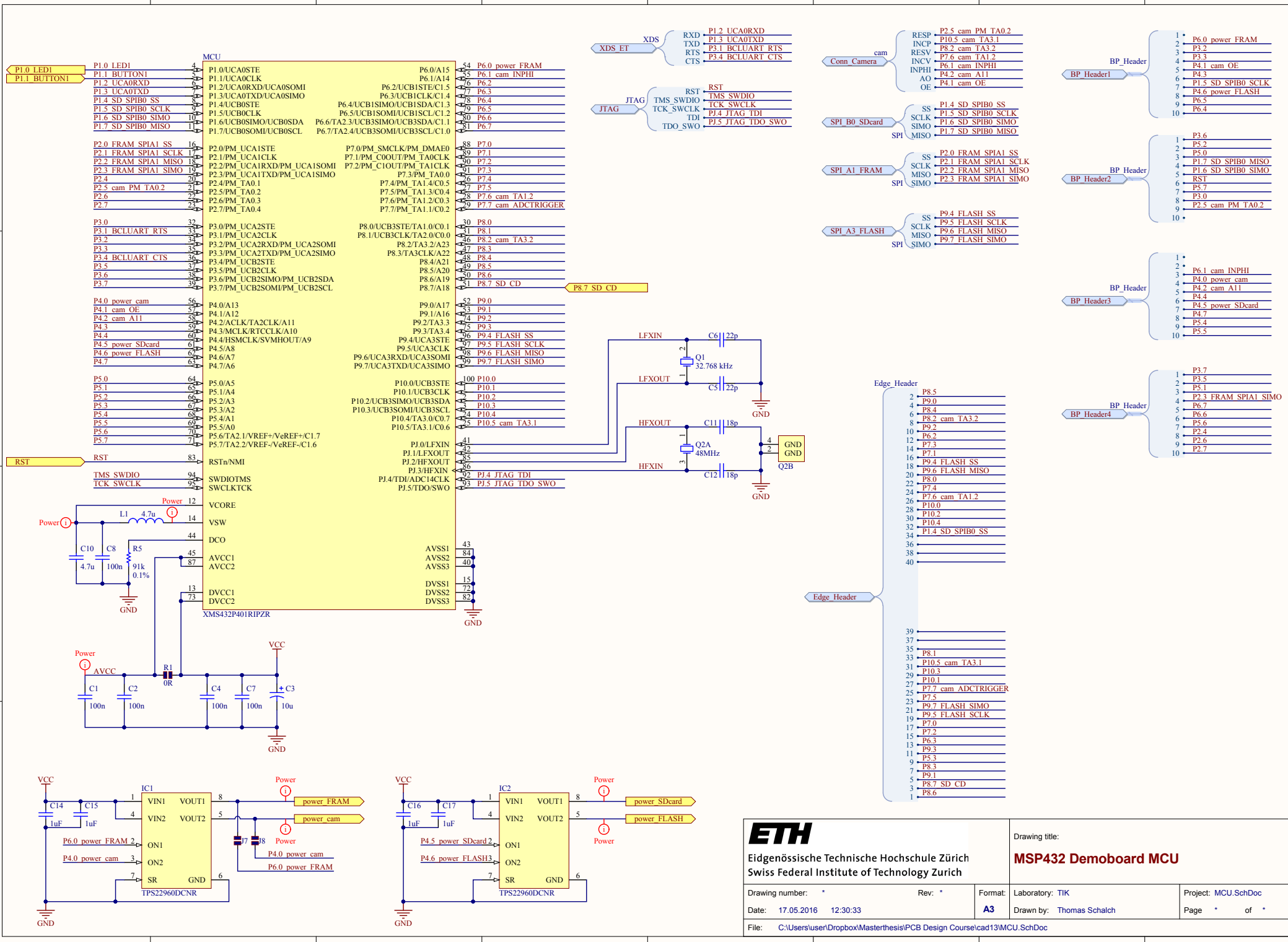
ETH

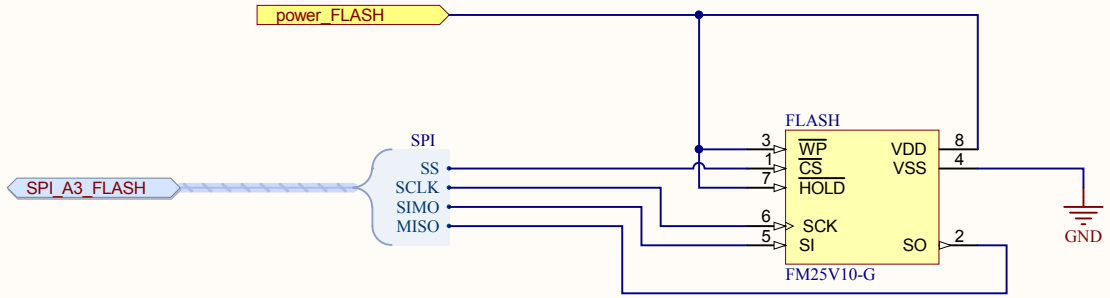
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Drawing title:

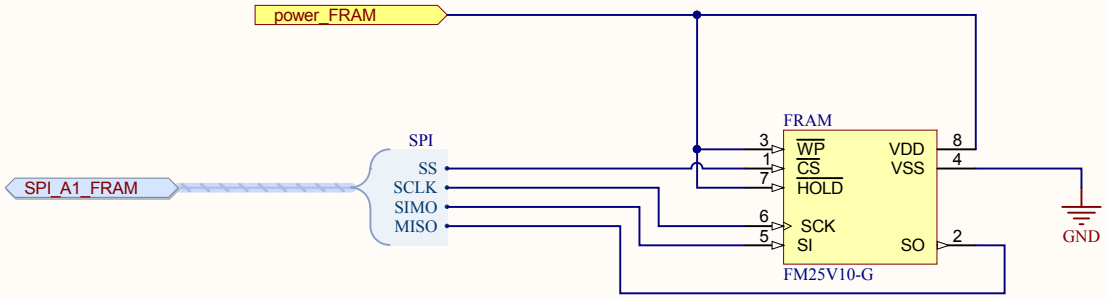
MSP432 Demo Board

Drawing number: * Rev: *	Format: A4 Q	Laboratory: TIK	Project: MSP432_demoboard.SchDoc
Date: 17.05.2016 12:30:34		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\PCB Design Course\cad13\MSP432_demoboard.SchDoc			

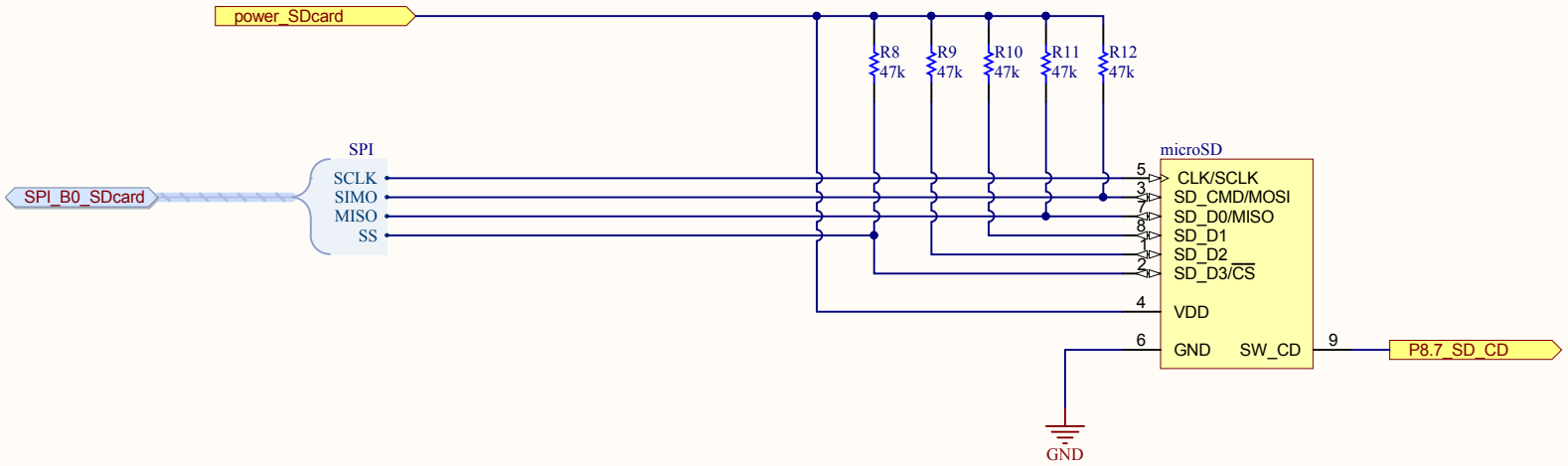

MCU
XMS432P401RIPZR
XDS
XDS_ET
JTAG
Conn_Camera
cam
SPI_B0_SDcard
SPI_A1_FRAM
SPI_A3_FLASH
BP_Header
BP_Header1
BP_Header2
BP_Header3
BP_Header4
Edge_Header
LFXIN
LFXOUT
HFXOUT
HFXIN
Q1
32.768 kHz
Q2A
48MHz
Q2B
GND
VCC
Power
AVCC
IC1
IC2
TPS22960DCNR
power_FRAM
power_cam
power_SDcard
power_FLASH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Drawing title:
MSP432 Demoboard MCU
Drawing number: *
Rev: *
Format: A3
Laboratory: TIK
Project: MCU.SchDoc
Date: 17.05.2016 12:30:33
Drawn by: Thomas Schalch
Page * of *
File: C:\Users\user\Dropbox\Masterthesis\PCB Design Course\cad13\MCU.SchDoc



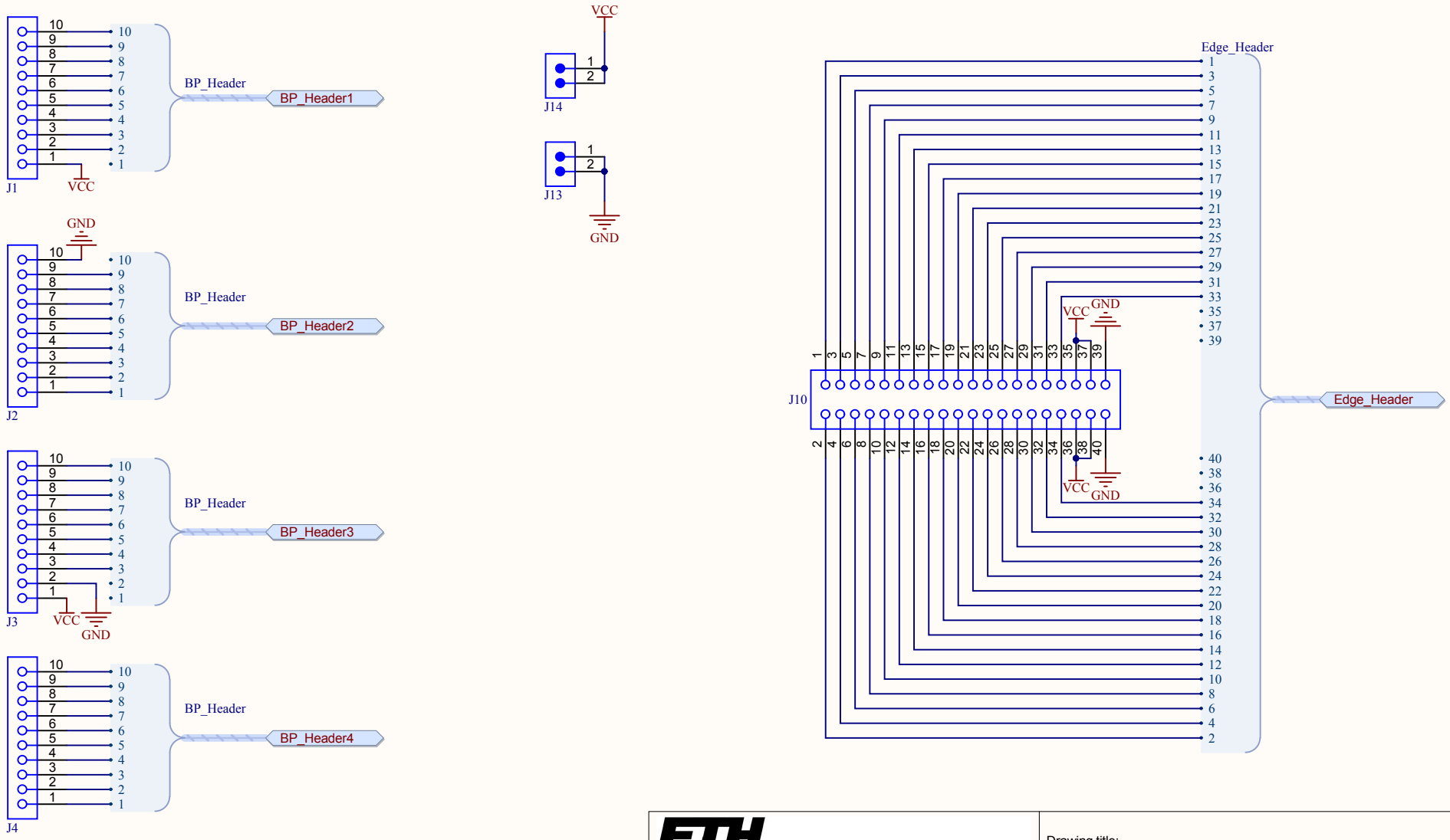
ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Drawing title: **MSP432 Demoboard FLASH memory**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: TIK	Project: FLASH.SchDoc
Date: 17.05.2016 12:30:33		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\PCB Design Course\cad13\FLASH.SchDoc			



ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Drawing title: **MSP432 Demoboard FRAM memory**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: TIK Drawn by: Thomas Schalch	Project: FRAM.SchDoc Page * of *
Date: 17.05.2016 12:30:33			
File: C:\Users\user\Dropbox\Masterthesis\PCB Design Course\cad13\FRAM.SchDoc			



ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Drawing title: **MSP432 Demoboard microSD card**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: TIK	Project: SDcard.SchDoc
Date: 17.05.2016 12:30:34		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\PCB Design Course\cad13\SDcard.SchDoc			



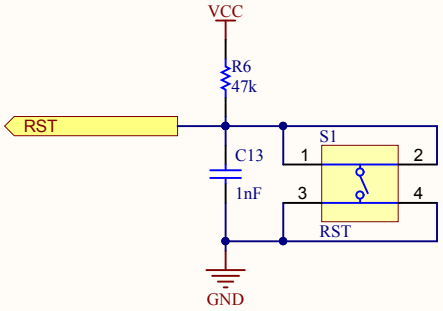
Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Drawing title: **MSP432 Demoboard Connection Headers**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: TIK Drawn by: Thomas Schalch	Project: Connectors.SchDoc Page * of *
Date: 17.05.2016 12:30:33			
File: C:\Users\user\Dropbox\Masterthesis\PCB Design Course\cad13\Connectors.SchDoc			

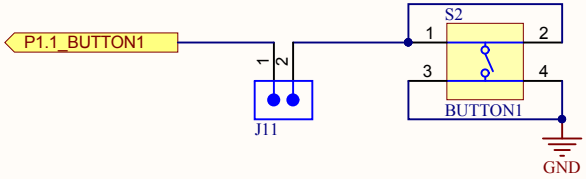
M1
Mounting Hole

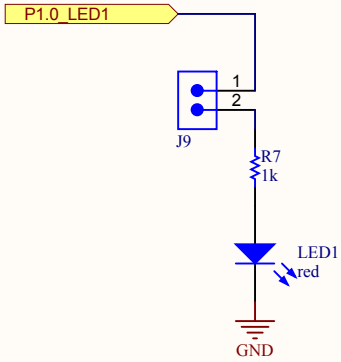
M2
Mounting Hole

M3
Mounting Hole

M4
Mounting Hole







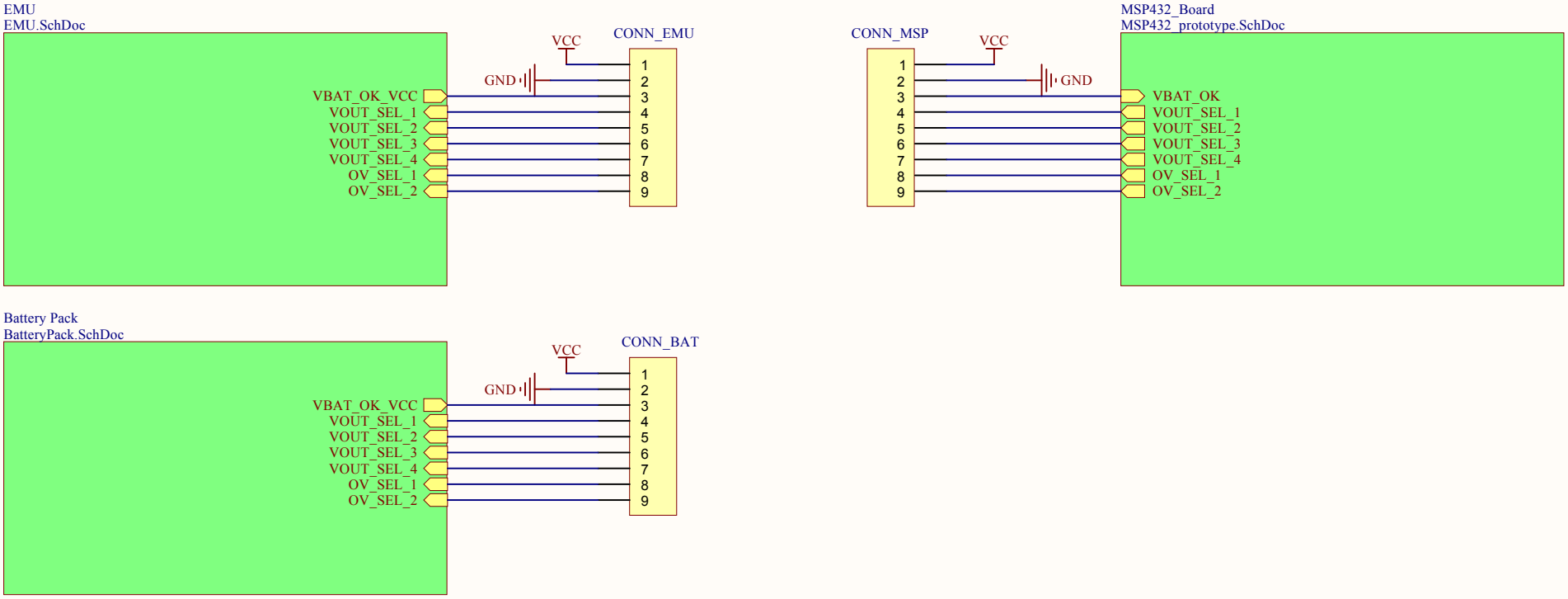
ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Drawing title: **MSP432 Demoboard User Interfaces**		
Drawing number: * Rev: *	Format: A4 Q	Laboratory: TIK	Project: User_Interfaces.SchDoc
Date: 17.05.2016 12:30:34		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\PCB Design Course\cad13\User_Interfaces.SchDoc			

Appendix B

Vision Sensor Prototype

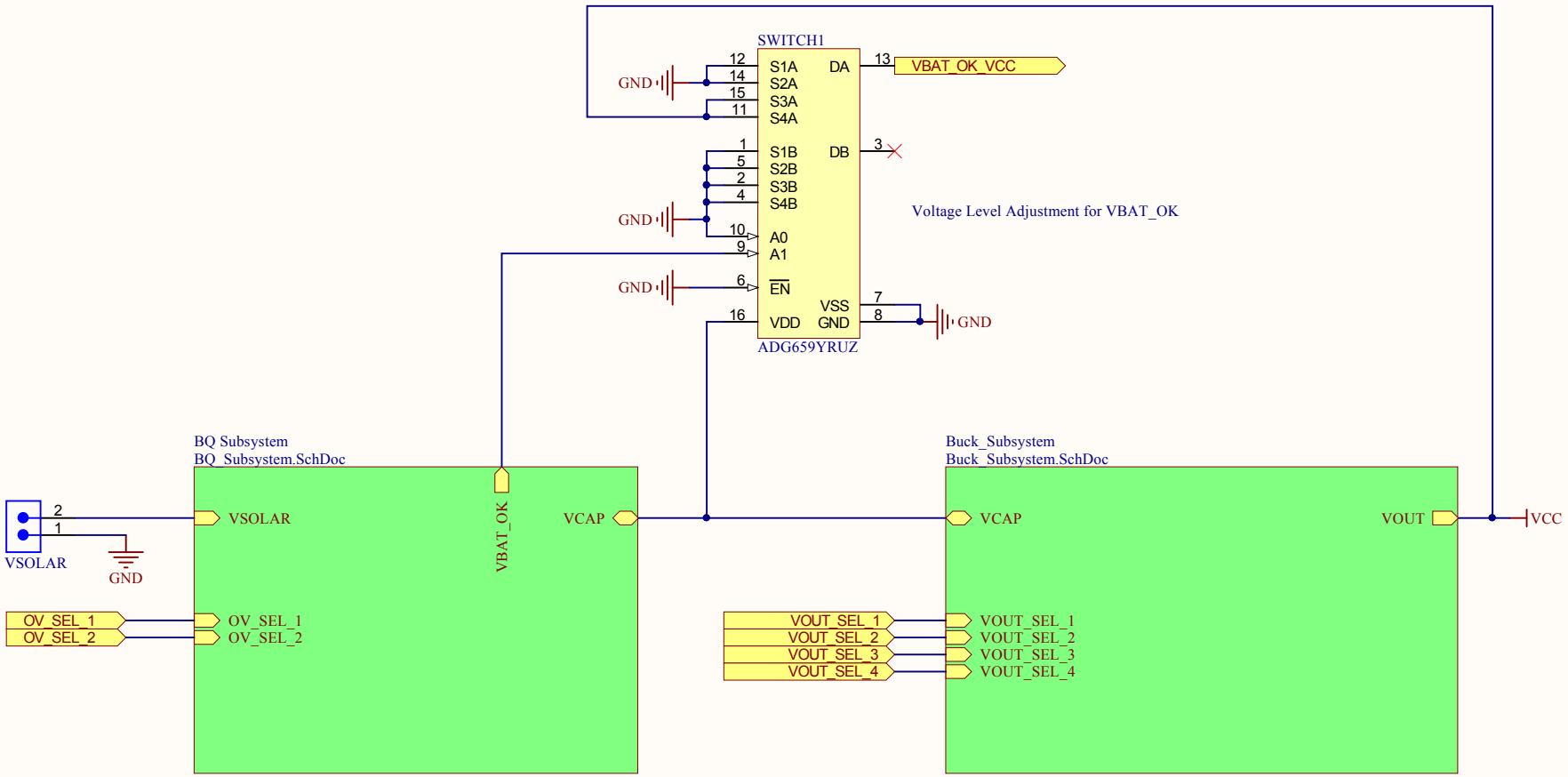
On the following pages, the schematics of the vision sensor prototype are shown. The board was designed with Altium Designer.

Vision Sensor Prototype



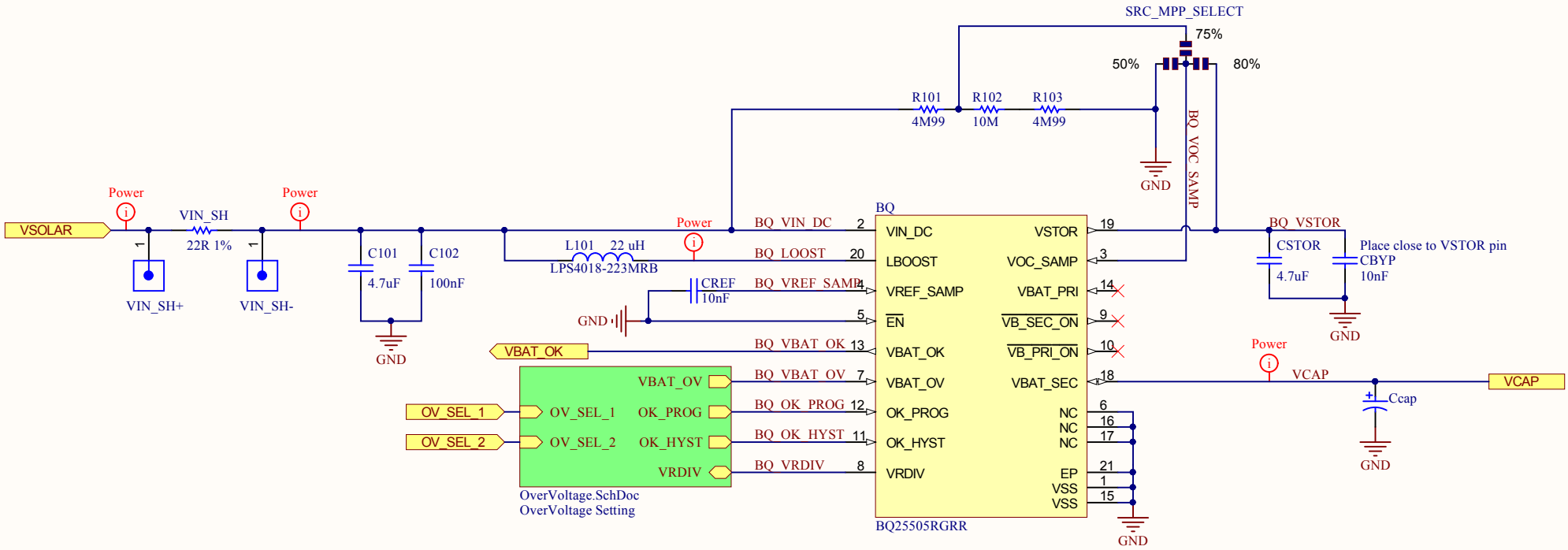
Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Drawing title: **Vision Sensor Prototype**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: *	Project: Prototype.SchDoc
Date: 28.07.2016 11:33:29		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\MA_TransientOpticalFlow\pcb\MSP432_prototype\Prototype.SchDoc			

Energy Management Unit (EMU)



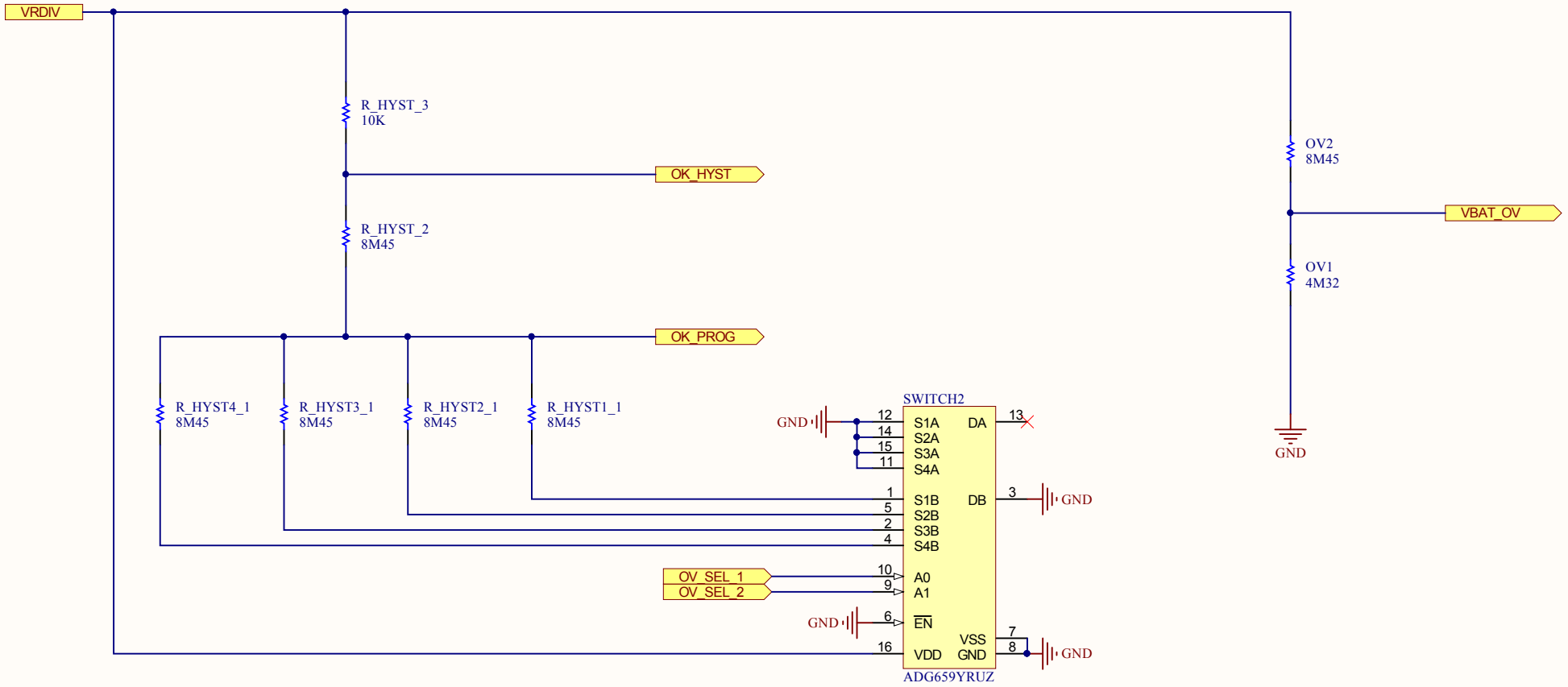
Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Project: **Vision Sensor - EMU**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: ETHZ TIK	Sheet: EMU.SchDoc
Date: 28.07.2016 11:33:29		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\MA_TransientOpticalFlow\pcb\MSP432_prototype\EMU.SchDoc			

BQ Energy Harvesting Subsystem



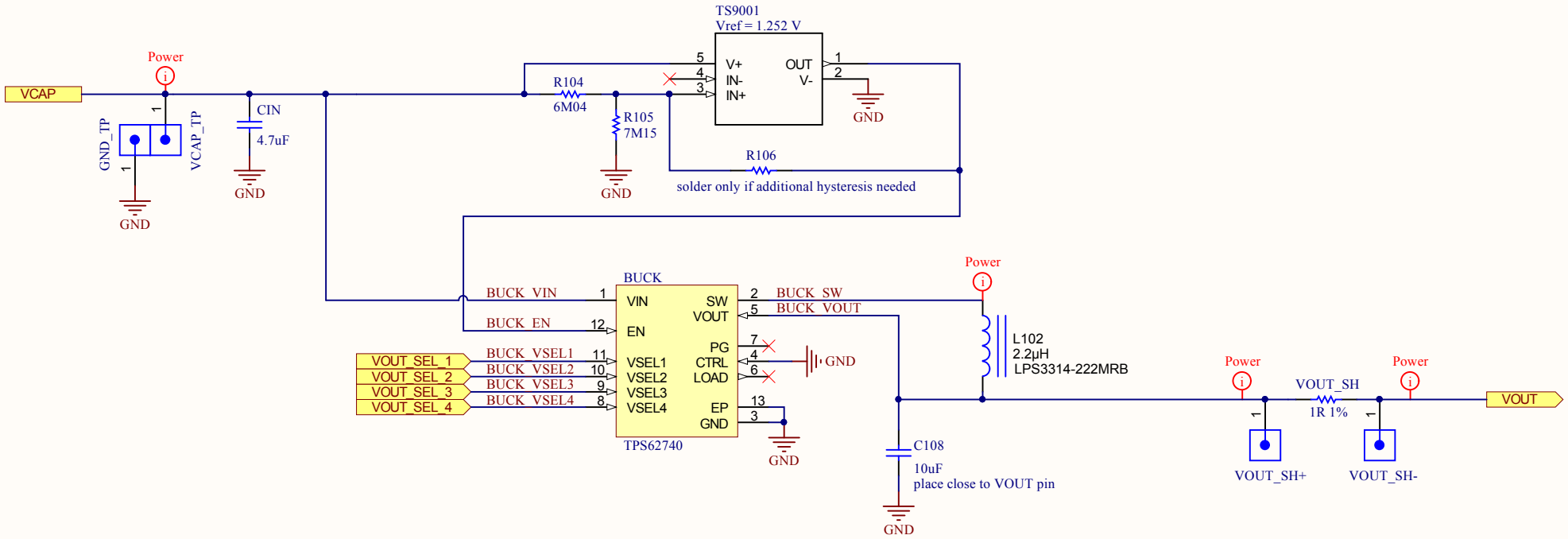
ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Project: **Vision Sensor - EMU - BQ Subsystem**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: ETHZ TIK	Sheet: BQ_Subsystem.SchDoc
Date: 28.07.2016 11:33:28		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\MA_TransientOpticalFlow\pcb\MSP432_prototype\BQ_Subsystem.SchDoc			

BQ Overvoltage and Hysteresis Selection

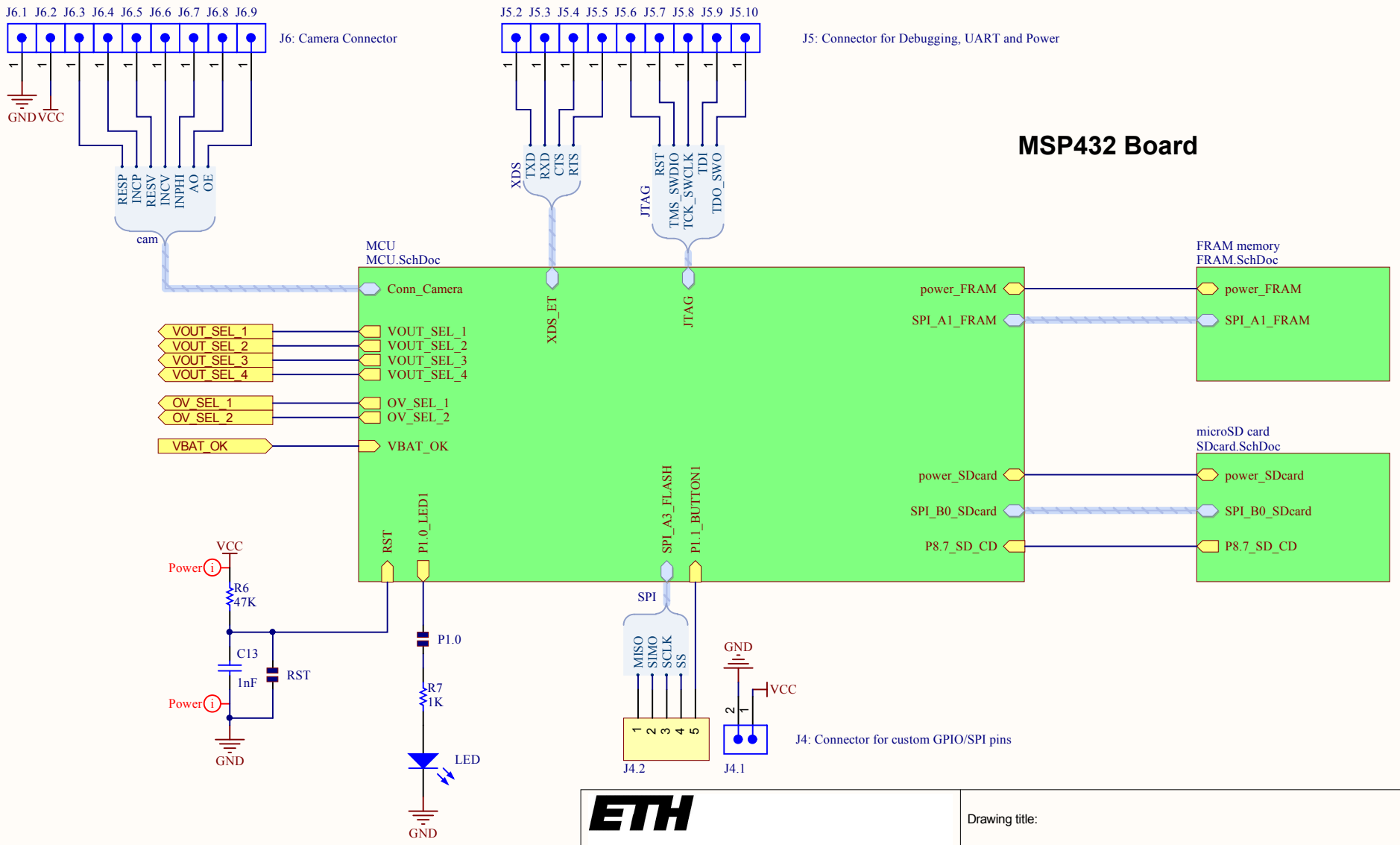


ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Project: **Vision Sensor - EMU - OverVoltage Settings**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: ETHZ TIK	Sheet: OverVoltage.SchDoc
Date: 28.07.2016 11:33:29		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\MA_TransientOpticalFlow\pcb\MSP432_prototype\OverVoltage.SchDoc			

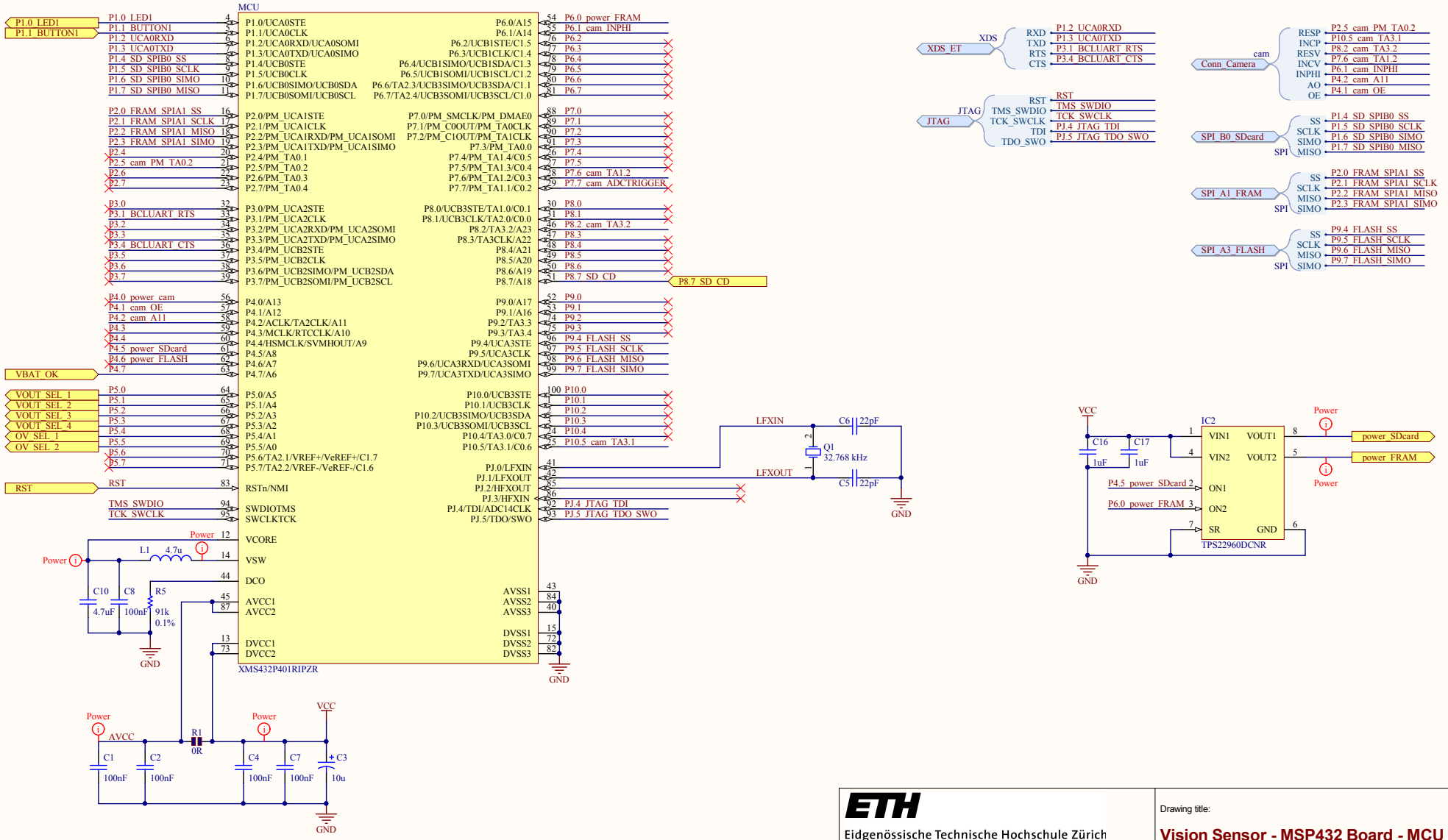
Buck Converter Subsystem



ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Project: **Vision Sensor - EMU - Buck Subsystem**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: ETHZ TIK	Sheet: Buck_Subsystem.SchDoc
Date: 28.07.2016 11:33:29		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\MA_TransientOpticalFlow\pcb\MSP432_prototype\Buck_Subsystem.SchDoc			

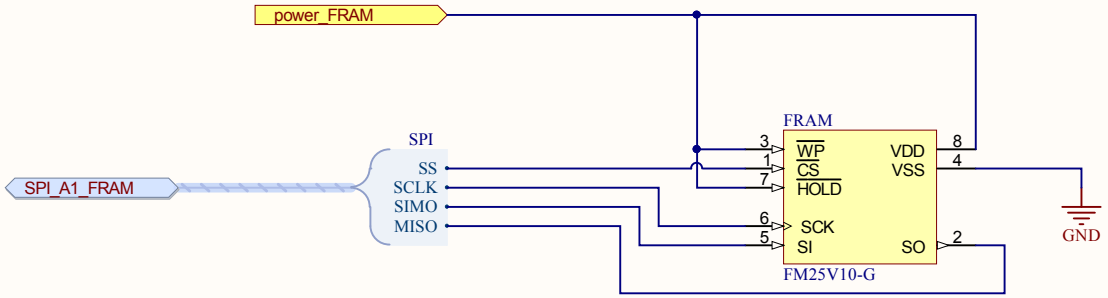

J6.1 J6.2 J6.3 J6.4 J6.5 J6.6 J6.7 J6.8 J6.9
J6: Camera Connector
GND VCC
RESP INCP RESV INCV INPHI AO OE
cam
J5.2 J5.3 J5.4 J5.5 J5.6 J5.7 J5.8 J5.9 J5.10
J5: Connector for Debugging, UART and Power
XDS TXD RXD CTS RTS
JTAG RST TMS_SWDIO TCK_SWCLK TDI TDO_SWO
MSP432 Board
MCU
MCU.SchDoc
Conn_Camera
XDS_ET
JTAG
VOUT_SEL_1
VOUT_SEL_2
VOUT_SEL_3
VOUT_SEL_4
OV_SEL_1
OV_SEL_2
VBAT_OK
RST
P1.0_LED1
SPI_A3_FLASH
P1.1_BUTTON1
power_FRAM
SPI_A1_FRAM
FRAM memory
FRAM.SchDoc
power_SDcard
SPI_B0_SDcard
P8.7_SD_CD
microSD card
SDcard.SchDoc
VCC
Power
R6
47K
C13
1nF
RST
GND
P1.0
R7
1K
LED
SPI
MISO SIMO SCLK SS
J4.2
GND
VCC
J4.1
J4: Connector for custom GPIO/SPI pins
ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich
Drawing title:
Vision Sensor - MSP432 Board
Drawing number: * Rev: *
Date: 28.07.2016 11:33:29
Format: A4 Q
Laboratory: *
Drawn by: Thomas Schalch
Project: MSP432_prototype.SchDoc
Page * of *
File: C:\Users\user\Dropbox\Masterthesis\MA_TransientOpticalFlow\pcb\MSP432_prototype\MSP432_prototype.SchDoc

MSP432 MCU



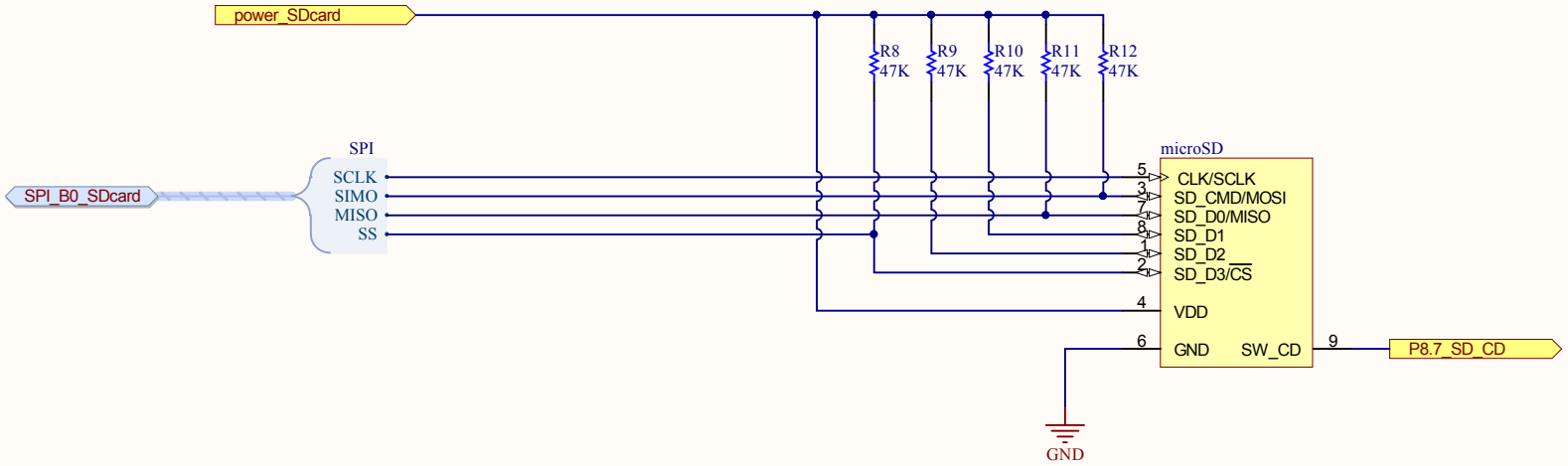
ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich			Drawing title: **Vision Sensor - MSP432 Board - MCU**
Drawing number: * Rev: *	Format: A3	Laboratory: *	Project: MCU.SchDoc
Date: 28.07.2016 11:33:29		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\MA_TransientOpticalFlow\pcb\MSP432_prototype\MCU.SchDoc			

FRAM Memory Chip



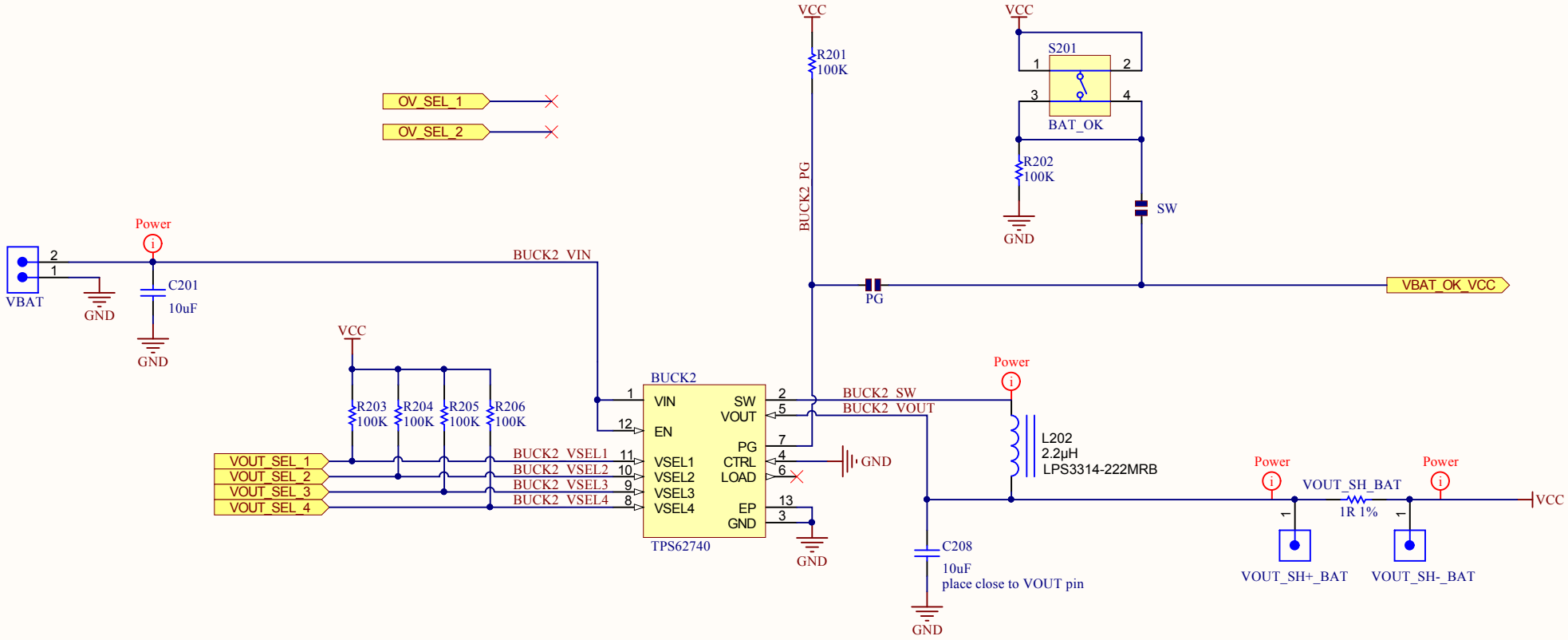
Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Drawing title: **Vision Sensor - MSP432 Baord - FRAM Chip**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: *	Project: FRAM.SchDoc
Date: 28.07.2016 11:33:29		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\MA_TransientOpticalFlow\pcb\MSP432_prototype\FRAM.SchDoc			

microSD card memory



ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Drawing title: **Vision Sensor - MSP432 Board - microSD**		
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: *	Project: SDcard.SchDoc
Date: 28.07.2016 11:33:29		Drawn by: Thomas Schalch	Page * of *
File: C:\Users\user\Dropbox\Masterthesis\MA_TransientOpticalFlow\pcb\MSP432_prototype\SDcard.SchDoc			

Battery Pack and Buck Converter



ETH Eidgenössische Technische Hochschule Zürich Swiss Federal Institute of Technology Zurich	Project: **Vision Sensor - Battery Pack**	
Drawing number: * Rev: *	Format: **A4 Q**	Laboratory: ETHZ TIK
Date: 28.07.2016 11:33:28		Drawn by: Thomas Schalch
File: C:\Users\user\Dropbox\Masterthesis\MA_TransientOpticalFlow\pcb\MSP432_prototype\BatteryPack.SchDoc		

Sheet:	BatteryPack.SchDoc
Page *	of *

Appendix C

Task Characterization

The complete list of all task characterization measurements is provided on the next page. The task characterizations are discussed in Chapter 5.1.

Task Characterization

MSP432 Vision Sensor Prototype

Task	CPU Freq. [MHz]	Vcc [V]	Rshunt [Ohm]	Exp Runtime [s]	Executions	Task Energy [µJ] *			Task Execution Time [ms] *			Application Energy [µJ] **			Application Exec. Time [ms] **		
						Min	Max	Mean	Min	Max	Mean	Min	Max	Mean	Min	Max	Mean
Camera: Acquire two images with on-line FPN compensation	48	3	10	100	1152	919.85	944.48	929.57	53	54	53.09	1250.94	1272.25	1261.78	83	84	83.398
Camera: Acquire two images (no FPN compensation)	24	3	10	100	929	481.19	501.55	488.41	53	54	53.11	767.83	792.74	780.5	103	105	104.24
Camera: Acquire two images, FPN compensation (in sequence)	24	3	10	200	1829	529.17	549.64	537.47	60	62	60.996	825.17	852.67	838.96	105	107	105.93
FPN: Compensation of both buffered images	48	2.2	10	100	636	48.51	62.04	54.84	3	4	3.52	789.03	822.78	807.57	149	157	153.31
FPN: Compensation of both buffered images	24	2.2	10	200	1161	39.68	47.57	45.22	6	7	6.83	677.75	694.92	685.39	167	170	168.43
Processing: Block Matching (search area: ±8,±8, blocksize: 48px)	48	2.2	10	200	883	1413.91	1433.89	1420.59	97	98	97.17	1835.35	1854.24	1843.67	222	223	222.36
Processing: Block Matching (search area: ±8,±3, blocksize: 48px)	48	2.2	10	200	1079	743.49	762.43	757.11	55	56	55.9	1174.21	1190.28	1180.39	181	182	181.14
Processing: Block Matching (search area: ±8,±3, blocksize: 48px)	24	2.2	10	300	1254	630.66	641.84	634.75	110	111	110.01	1045.13	1058.28	1050.24	235	236	235.13
Processing: Block Matching (search area: ±8,±3, blocksize: 48px)	12	2.2	10	400	770	1366.73	1374.86	1370.66	387	388	387.51	1786.09	1797.08	1781.91	514	516	514.97
Processing: Block Matching (search area: -16+0,±3, blocksize: 48px)	48	2.2	10	200	1079	745.26	764.93	759.14	55	56	55.91	1175.94	1192.28	1182.63	181	182	181.15
SD flush: 5 datasets from FRAM to microSD (FAT***), Kingston 8GB C10	48	3	1	400	126	311249.6	416742.7	343582.3	2831	3723	3129.94	311291.2	416780.6	343630.3	2836	3728	3135.3
SD flush: 5 datasets from FRAM to microSD (no FAT), Kingston 8GB C10	48	3	1	400	454	96290.9	165917.1	102629.4	835	1314	869.86	96330.6	165966.7	102677.1	840	1320	875.2
SD flush: 5 datasets from FRAM to microSD (no FAT), Kingston 8GB C10	24	3	1	400	367	104845.3	171910.8	110761.3	1047	1517	1079.75	104864.1	171931.1	110791.8	1052	1521	1084.45
SD flush: 5 datasets from FRAM to microSD (no FAT), SanDisk 2GB	48	3	1	400	1134	12677	28589.8	13533.6	339	455	343.09	12739.9	28632.9	13579.5	344	461	348.49
SD flush: 5 datasets from FRAM to microSD (no FAT), Kingston 2GB	48	3	1	400	841	16483.3	86836.9	36155.1	330	801	465.9	16522.8	86897.6	36206.7	339	806	471.32
SD flush: 5 datasets from FRAM to microSD (no FAT), Samsung 8GB C2	48	3	1	400	422	38160.2	216093.3	85746.5	524	2075	934.8	38225.7	216139.3	85793.3	528	2081	940.29
SD write: 1 dataset from internal SRAM to microSD (FAT***), Kingston 8GB C10	48	3	1	400	341	124979.9	213886.8	131800.6	989	1612	1037.08	125539.3	214560.2	132379.3	1115	1739	1163.29
SD write: 1 dataset from internal SRAM to microSD (no FAT), Kingston 8GB C10	48	3	1	400	541	78995.4	144659	83162	586	1048	607.07	79543.1	145233.4	83722.3	713	1174	733.3
SD write: 1 dataset from internal SRAM to microSD (no FAT), SanDisk 2GB	48	3	1	400	1739	6003.5	22529.7	6684.4	96	230	99.33	6550.2	23095.6	7240.8	222	357	225.96

* purely task-related figures; overheads like state retention activities are not considered

** figures related to complete application; including task execution, state retention, changes of power regulators/clocks etc.; startup CPU frequency is 12MHz for all experiments

*** using the FAT file system, the energy and time figures heavily depend on the prehistory of the microSD Card