Design of a Touch Screen Using Only an LED Matrix

DA674

Diploma Thesis

at

Graz University of Technology

Submitted by

Christoph Hönck

Institute for Electronics (IFE)
Graz University of Technology
A-8010 Graz, Austria

Head: Univ.-Prof. Dipl.-Ing. Dr.techn. Wolfgang Pribyl

Advisors: Ass.Prof. Dipl.-Ing. Dr.techn. Peter Söser
Dipl.-Ing. Peter Trattler (austriamicrosystems AG)

Graz, March 2011

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources and that I have explicitly marked all material which has been quoted either literally or by content from the used sources.

…… ……..
Date (Signature)

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

…… ……..
Datum (Unterschrift)

Abstract

The present thesis explores the opportunity of ordinary Light-emitting Diodes (LED) acting as a light-sensitive sensor and how to use this ability to build a new kind of touch screen. LEDs are normally used to emit light, but they are also able to detect light by producing a photocurrent, depending on the ambient light, similar to photodiodes. LEDs can be arranged in form of a matrix to appear like a screen with low resolution which is able to show simple patterns. In this constellation it is possible to use some of the LEDs as a light source while other LEDs act as light-sensitive sensors. An object or even a finger coming near the surface of the matrix reflects its emitted light and the detecting LEDs can identify this as a touch. The crux of using this technique in the field of touch interfaces is to find an algorithm which empowers the screen to emit and detect light with the LEDs at the same time. Switches drive the LED matrix, following a defined algorithm. In contrast to existing algorithms, the invented algorithm is able to achieve these requirements. While the LED matrix shows a pattern the LEDs simultaneously sense the incident light. Another demand the algorithm is to handle these challenges in different ambient light conditions. The invented algorithm reduces the influence of ambient light to a minimum. In summary it can be said that this technology, combined with the invented algorithm, is able to build a touch screen using only ordinary LEDs. To demonstrate the abilities of the invented and related algorithms this thesis comprises the development of an embodiment, containing a six by six LED matrix used as a touch screen and a Graphical User Interface (GUI) running on a PC to make the results visible.

Die vorliegende Arbeit beschäftigt sich mit der Möglichkeit, herkömmliche Leuchtdioden als lichtempfindliche Sensoren zu verwenden und damit eine neue Art von Touchscreen zu definieren. Leuchtdioden werden in erster Linie dazu verwendet, Licht abzustrahlen; sie können aber auch, abhängig vom eingestrahlten Licht, einen Photostrom produzieren, ähnlich wie es bei Photodioden genutzt wird. Werden LEDs in Matrixform angeordnet, bilden sie einen Bildschirm mit geringer Auflösung, der einfache Muster darstellen kann. In einem solchen Aufbau wird es möglich LEDs zur Darstellung zu verwenden, während andere LEDs als lichtempfindliche Sensoren agieren. Wenn sich ein Objekt oder ein Finger der Matrix-Oberfläche nähert, wird das abgestrahlte Licht reflektiert. Die zur Detektion verwendeten LEDs können die Änderung des eingestrahlten Lichts als Berührung erkennen. Die Krux dabei liegt darin, einen Algorithmus zu finden, der es erlaubt, zur selben Zeit sowohl Muster darzustellen, als auch Berührungen zu erkennen. Schalter steuern die LED-Matrix an und müssen einem Algorithmus folgend arbeiten. Im Gegensatz zu bereits verwendeten Algorithmen ist der neu erarbeitete Algorithmus in der Lage, die Vorgaben des gleichzeitigen Darstellens von Mustern und Detektierens von Berührungen zu erfüllen. Zusätzlich wird vom Algorithmus gefordert, unter verschiedenen Lichtverhältnissen zu arbeiten und den Einfluss vom Umgebungslicht möglichst gering zu halten. Zusammengefasst kann gesagt werden, dass gezeigt wird, wie ein berührungsempfindlicher Bildschirm aus einfachen LEDs gebildet und angesteuert werden kann. Um die Fähigkeiten des entwickelten und bereits verwendeter Algorithmen zu demonstrieren, umfasst die Arbeit auch die Entwicklung eines Prototypen, bestehend aus seiner sechs mal sechs LED Matrix, die als Touchscreen verwendet wird, und einer graphischen Benutzeroberfläche am PC, die die Ergebnisse sichtbar macht.

Contents

List of Figures

List of Abbreviations

A	area
AC	alternating current
ADC	analog-to-digital converter
AlGaAs	aluminium gallium arsenide
A_{NOISE}	noise gain
A_{OL}	open loop gain
c	speed of light in free space
CB	conducion band
CD	junction capacitance
C_{Dep}	depletion layer capacitance
C_{Diff}	diffusion capacitance
C_F	feedback capacitor
CHx	channel x
C_i	total input capacitance
C_{icm}	common-mode input capacitance
C_{id}	opamp differential input capacitance
CMOS	comlementary metal oxid semiconductor
C_p	package capacitance
C_S	stray capacitance
D	duty cycle
DAC	digital-to-analog converter
DC	direct current
e	electronic charge
E	energy
E	illumination level
E_c	conduction band edge energy
LDO	low-drop regulator
LED	light-emitting diode
L_p	diffusion length of holes
LSB	least significant bit
MUX	multiplexer
N_a	acceptor dopant concentration
n_{column}	number of columns
N_d	donor dopant concentration
n_{row}	number of rows
Opamp	operational amplifier
PCB	printed circuit board
PLL	phase-locked loop
PMOS	p-channel metal-oxide semiconductor
p_{n0}	hole concentration in n-type material
QT	Quasar Technologies
R	flux responsivity
RC	resistor-capacitor oscillator
R_D	diode resistance
R_F	feedback resistor
R_{off}	off restistance
R_{on}	on resistance
R_S	serial resistance
r_S	diode resistance
S/H	sample and hold
SAR	successive approximation register
Si	silicon

E_g	bandgap energy
en	input noise voltage
ESD	electrostatic discharge
E_v	valance band edge energy
F.S.	full scale
f_p	pole frequency
FRC	fast resistor-capacitor oscillator
f_z	zero frequency
GaAs	gallium arsenide
GaAsP	gallium arsenide phosphide
GaInN	gallium indium nitrogen
Ge	germanium
GND	ground
GUI	graphical user interface
h	Planck's constant
HID	human interface device
HSV	hue saturation value
I	applied current
I/O	input/output
I_{ADJ}	adjustment current
I_B	opamp input bias current
I_D	diode current
I_L	photocurrent
I_{LED}	LED supply current
in	input noise current
I_S	reverse saturation current
k	wavevector
k_B	Boltzmann constant
SMD	surface-mounted device
SPDT	single pole double throw
T	temperature in Kelvin
T_{AD}	conversion time periode
T_{CY}	instruction cylce time periode
T_{SMP}	sampling time
USB	universal serial bus
USB-IF	USB implementers forum
v	velocity
V	applied voltage
V-	negative supply voltage
V+	positive supply voltage
V_0	built-in voltage
VB	valance band
V_D	diffusion voltage
V_{DD}	supply voltage
V_F	forward voltage
V_{IN}	input voltage
V_{Off}	offset voltage
V_{OUT}	output voltage
V_{Ref}	reference voltage
V_{th}	threshold voltage
W	width
ε	permetivity of a medium
η	quantum efficiency
η_{ideal}	diode ideality factor
λ	free space wavelength
µC	microcontroller

Acknowledgements

The present master thesis was written in association with austriamicrosystems AG who I want to thank for the professional working sphere they kindly provided to me and the sincere respect for my work. I was able to work without any pressure to succeed, but always with the goal in mind. A special thanks to my responsible advisor, Peter Trattler, for his patience explaining me the details of the necessary electronic knowledge and for giving me fruitful suggestions to find new approaches dealing with this novel field of application. Furthermore for his always positively put objections whenever I was enthusiastic for a solution too soon and for keeping me from banging my head against a brick wall. Apart from my advisor I am indebted to a lot of other colleagues in the company who helped with words and deeds. I cannot even number all of them but whether at the coffee machine or even at lunch I always found a sympathetic ear listening to any of my requests. It was a pleasure being a part of the team.

Of course I wish to thank my advisor on the part of the university, Peter Söser, too. He offered me a climate open for discussion and was always available for my requests - not even the weekend could stop him from answering back to me very soon.

Last but not least, I owe my family and friends gratitude for tolerating my tempers, for accepting my unavailability and for proofreading my thesis over and over again. Without all of your support this thesis would not have worked out so well.

Christoph Hönck
Graz, Austria, March 2011

1 Introduction

The present Master Thesis handed in at the Graz University of Technology is authored in association with austriamicrosystems AG. Apart from scientific reasoning, also practical reflection for laying a foundation for prospective products are the motivating forces to do research in this field of electronics. Over a duration of six month research was done in the laboratories of austriamicrosystems AG and after that the achieved findings and deliverables were embedded in the necessary theoretical substructure of this thesis.

This thesis describes the use of ordinary LEDs as light-sensitive sensors and beyond that the thesis tries to answer the question how to implement this technique in the field of touch screens. Established on the use of LED matrices, the present work invents a novel kind of algorithm enabling to show patterns and detect light in one and the same LED matrix simultaneously.

First of all the thesis gets to the bottom of the fundamental structure of LEDs, explaining its attributes and parasitic components and showing its commonalities with photodiodes on the basis of theoretical literature survey. It confirms that an LED can be used as a sensor and is completed with an overview of related work in this field of application. The second chapter covers the development of the embodiments' hardware and its critical parts. The current-to-voltage converter is explained in special detail and builds the core of chapter 3. The subsequent chapter 4 embraces the whole software and the underlying logical connections as well as a detailed description of the invented algorithm. This chapter also discusses the implementation of the software along with its technical realisation both on the microcontroller and on the level of the PC. Toward the end of the thesis, chapter 5 summarizes the evaluated results of the present work and underlines them with necessary measurement results. Finally, chapter 6 concludes the thesis with a short summary of the project status and an outlook on future trends.

Appendix A completes the thesis with related schematics and layouts of the present work.

1.1 Motivation

The company austriamicrosystems AG has announced this master thesis topic officially at the Institute for Electronics, for an estimated duration of six month with the following rough tasks and schedule.

Phase 1 (2 months):

- Design of a measurement setup using an existing LED matrix

- Measurements with reflected light (one or more LEDs emit light and measure reflected light e.g. by a human finger depending on the distance of the finger)

- And emission (ambient light directly shines or casts a shadow on the LED matrix)

Phase 2 (2 months):

- Design of a test system which allows verification of different algorithms for detection

- Development of an algorithm to detect a touch: reliability of the detection with different levels of ambient light and allowing to show any LED pattern on the LED matrix (time multiplex is allowed)

- Optional: support multi-touch

Phase 3 (2 months):

- Documentation

- Optional (depending on progress of previous steps): design of a demonstration board

Another important target of the thesis is to spot the demands for an integrated solution of the acquired embodiment. The thesis should give an idea where the key challenges for a possible integrated chip solution are and which parameters would be most important for a satisfactorily operation.

1.2 Applications

The field of applications is still very wide and different. The main use case which is also demonstrated with the resulting embodiment of this thesis is the use as a touch screen formed by ordinary LEDs. Not only a screen built of an LED matrix can use this technology. All other forms of LED arrangements and even a pair of LEDs can use the explained findings. Nowadays LEDs are commonly used in a huge number of devices and therefore detection in form of the present findings can be implemented without any high additional costs. Mobile phones, portable music players, televisions or even household appliances - nearly every device with some kind of user interaction could be a potential field of application. Some examples for implementation could be:

- A single LED row acting as a controller, for example for volume as intensity.

- Two LEDs replace a mechanical switch.

- A handheld device recognizes a user's grip via LED detection, giving also feedback to it.

- Replacement of different capacity sensors, particularly when LEDs are already available in the device.

- And much more…

2 Theoretical Background

This chapter discusses the theoretical background which is the basis of the resulting findings and conclusions. First of all it is important to know how and why an LED works and especially under which conditions an LED is able to detect light. The contemplation of the fundamental structure of the LED and a comparison with the structure of photodiodes explains why an LED can act as light sensitive sensor. This chapter further discusses the parasitic components of an LED and its interaction with other circuit parts as well as the behaviour of the LEDs in case of a matrix structure. Moreover the present chapter gives a summary of related exploitations of LEDs as sensors. The chapter is written on the basis of a wide literature survey. A very detailed description of different kinds of LEDs found in [SCHUBERT1] as well as the physical basics of semiconductors in [SZE1] help to build up this theoretical chapter. Additionally, a very helpful discussion about optoelectronic devices and their practical application are given in [KASAP1] and [STINY1].

2.1 Attributes of LEDs

LEDs are semiconductors which emit light. This is caused by a radiative recombination of electrons and holes in their p-n junction (luminescence). The unbiased p-n junction has a region near the p-n junction bare of free carriers called depletion region or space charge region. Without free carriers the only charge in this region depends on ionized donors and acceptors. A potential difference with a diffusion voltage V_D arises. This voltage can be seen as a barrier for free carriers which have to overcome to get in the opposite neutral region.

When an LED is forward-biased, minority carriers are injected in the opposite region and recombine, thereby emitting photons. The current through the LED strongly increases when the diode's forward voltage reaches the diffusion voltage. The voltage level is for this reason so-called threshold voltage.

To accomplish an efficient radiative recombination it is advised to use a direct-bandgap semiconductor. An indirect-bandgap semiconductor with isoelectronic impurities could also be used, but in addition to photons also phonons would be generated and would reduce the efficiency. In Figure 2.1 both a direct recombination and an indirect recombination are shown.

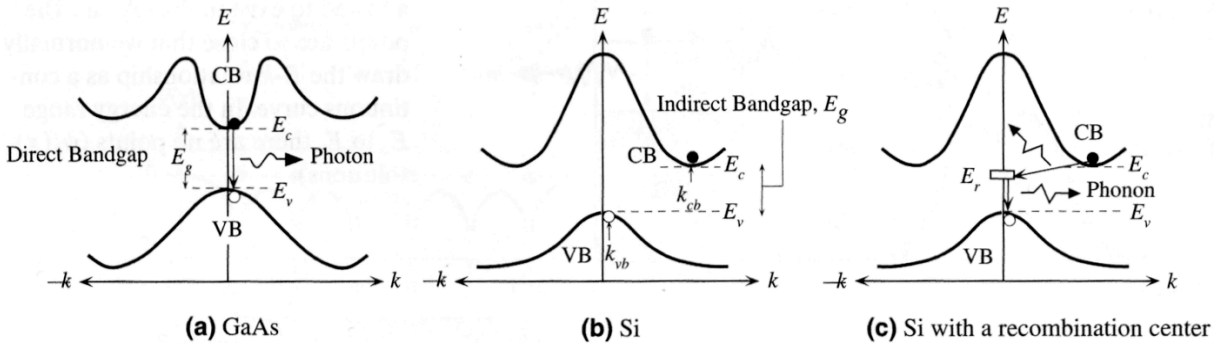


Figure 2.1: (a) direct bandgap (b) indirect bandgap (c) indirect bandgap with impurities
[Figure extracted from [KASAP1, p.122]]

2.1.1 Energy and Wavelength of the Emitted Light

The diffusion voltage of the semiconductor material is related to the energy gap of the semiconductor material. Equation (1) expresses this connection mathematically.

$$V_{th} \approx V_D \approx E_g / e \tag{1}$$

In Figure 2.2 the current-voltage diagram for different semiconductor materials are shown. One can see that the correlation between the diffusion voltage V_D and the energy gap E_g is different, depending on the semiconductor material of the p-n junction.

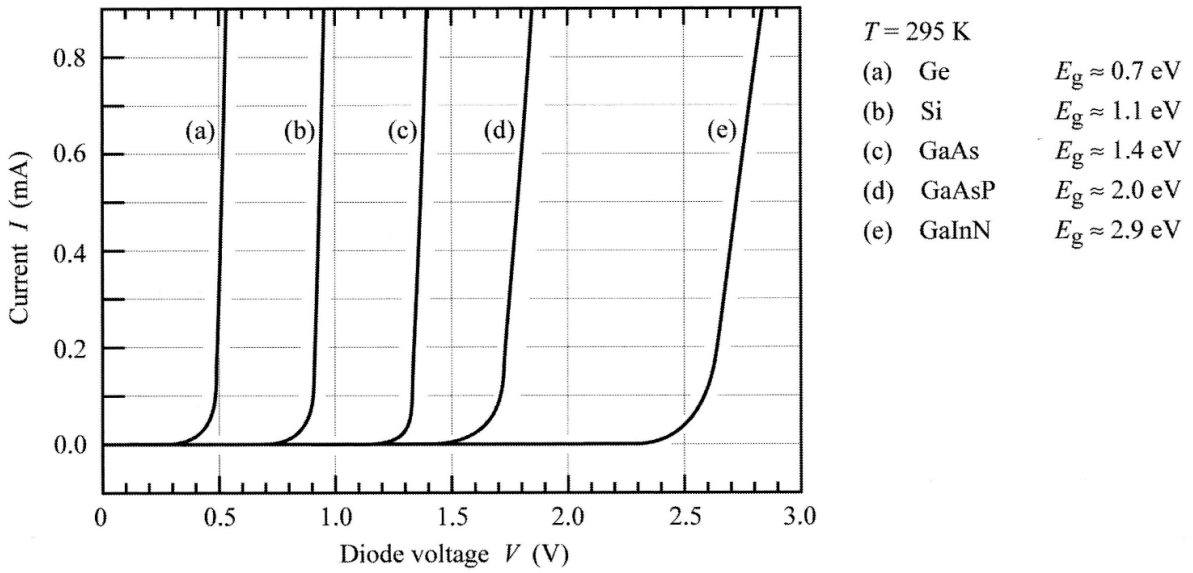


Figure 2.2: Current-voltage characteristics of p-n junctions at different semiconductors
[Figure extracted from [SCHUBERT1, p.62] with kind permission of Cambridge University Press]

The energy of the emitted photons depends on the energy gap of the semiconductor material between the conduction and the valence band. As visible in Equation (2), the velocity of the photons v multiplied by the Planck constant equals the energy gap. One can draw the conclusion that a higher energy gap increases the velocity of the photons and therefore decreases their wavelength. In the best case every injected electron recombines in the p-n junction of the LED and generates a photon. This case leads to energy preservation as shown in Equation (3).

$$hv = E_g \tag{2}$$

$$eV = hv \tag{3}$$

The wavelength of the emitted light complies with the wavelength of the emitted photons and is therefore related to the energy gap of the semiconductor. Equation (4) shows how the emitted wavelength corresponds to the bandgap of the material. In Figure 2.3 the relationship between the two properties is shown graphically.

$$\lambda[nm] = \frac{h * c}{E_g} = \frac{1240}{E_g} \quad \ldots E_g[eV] \tag{4}$$

In the following, one will see that the energy and therewith the wavelength of the emitted light plays a major role in exciting semiconductor materials to generate a photocurrent.

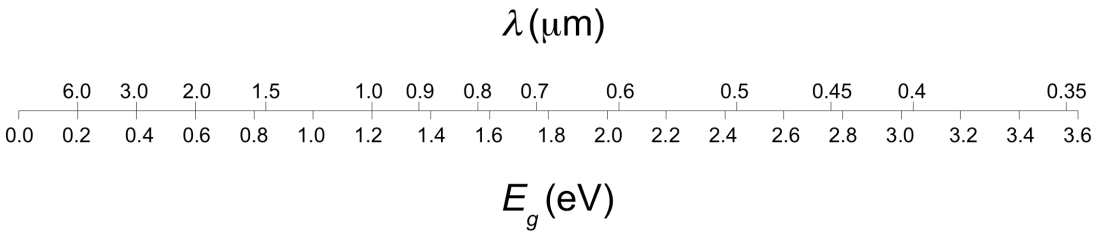


Figure 2.3: Wavelength and bandgap relationship in semiconductors
[Figure adopted from [SINGH1, p.460]]

2.1.2 Capacitance

Another important property of an LED is its capacitance. The key part of an LED is its p-n junction which is also mainly responsible for the capacitance of an LED. Two different capacitances should be contemplated: depletion layer capacitance and diffusion capacitance.

- **Depletion Layer Capacitance C_{dep}**

In the space charge region there are positive carriers on one side and negative carriers on the other, separated by the distance *W*. It could be seen as a dielectric parallel plate capacitor with the width of the depletion region and its area, following the relationship of Equation (5).

$$C_{dep} = \varepsilon * \frac{A}{W} \tag{5}$$

A change in the applied voltage affects the width of the depletion region. As the dependence between voltage and depletion region, the depletion layer capacitance also depends on the applied voltage. Equation (6) shows the voltage-dependants of the depletion region width.

$$W = \sqrt{\frac{2 \bullet \varepsilon \bullet \left(N_a + N_d\right) \bullet \left(V_0 - V\right)}{e \bullet N_a \bullet N_d}} \tag{6}$$

Apart from the applied voltage, the doping of the semiconductor, its doping profile and the area of the p-n junction are responsible for the depletion capacitance. Equation (7) exactly describes the depletion layer capacitance. As one can see, a reverse bias ($V = -V_r$) applied to the p-n junction decreases the capacitance, and an unbiased p-n junction causes the largest possible depletion layer capacity following the equation.

$$C_{dep} = \frac{A}{\sqrt{V_0 - V}} * \sqrt{\frac{e \bullet \varepsilon \bullet \left(N_a \bullet N_d\right)}{2 \bullet \left(N_a + N_d\right)}} \tag{7}$$

If the bias voltage $V \rightarrow V_0$ the equation diverges and will be complex for higher values of V. This does not correspond to the real behaviour. Under forward conditions the approximation cannot be used. In simulation tools like SPICE it is common to make a linear extrapolation for higher values of V. Furthermore the diffusion capacitance is much higher than the depletion capacitance under forward conditions. Typically values for the depletion layer capacitance of an unbiased p-n junction are smaller than 100 pF.

- **Diffusion Capacitance C_{diff}**

In reverse biased conditions the depletion layer capacitance is dominant while in forward biased conditions the depletion capacitance is already present but the diffusion capacitance is much larger.

Under forward biased conditions the width of the space charge region is reduced and with it the potential barrier for the carriers. Now it is much more likely that free majority carriers diffuse into the opposite neutral region. An excess of minority carriers occurs and allowing them to recombine with the majority carriers. At the same time new carriers must be supplied from the voltage source to let the regions stay neutral. The voltage-depending change of carrier density is expressed by the diffusion capacitance.

$$C_{diff} = \frac{e^2}{k \bullet T} \bullet A \bullet L_p \bullet p_{n0} \bullet e^{\frac{e \bullet V_F}{k_B \bullet T}} \tag{8}$$

Equation (8) shows apart from the dependence of size and material properties the diffusion capacitance increases exponentially with the forward voltage. Under reverse biased conditions the diffusion capacitance disappears.

2.1.3 Equivalent Circuit

For a better understanding of the behaviour of an LED in electronic circuits it is necessary to know an accurate equivalent circuit of it. It is possible to take into account the parasitic properties and know well how it will collaborate with other parts of the circuit.

Again the p-n junction is crucial for the structure of the equivalent circuit. The main parts are the diode resistance *rs* (given by the I-V characteristics), the serial resistance of the neutral n- and p-regions *Rs* and its capacity. As mentioned above, in reverse biased conditions the diffusion capacitance C_{diff} disappears and only the depletion layer capacitance C_{dep} is essential. Additionally a capacitance related to the diode packaging C_P is taken into account in the equivalent circuit. For the subsequent chapters it is not necessary to maintain the three

different parasitic capacitances in detail, hence a single parasitic capacitance C_D will cover the elemental effects of all of them.

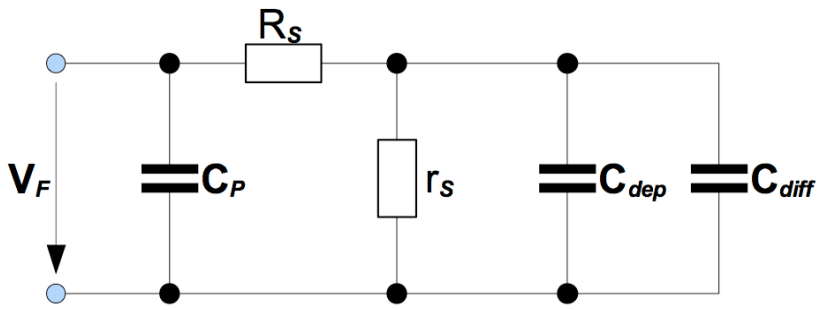


Figure 2.4: The equivalent circuit of a forward-biased diode
[Figure adopted from [SINGH1 p.200]]

The effects of the parasitic resistors R_S and r_S can be seen in the I-V characteristic of the LED. The theoretical behaviour of the p-n junction without any parasites is given by the Shockley equation. The following Shockley Equation (9) includes an ideality factor η_{ideal}. This factor has the value one for an ideal p-n junction and can assume higher values for real junctions.

$$I = I_S \bullet e^{\frac{e \bullet V}{\left(\eta_{ideal} \bullet k_B \bullet T\right)}} \tag{9}$$

To consider the parasitic resistances in the I-V characteristic the Shockley equation has to be modified to get a connection like Equation (10).

$$I = \frac{(V - I \bullet R_S)}{r_S} = I_S \bullet e^{\frac{e \bullet (V - I \bullet R_S)}{\left(\eta_{ideal} \bullet k_B \bullet T\right)}} \tag{10}$$

As one can see, if $R_S \to 0$ and $r_S \to \infty$ the equation reduces back to its general form. Figure 2.5 shows the effect of the parasitic resistances graphically. The reasons for a series resistance could be the contact resistance or the resistance of the neutral regions. However, for the parallel resistance any channel that avoids the p-n junction is responsible. A series resistance can flatten the exponential growth of the I-V diagram and a parallel resistance which divides the current between the LED and the resistor following Kirchhoff's current law could affect the turn-on voltage.

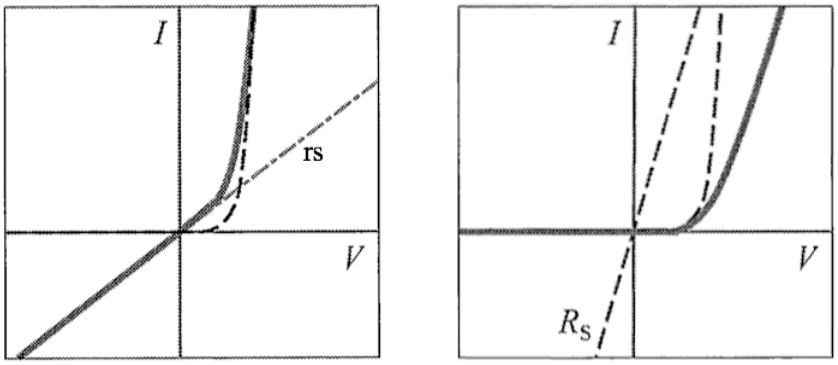


Figure 2.5: Effect of parasitic resistors in the I-V diagram
[Figure extracted from [SCHUBERT1, p.66] with kind permission of Cambridge University Press]

2.1.4 LED Device Structure

The recombination process occurred in a forward biased LED causes emissions of photons in random directions. The structure of an LED must take this into account so that the photons can escape the component without being absorbed or reflected by the semiconductor material. A very good enumeration of the different device structures and the appropriate explanations are found in [KASAP1] and are roughly elaborated.

The simplest form of fabricating an LED is by epitaxial growth of layers on a substrate (e.g. GaAs). First the n-layer and then a narrow p-layer of a few microns (to avoid absorption of the photons) are formed. A heavily doped n-side secures that most of the recombination happens in the p-side. Additionally the p-side can be built by diffusing dopants into the epitaxial n-layer. This type of junction between two doped semiconductors out of the same material is called a homojunction. It is important to take care of the lattice-match between the LED layers and the substrate crystal to avoid lattice strains. If not it could lead to crystal defects and boost the radiationless recombination of electron-hole pairs. In Figure 2.6 the two possibilities of planar LED structures are shown. Due to the internal reflections still not all light rays are able to escape.

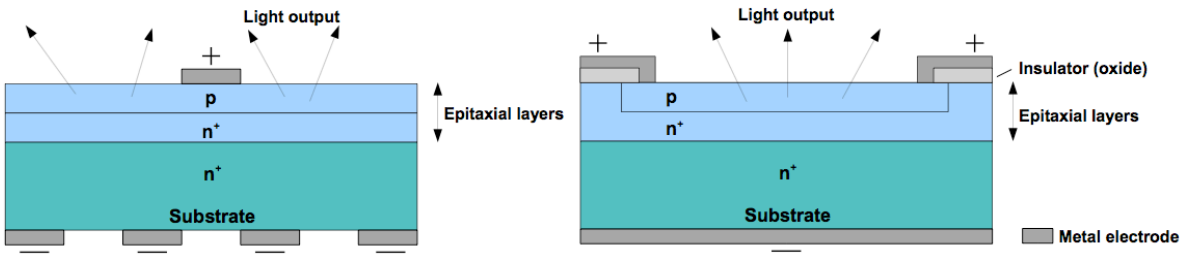


Figure 2.6: Typical planar surface emitting LED devices
[Figure adapted from [KASAP1, p.141]]

Beside homojunctions also heterojunctions which are much more efficient are more and more used in LEDs to increase the light output. Heterojunctions consist of two semiconductor materials with different bandgaps. LEDs which use this type of junction are called heterostructure devices. In contrast to heterojunction LEDs, homojunction LEDs have the disadvantage of a narrow p-layer which allows some of the injected electrons of the p-side to diffuse to the surface and recombine through crystal defects without radiation. This results in

a reduction of the light output and an abated efficiency. As an example a double-heterostructure device is shown in Figure 2.7, this means even two junctions of different semiconductor materials of different bandgaps are used.

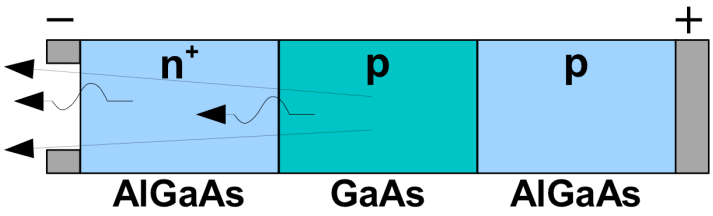


Figure 2.7: A double heterostructure diode
[Figure adapted from [KASAP1, p.146]]

The semiconductor material GaAs in the middle of the arrangement has a lower bandgap ($E_g \approx$ 1.4 eV) than the different doped AlGaAs materials ($E_g \approx 2$ eV) on the outsides. If the diode is under forward biased condition, the active region is flooded with carriers and electron hole pairs recombine by emitting photons. Because of the higher bandgap of AlGaAs compared to GaAs the emitted light is not reabsorbed by the n-layer.

2.2 LED as Sensor

A parasitic effect of LEDs is that an LED produces a small amount of photocurrent when light impinges on its surface, as described in [SCHUBERT1]. This can be seen by measuring the I-V characteristic of the LED under the influence of light. In the present use case this behaviour is actually a desired effect. It is a physical fact which is the foundation of the invented embodiment.

Figure 2.8 shows the I-V characteristic of an LED under the influence of light. The diagram has a magnified negative current scale to highlight the effect of a generated photocurrent more clearly. The dashed line corresponds to the ideal I-V characteristic given by the Shockley equation.

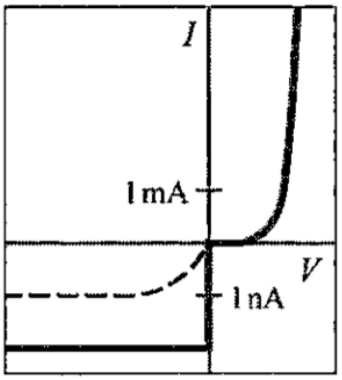


Figure 2.8: I-V characteristic of an LED under the influence of light
[Figure extracted from [SCHUBERT1, p.65] with kind permission of Cambridge University Press]

2.2.1 A Single LED as Sensor

To understand why an LED produces a small amount of photocurrent it is necessary to know how a photodiode works and how to elaborate its similarities to an LED. In its first chapter [GRAEME1] explains the abilities and attributes of a photodiode in a very understandable way. Both LEDs and photodiodes are based on the same physical principles but photodiodes used to apply exactly the opposite effect of LEDs. Instead of emitting photons a photodiode's goal is to absorb photons and generate a photocurrent.

Based on the principles of the inner photo effect incident light on a direct bandgap semiconductor can force an electron to move from the valence band into the conduction band. As a result free carriers are generated and can lead to a radiation-dependent current. In semiconductors without a direct bandgap photons can only be absorbed with the help of phonons which reduce the absorption coefficient by approximately one hundredth. The different forms of electron excitations in direct and indirect semiconductors are shown in Figure 2.9. Similar to the LED the direct semiconductors provide a far better efficiency.

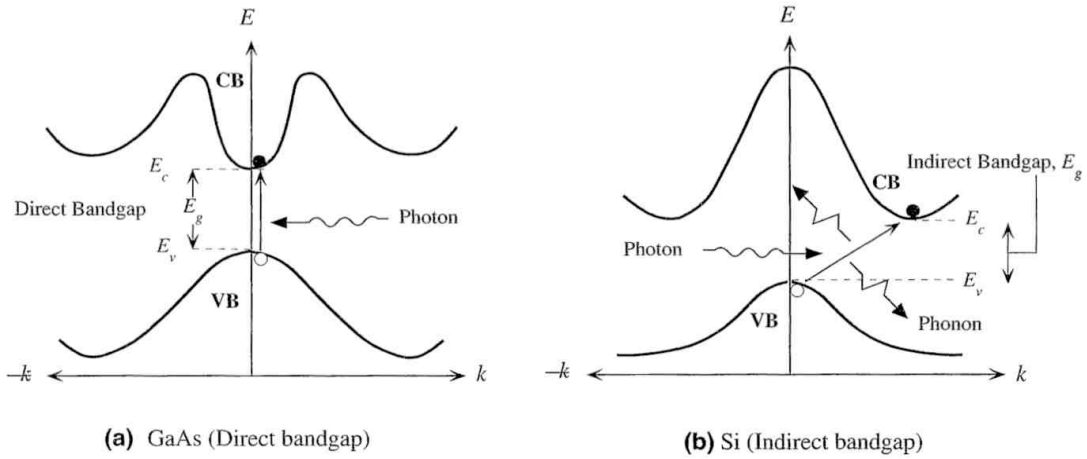


Figure 2.9: Photon absorption in direct and indirect semiconductors
[Figure extracted from [KASAP1, p.223]]

If the wavelength and thus the energy of the incident light are suitable photons are absorbed in the depletion zone of the p-n junction. The acting electric field inside the depletion zone forces the holes into the neutral region of the p-layer and the electrons to the neutral region of the n-layer. If the photodiode is connected to a closed circuit the generated carriers contribute to a current flow, a so-called photocurrent.

Figure 2.10 exhibits the fundamental processes in a photodiode when photons are absorbed. The released carriers in the depletion region are accelerated by the electric field, carriers outside just travel by diffusion or recombine quickly.

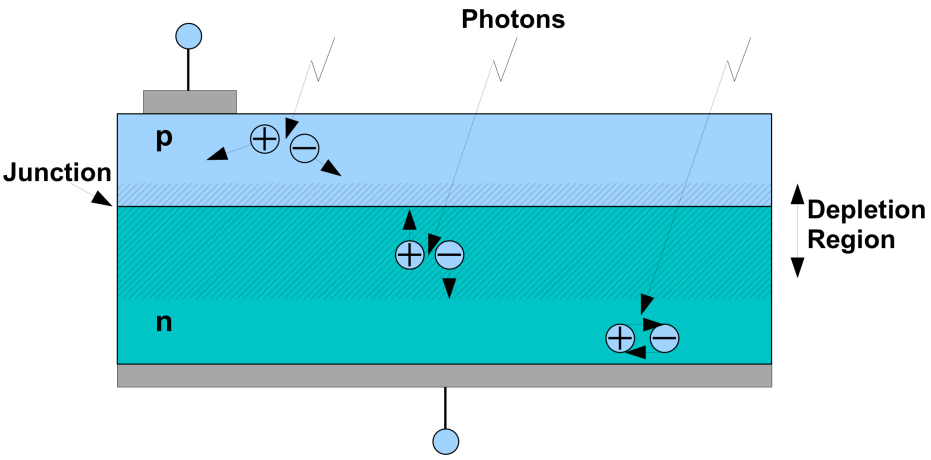


Figure 2.10: Absorption of photons in a semiconductor
[Figure adopted from [GRAEME1, p.2]]

Not every incident light is able to produce a photocurrent. The wavelength depends on the used semiconductor material of the photodiode. Therefore only light in a special wavelength range is capable of generating free carriers. The upper cut-off wavelength is given by the energy gap of the semiconductor, as it appears in Equation (4). The shorter the wavelength of the incident light, the higher its energy. It can roughly be said that light with a too long wavelength has not enough energy to dissolve new carriers out of the semiconductors valence band and lifts them up into the conduction band. Hence, light with a too short wavelength does not reach useful absorption because carriers recombine near the surface due to surface defects and do not contribute to the photocurrent.

For LEDs this means that only light with the same wavelength or shorter can be detected. As an example an LED that emits green light ($\lambda_{green} \approx 550$ nm) cannot detect red light ($\lambda_{red} \approx 650$ nm). However, a red LED is be able to detect blue light ($\lambda_{blue} \approx 450$ nm).

Compared with Figure 2.8 a photodiode has a much more pronounced I-V characteristic with respect to the photocurrent. It can also be seen as a diode characteristic with an offset depending on the illumination of the photodiode. Figure 2.11 shows the measured I-V characteristic of a photodiode and its different areas of work.

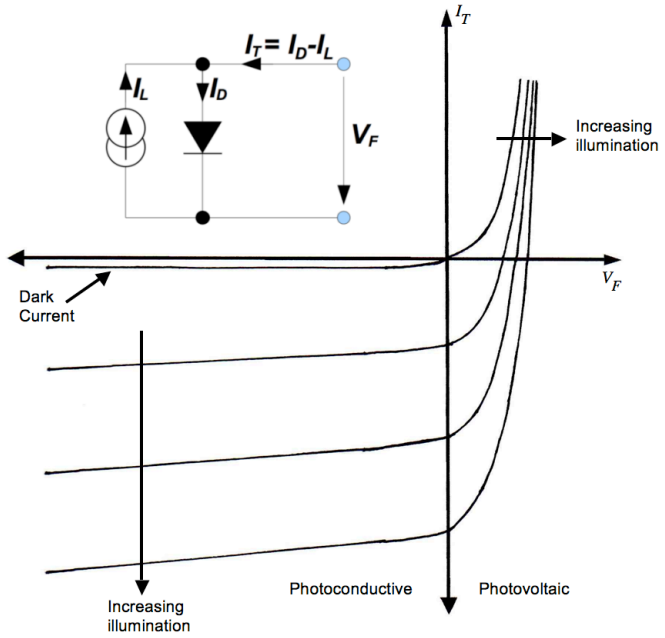


Figure 2.11: I-V characteristic of a photodiode with different levels of incident light
[Figure adapted from [GRAEME1, p.6]]

In case of an applied voltage $V_F < 0$ V the photodiode works in photoconductive mode as a consumer, also called current-output region. The measured current is nearly linear to the power of illumination and the photocurrent i_L flows in the same direction as the dark current i_D. If the photodiode works in the fourth quadrant of the diagram, where $V_F > 0$ V forward biases the diode, it works in photovoltaic mode. In this mode the photodiode acts as producer of electrical energy. The photocurrent and the dark current flow in opposite directions. Without an external voltage source of $V_F = 0$ V the photodiode's operation mode is called short circuit operation mode. With a small load resistance and the absence of an applied voltage, over an order of magnitude of eight decimal powers, a linear photocurrent depending on the illumination can be monitored. Especially in the light measurement the short circuit operation mode is the preferred one and is therefore used in the present embodiment.

Similar to the LED the equivalent circuit of a photodiode consists of a parallel resistor R_D, a serial resistor R_S and a capacitance C_D depending on the applied voltage. Additionally a photocurrent source is mentioned depending on the incident light and an ideal diode represents the diode behaviour (Figure 2.12). The current source is also present in an LED when light strikes its surface. The photocurrent flows in the same direction as the reverse current of the diode.

It can be concluded that the LED and the photodiode are abstracted by the same equivalent circuit although of course the current source does not generate the same amount of current. The device structure of an ordinary p-n photodiode is in some parts similar to the structure of a homojunction LED (Figure 2.6). In both cases it is necessary to build the p-n junction with a narrow p-layer. Avoid reflections and absorption of the emitted light in case of an LED and to allow photons to enter easily the depletion zone in the field of application as a photodiode.

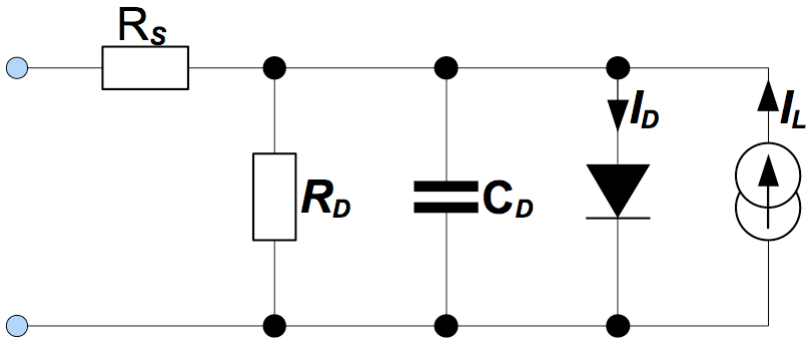


Figure 2.12: Equivalent circuit of a photodiode
[Figure adapted from [GRAEME1, p.5]]

Due to the different device structure on account of its field of application the ability of LEDs to absorb photons is not as good as the ability of photodiodes but they are still able to do so. Characteristic values of the parts' efficiency to convert the energy of photons into an electrical signal are, for example, the quantum efficiency η and the flux responsivity R. The quantum efficiency represents how much of the incident photons actually contribute to the photocurrent. Then again the flux responsivity R represents the parts' ability of generating a photocurrent out of incident optical power based on a special wavelength.

$$\eta = \frac{Number\ of\ \ free\ carriers\ generated\ and\ collected}{Number\ of\ incident\ \ photons} \tag{11}$$

$$R = \frac{Photocurrent\ [A]}{Incident\ Optical\ Power\ [W]} \tag{12}$$

The characteristic values described above are a lot smaller for LEDs than for photodiodes. For the following Chapters it can be postulated that an LED behaves like a photodiode but with a worse conversion of light leading to a significantly smaller photocurrent.

2.2.2 An LED Matrix as Sensor

In this thesis, it is important to know how an LED acts as a photosensitive sensor but it is also essential to consider that the upcoming use case needs the interaction of a whole matrix of LEDs which is used as sensors. The use of a matrix-array is very useful in this field of application but brings some difficulties with it. In Figure 2.13a it can be seen that the cathodes of each column and the anodes of each row are connected.

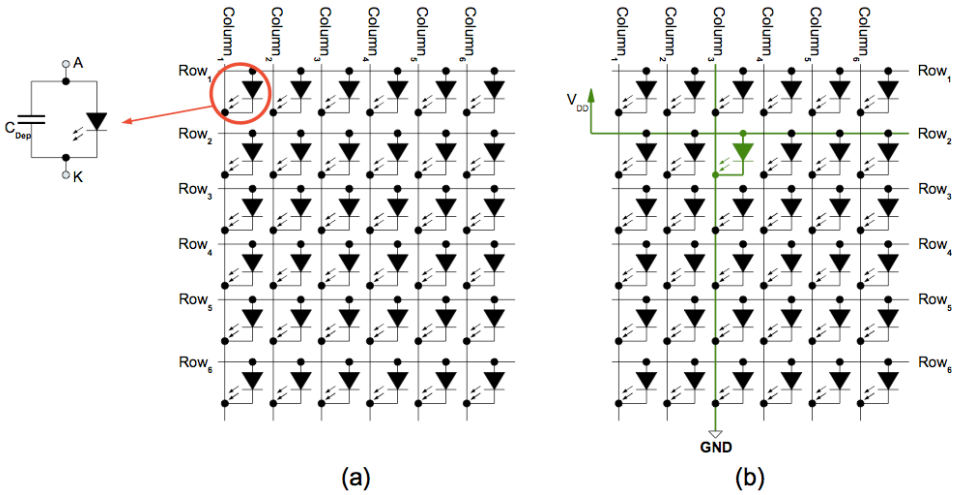


Figure 2.13: (a) LED Matrix (b) Forward bias one LED of the matrix

The connection of the rows' anodes and the columns' cathodes facilitate the capacitive coupling in the matrix. The enlarged section of Figure 2.13a stresses how the parasitic capacitance of each LED appears in the matrix array. Outwardly a much greater capacitance is perceived than a single LED would have. Forward biasing an LED in the matrix means to set the corresponding row high and the associated column to ground (Figure 2.13b). To supply a forward voltage and a connection to ground to each LED, the rows and the columns are normally connected to switches and are controlled by a microcontroller. Because of the matrix arrangement only LEDs of one column are able to emit light (and so connected to ground) at the same time. If two or more columns are connected to ground the forward voltage, supplied by each row, is divided and no appropriate current flows to achieve the desired brightness. Owing to the fact that the LED forward voltages differ a little bit the LEDs can even be damaged.

To obtain a static pattern on the matrix array it is necessary to switch fast enough from column to column so that the human eye does not recognize a flicker. An LED forced by a frequency of 60 Hz is sufficient to create the illusion of a stationary pattern. Apart from the discussed disadvantages by using an LED matrix a big advantage can be registered. The LED provides additional light conditions. The detailed explanation can be found in sub chapter 3.2.

2.3 Related Exploitation of using an LED as Sensor

The idea to use an LED as an optical sensor was presented to a large public in 1970 by Forrest M. Mims III [MIMS1]. For a long time the principle seemed to be forgotten but several technicians took up the idea again. In literature a lot of papers and patents can be found about this topic. In the following an exemplary list of some patents related to LEDs used as optical sensors is introduced.

- DORN Alfred: *Control Device Using Light-Emitting Diodes for Both Manual Input and Display Data*. U.S. Patent 4,692,739, Sep. 8, 1987.

- NAKAMURA Yukio et al.: *Control Circuit for Array of Light-Emitting Diodes*. U.S. Patent 5,424,855, Jun. 13, 1995.

- DIETZ Paul H.: *Automatic Backlight for Handheld Device*. U.S. Patent 6,664,744 B2, Dec. 16, 2003.

- DIETZ Paul H. et al.: *LED With Controlled Capacitive Discharge for Photo Sensing.* U.S. Patent 6,870,148 B2, Mar. 22, 2005

- HAN Jefferson Y.: *Multi-Touch Sensing Light Emitting Diode Display and Method for Using the Same*. U.S. Patent 7,598,949 B2, Oct. 6, 2009.

There are also some prior patents and papers which define the fundamental use of photosensitive devices to detect light, wherever reflected light or the shade of light. Some of the publications are similar in some parts but also deal with different kind of uses. Summarized it results in roughly two types of reactions: On the one hand there is the idea of using the junction capacitance of the LED (indirect method) and on the other hand the output current or voltage of an LED can be monitored (direct method) and evaluated.

2.3.1 Indirect Method (Using the LED Capacitance)

The crux of the indirect method is to take advantage of the parasitic junction capacitance of the LED. This technique is already used in some applications and some patents like [DIETZ1] explain the method in detail. As described in subchapter 2.1.2, every LED has a parallel parasitic capacitance. Normally an LED is used to emit light often related to the output of a microcontroller (Figure 2.14a). The resistor limits the current because of the exponential I-V characteristic of the LED.

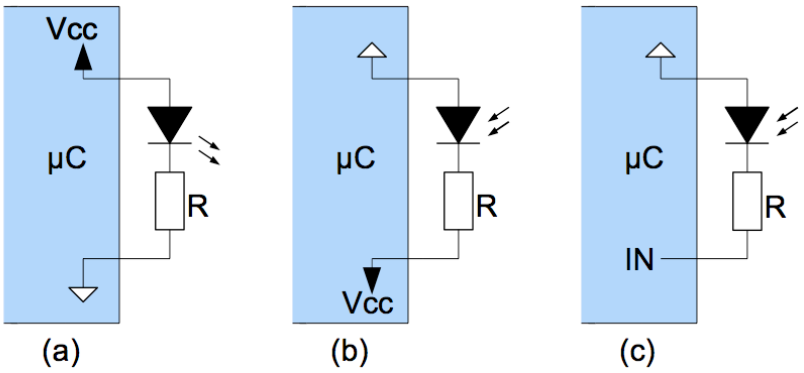


Figure 2.14: LED as emitter and detector by using its junction capacitance
[Figure adapted from [DIETZ1]]

If the LED is supposed to act as a photosensitive sensor in this method the first thing to do is to connect the anode of the LED to ground and the cathode of the LED to a CMOS I/O pin of the microcontroller driven high (Figure 2.14b). A reverse current flows through the diode and charges the junction capacitance. The next step is to switch the CMOS I/O pin at the anode of the LED to input mode (Figure 2.14c). The photocurrent will discharge the capacitance until the voltage at the CMOS input reaches the digital threshold. The more light is incident on the LED, the shorter the discharge time of the junction capacitance will be because of the resulting increase in photocurrent. If the time during the discharging of the capacitance is measured, it can be taken as an index related to the amount of incident light.

If the incident light is low, the time to discharge the capacitance can be very long. This disadvantage of the indirect method makes it impossible to implement real-time applications and high data rate communications.

2.3.2 Direct Method (Monitoring the LED Voltage or Current)

As the name suggests the direct method is able to measure one of the direct output signals of the LED. This can be either the voltage or the current produced by the photoelectric effect in the LED's p-n junction. The current monitoring is the better alternative because of a far better linearity, a wider bandwidth and less dc offset. In this field of application a standard circuit which is also used with ordinary photodiodes is the monitoring of the output current with a current-to-voltage converter (Figure 2.15). Of course there are many other approaches for using the output signals of an LED in a direct way but this schematic is well described in the literature, has good linearity and is therefore widely used.

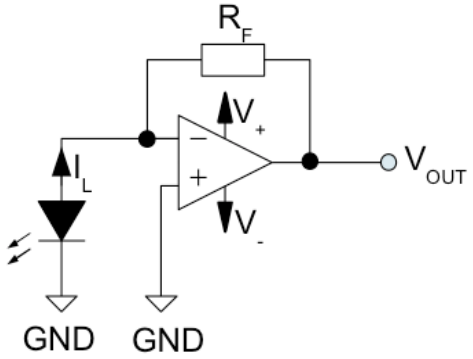


Figure 2.15: LED as sensor with a current-to-voltage converter

According to the schematic in Figure 2.15 an LED is connected to the negative input of an operational amplifier. A generated photocurrent I_L flows over the feedback resistor R_F and causes a negative voltage V_{OUT} at the output of the operational amplifier. The relationship of the generated photocurrent and the output voltage of the operational amplifier are explained by the Equation (13).

$$V_{OUT} = -I_L * R_F \tag{13}$$

In one of the subsequent chapters the current-to-voltage converter is described in more detail because the invented embodiment also uses this kind of schematic for the purpose of getting useful signals. At this point of the thesis, discussing previous methods, no more detailed knowledge is needed.

2.4 Related Algorithms

In addition to using different methods of evaluating the signals of an LED caused by the inner photo effect there are also different approaches of how to configure a matrix of LEDs to emit and detect light alternately. Moreover the algorithm which drives the LEDs and sets their configuration is responsible for the sensing resolution and the maximum size of the matrix. The subsequent discussed algorithms are based on the patent of Jefferson Han [HAN1] which was filed 2009.

2.4.1 Han Even/Odd with Single Colour LEDs

Based on the principles of earlier patents using an LED as emitter and detector (today it can be seen as prior art) Jefferson Y. Han invented an algorithm using those principles in LED matrices, too. To monitor the LEDs the direct method is used and the evaluated output parameter of each LED is the generated photocurrent.

For an easier understanding of the algorithm it should be detained what the matrix configuration looks like. The algorithm uses the same LED matrix configuration as pictured in Figure 2.13. All cathodes of each column and all anodes of each row are interconnected. Additionally every row is in series with an amplifier to acquire the output signal of the LEDs. Every row and every column gets an independent switch to connect the cathodes to ground and the anodes to the supply voltage, driven by a controller.

The algorithm itself controls the switches which surround the LED matrix. All of the matrix LEDs are captured in three steps per column. First of all, a column is connected to ground. After this connection is set, the even rows are connected to its supply voltage. Now the generated photocurrent of each amplifier, connected with an even row, is sampled. The same procedure follows for the reverse case in which the even LEDs emit light and the odd ones detect. Finally all LEDs are set to detect light and their photocurrent is sampled. This continues until the last column of the matrix is measured under all three conditions. Figure 4.3a and Figure 4.4 show the timeline of the algorithm graphically. A significant disadvantage of the algorithm is that the matrix cannot be used to display patterns at the same time. Further details of the algorithm are discussed in the upcoming subchapter 4.1.1.

2.4.2 Han Even/Odd with Multicolour LEDs

In case of multicolour LEDs each colour in the same LED package must have an independent anode connection. Principally the algorithm acts like the single-colour algorithm but the distinction between even and odd LEDs is not necessary anymore. The multicolour LEDs can use different colours instead. For example a matrix with multicolour LEDs with the colours blue and red can use the red ones to detect and the blue ones to emit light. This allows a faster stepping through the matrix. A flowchart of the algorithm is shown in Figure 4.3b. The faster the algorithm completes the matrix, the brighter the emitted light of the matrix seems. A detailed explanation can be found in subchapter 4.1.1.

3 Hardware Realisation

The first thing which is needed when implementing a touch screen out of ordinary LEDs is a hardware system that provides the possibilities to evaluate the optical input signals captured by the LEDs. Whether reflected light, shading or other lighting conditions are used, it is always a challenge to convert the optical input signal into an appropriate electrical signal, regardless of different environmental conditions. After a stable electrical signal is available it has to be digitized and processed to create the possibility for the software to perform the desired calculations and make the results visible for the user. To cope with all these tasks the three main parts LED matrix, amplifier chain and microcontroller have to be designed well and match together. This chapter focuses on the particular parts of the hardware and its dimensioning as well as its interaction.

3.1 Block Diagram of the System

To get an expedient overview of the embodiment's hardware Figure 3.1 shows the block diagram of the system. As mentioned in the introduction of this chapter the main parts of the hardware are the LED matrix with its surrounding switches, the amplifier chain and the microcontroller.

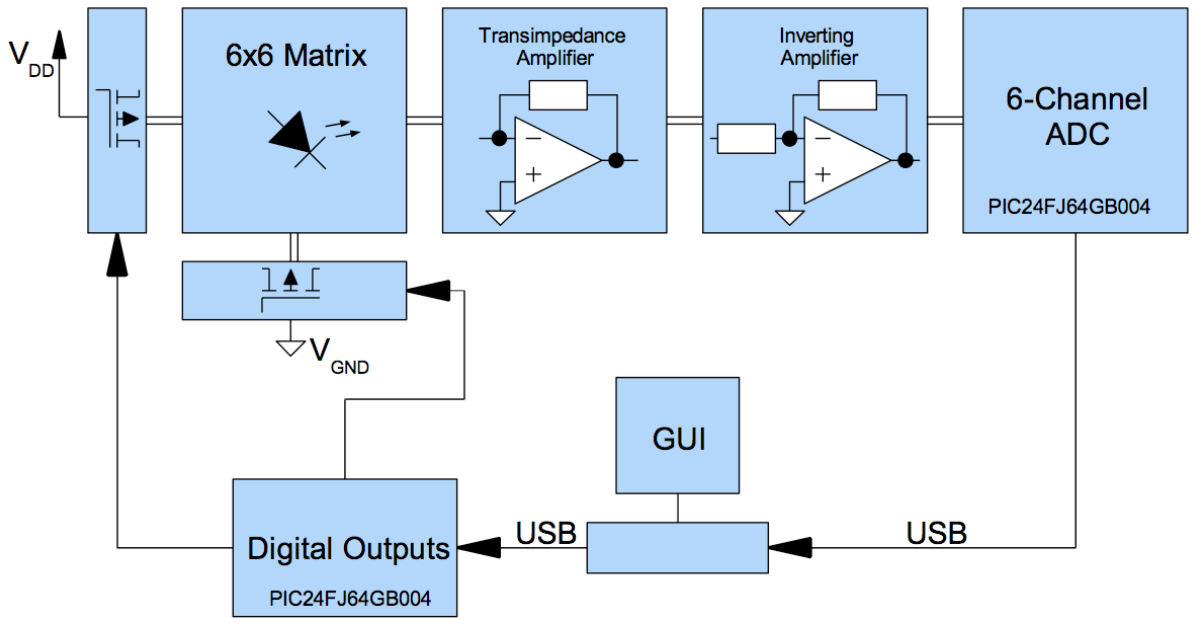


Figure 3.1: Block diagram of the system

The first core element is the LED matrix. In the present work it is a 6x6 matrix, therefore 36 single LEDs. These photosensitive components are responsible for generating an electrical

signal out of the incident light according to the user's gestures. In detail the generated photocurrent is further processed. The amplifier chain, consisting of two steps, is the next element which gets the generated electrical signal. The first stage is a transimpedance amplifier, also known as current-to-voltage converter followed by an inverting amplifier to amplify the output voltage of the previous amplifier stage. The voltage signal, now located in a suitable range, has to be digitized. The microcontroller fetches the signal and performs the analog to digital conversion plus the further processing of the apprehend data. This means that a defined algorithm running on the microcontroller does some calculations and forwards the results to a personal computer (PC). Not to forget that the microcontroller and with it the running algorithm are also responsible for the control of the matrix by setting the switches of the columns and rows at defined times. The PC itself does some corrective improvements with the available data and shows the adjusted results on its screen in a Graphical User Interface (GUI).

The basis of all listed circuit parts are a stable voltage supply. The system on PCB has the advantage of getting an input voltage of 5 V directly from the connected USB Host (PC). To benefit from this voltage source all other supplies are generated out of it. This leads to a self-contained system without any additional supply. Figure 3.2 shows an overview of the supply voltage segmentation. The subsequent chapters explain each of the important system parts in detail and also the according voltage conversion. Nevertheless the PCB provides the option to replace the USB supply by an external voltage supply of 5 V. This may help in case of some board evaluations or measurements.

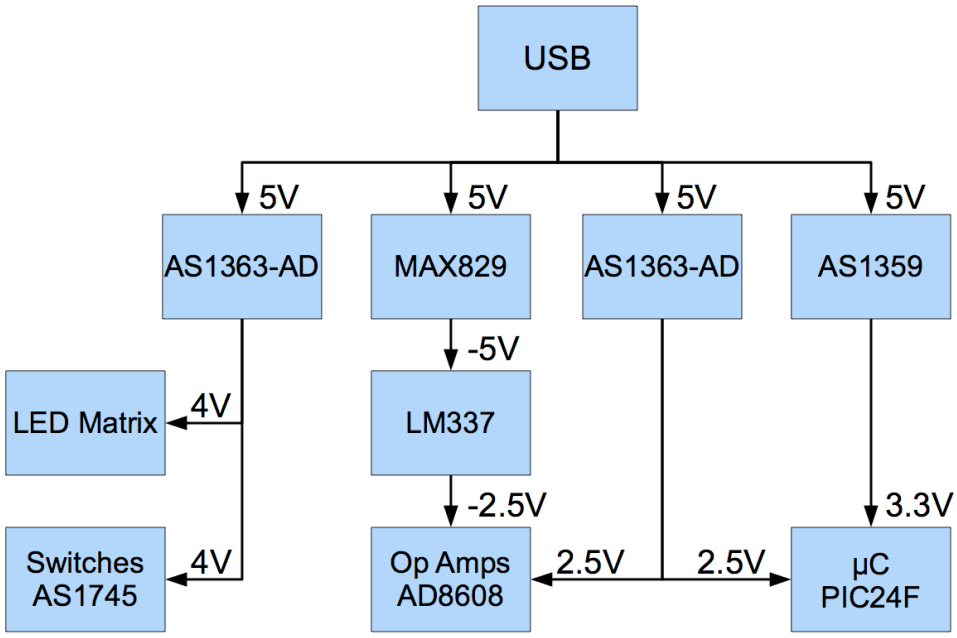


Figure 3.2: Supply voltage segmentation

3.2 The Optical Input Signals

The interpretation of the optical input signals is essential for the whole system. It is also essential to define which optical inputs should be interpreted and which environmental light conditions have to be excluded.

Based on the assumptions in the theoretical part of this thesis it is known that every incident light with an appropriate wavelength is detected by the LEDs but not all of them are interesting for the use case. When a finger or an object comes close to an LED or even touches it, different lighting effects occur. The first possibility when none of the neighbor LEDs in the matrix emits light is that the finger or the object shades the LED from environmental light (Figure 3.3a). As a result the photocurrent drops. Another possible lighting effect is the reflection and the scattering of light emitted by one or more neighbors of the detecting LED, by a finger or an object (Figure 3.3b). The creation of this reflection is the big advantage if a matrix is used instead of a single LED. The impinging of the reflected light results in an increase of the photocurrent, at least in comparison to the shade.

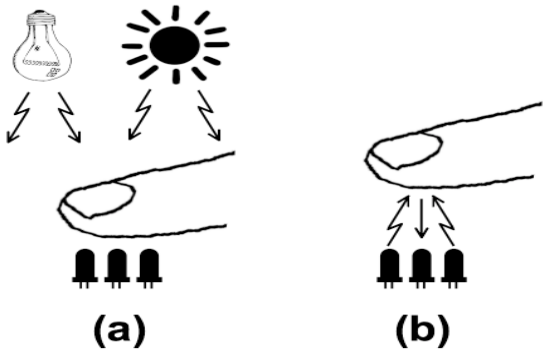


Figure 3.3: The different lighting effects (a) shading (b) reflection and scattering

In the upcoming chapters, where the used algorithms are discussed, some problems produced by different lighting conditions will play a major role. These problems are related to the change of environmental light conditions like very bright environmental light or artificial light incidents on the matrix. One can imagine that it cannot be tolerated if a touch is detected only because of changing light conditions. To handle these problems the algorithm has to do calculations and drive the right switching sequences for the matrix. The hardware itself could just forward signals whether in good or bad environmental conditions. In the detailed description of the algorithms and their challenges in subunit 4.1 it is explained how the algorithms try to overcome this problem.

3.3 Signal Flow

The block diagram gives a rough idea of how the incident light signal is converted and forced through the circuit until it can be interpreted and pictured accordingly. Figure 3.4 is a more detailed description, focusing on the physical meaning of the signals.

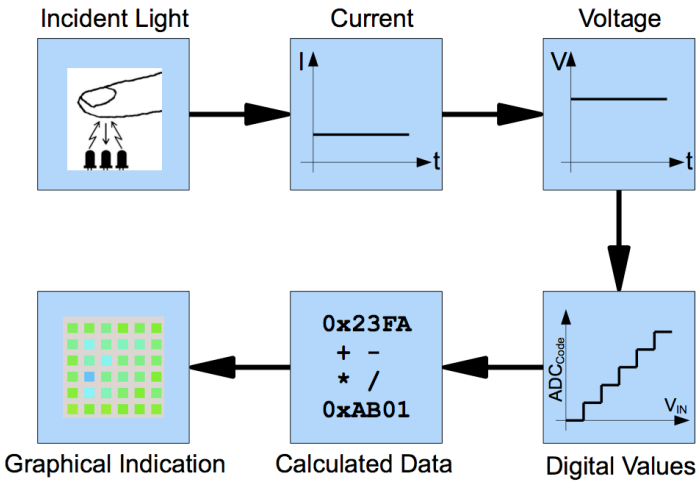


Figure 3.4: Signal flow - from incident light to a graphical output

The first task is to transform the light signal into an electrical signal. The responsible parts for this process are the LEDs. Their output signal is a light-depending photocurrent. Then again this photocurrent has to be converted into a voltage by a transimpedance amplifier. A second amplifier stage amplifies the voltage signal to a more useful area of operation for the analog-to-digital converter (ADC). The electrical signal (now present as analog voltage) will be digitalised and saved in the microcontroller. Now these digital values are available for calculations and analyses of the implemented algorithm in the microcontroller. To make the results accessible for a user another transmission to an ordinary PC which runs the GUI-software occurs. Finally the data is shown on the screen.

3.4 Switches

The switches are used for two types of connections on the PCB of the invented embodiment. The LED matrix provides one connection per row (anodes) and one connection per column (cathodes). Each row has to be connected via a switch to the supply voltage for the LEDs and also fixed permanently to one of the transimpedance amplifier inputs. Each column needs to be connected via a switch to ground. The logic input of each switch has to be connected to an I/O-port of the microcontroller.

Important criteria for switches are a low on resistance R_{on} as well as a high off resistance R_{off} and a low capacitance. A good disconnection of the LED cathodes to ground is especially important for the present embodiment. Not only R_{off} has to be considered, also internal ESD diodes of the switches can lead to an imperfect disconnection if the voltage which has to be switched forces a leakage current over the diodes.

The present embodiment uses the switch *AS1745* from *austriamicrosystems*. It is a CMOS SPDT analog switch powered with a supply voltage of 4 V. Signals over the full supply voltage are able to pass and the I/O-ports of the used microcontroller can drive its logic input. Further details can be found in [AMS1].

The supply voltage of 4 V is transformed over a low-drop regulator (LDO) from the 5 V USB bus voltage (Figure 3.2). To downsize the USB bus voltage the LDO *AS1363-AD* from *austriamicrosystems* [AMS2] is used. To achieve an output voltage of 4 V an external voltage divider has to be connected at the output of the LDO as explained in Figure 3.5. The LDO

uses a voltage feedback as an input (SET pin) for an internal error amplifier to control the gate voltage of a PMOS which regulates the output of the LDO. Additionally capacitors against ground on the in- and output of the LDO are required to guarantee a stable operation. An output capacitor with low ESR has to be used to achieve an optimum output noise and stability characteristics.

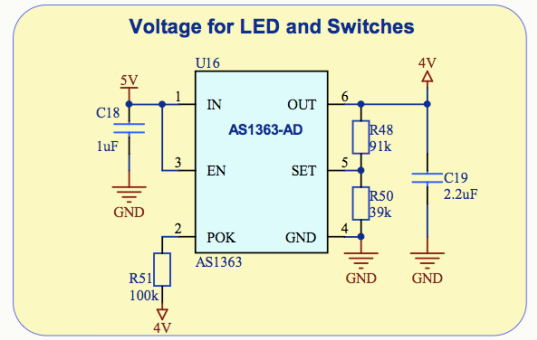


Figure 3.5: Voltage supply for the switches and the LEDs

The POK pin forces the shutdown logic of the chip and is not used. Therefore the pull-up resistor R_{51} provides a continuous function. The resistors R_{48} and R_{50} are responsible for the adjustment of the output voltage. According to the datasheet R_{50} should be in a range of 25 kΩ to 100 kΩ and R_{48} is calculated via Equation (14).

$$R_{48} = R_{50} * \left(\frac{V_{OUT}}{V_{SETBYP}} - 1 \right) \tag{14}$$

3.5 LED Matrix

One goal of the present embodiment is to handle realistic use cases. The major parts of LED matrices used in applications which are worth considering for the envisaged inventions are constructed like the matrix in Figure 2.13. That means each LED of a column can only be addressed by a common anode connection and each LED of a row can only be addressed by a common cathode connection. Unfortunately matrices of this kind, especially SMD LEDs matrices, have ingrained matrix drivers which are already connected to the matrix connections. It is nearly impossible to rip up the connections to use the matrix. Moreover these connections ought to be connected to the embodiment with free wires which cause a weak point for noise coupling because of the high impedance input of the opamp.

This is the reason why it was an elusive decision to build an own matrix directly on the PCB of the embodiment. Thus the connections to the amplifiers can be built substantially and short. Furthermore it is possible to choose LEDs which are well suited for the embodiment and close to the aspired applications. In the present embodiment red SMD Chip LEDs from Kingbright are attended. The LEDs provide an appropriate wavelength of 650 nm in a small 1.0x0.5 mm package and have a capacitance of 35 pF at a frequency of 1 MHz. Further details can be found in [KINGB1].

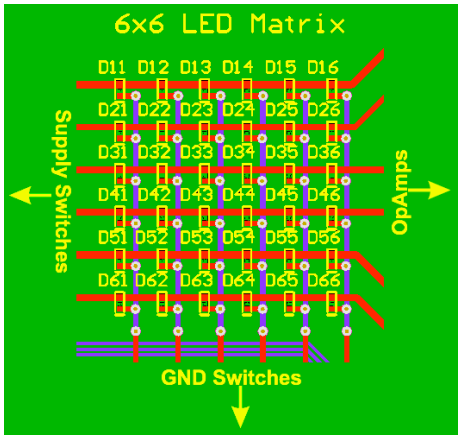


Figure 3.6: PCB-Layout of the LED matrix

Figure 3.6 is an extract of the PCB layout and shows the routing of the matrix connections. Two different layers were used to connect the LEDs in form of a matrix and short connections to the amplifier inputs were aspired. The supply voltage for the LEDs is provided through a series resistor per row behind the switches. Due to this hardware restriction the algorithm has to ensure that only one LED per row is connected to ground and therefore emits light. If this restriction is abided a circuit like Figure 3.7 supplies each LED.

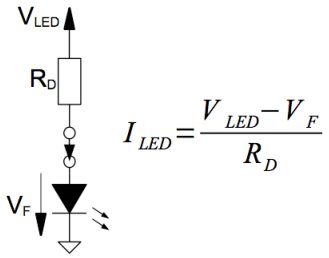


Figure 3.7: LED supply

With a series resistor of 100 Ω, the mentioned LED with a forward voltage of typically U_F=1.95 V and a supply voltage of 4 V a current of 20 mA flows through an active LED. The supply voltage has the same potential as the supply of the switches. Details can be found in subchapter 3.4 and Figure 3.5. The further discussion about the algorithms show that the LEDs are active only for a very short amount of time. Due to that it is possible to force a much higher current through the LEDs. The calculated current of 20 mA can also be used for a static use.

3.6 Transimpedance Amplifier

As mentioned in subchapter 2.3 the contrived embodiment monitors the photocurrent of the LEDs and converts it into a voltage signal by means of a current-to-voltage converter, also called transimpedance amplifier. After the upstream guesstimate of its operation now a detailed description of the circuit operation with some previous needed simulation results, the compromise between bandwidth and stability and the constraining hardware criteria are given. The exact schematic and the layout of the transimpedance amplifier of the embodiment can be found in Appendix A but subchapter 3.6.6 completes the discussion about the transimpedance amplifier with the main schematic used in the present embodiment in connection with the LED matrix at its input. A very helpful and detailed analysis of different circuit methods and

the stability of the amplifier is given by [GRAEME1] which builds the base of my theoretical research about it. A much more practical approach can be found in [BURRB1] and rounds off the literature for this part of the system.

3.6.1 Operation

The big advantage of the current-to-voltage converter is that zero load impedance is presented to the LED and on the basis of that nearly zero voltage is holding across the LED. Thus the circuit is able to achieve high linearity following the connection of photocurrent and flux energy of the incident light. Contrary to the transimpedance amplifier monitoring of the LED voltage would leave a voltage across the LED and leads to a logarithmic relationship between incident light and output voltage while the sensitivity of the LED changes with the bias voltage. As it is pictured in Figure 3.8 the circuit consists evidently of an LED, an opamp, a feedback resistor R_F and a feedback capacitor C_F.

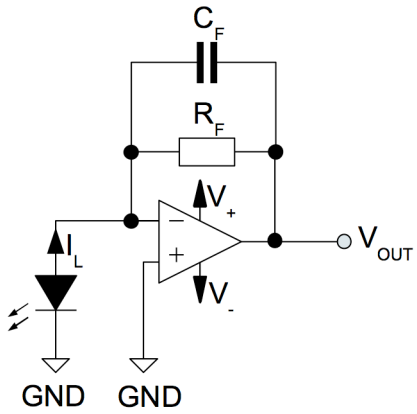


Figure 3.8: Current-to-Voltage Converter

For the fundamental operation of the transimpedance amplifier the feedback capacitance C_F is negligible. An opamp with negative feedback tries to keep the differential voltage between its inputs at zero. The non-inverting input of the opamp is connected to ground. To keep the inverting input at virtual ground too, the output of the opamp forces a current over the feedback resistor R_F. This compensates the voltage on the inverting input depending on the photocurrent I_L generated by the LED. Equation (15) shows the connection of photocurrent and output voltage. Although R_F normally takes high values the provided input impedance is only R_F/A_{OL} and therefore negligible compared to the high output resistance of the LED.

$$V_{OUT} = -I_L * R_F \tag{15}$$

In case of a leakage current over the LED it flows in the opposite direction of I_L. Because of the ability of the circuit that no voltage appears over the junction of the LED the leakage current over the LED can be neglected and with it the flux responsivity of the LED is constant. An input current at the inverting input of the opamp, however, may occur and producing a dc offset, extends Equation (15) to:

$$V_{OUT} = -I_L * R_F + (I_{B-} * R_F) \tag{16}$$

Particularly with regard to the very small photocurrent of an LED a comparable input bias current of the opamp could adulterate the output signal.

Bandwidth and Stability:

Four main facts limit the bandwidth of the circuit and therewith the stability of the current-to-voltage converter is determined. The parasitic capacitance of the feedback, the limited bandwidth of the opamp, also phase compensation and the frequency response of the LED. Normally one of the first three limits the bandwidth at a much lower frequency than the LED does. The response limits formed by the parasitic capacitance and the bandwidth of the opamp are inherent in any current-to-voltage converter.

- Parasitic capacitance

  The stray capacitance C_S bypasses the feedback resistor R_F and at higher frequencies the current I_L is thereby shunted away from the resistor (Figure 3.9). If the parasitic capacitance is dominant the -3-dB bandwidth is given by:

$$f_{p(cap)} = \frac{1}{2\pi * R_F * C_S} \tag{17}$$

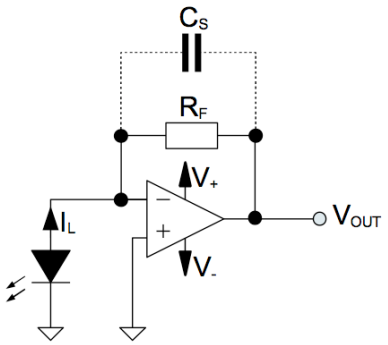


Figure 3.9: Transimpedance amplifier with a parasitic capacitance in its feedback

  A higher value of the feedback resistor R_F makes the response limit, caused by the parasitic capacitance, weightier by shifting down the resulting pole frequency. As it will be shown in some cases phase compensation is needed for the stability of the circuit. Such compensation already needs a capacitance to bypass R_F. Subsequent equations will show how a bypass capacitance is calculated. If the equations result in a smaller value than C_S or equal no more phase compensation is needed.

- Opamp bandwidth

  In contrast to the theory the voltage across the diode (and its junction capacitance) is only zero for low frequencies. In practise the opamp gain is not infinite and for high frequencies it is limited due to the opamp bandwidth. Therefore a voltage V_{OUT}/A_{OL} remains between the opamp inputs. Figure 3.10 shows a model of the circuit to explain the frequency limit of the transimpedance amplifier. The LED is substituted by a current source and the parasitic capacitance. Apart from the junction capacitance of the LED, also the opamp has input capacitance elements. A common –mode input capacitance C_{icm} and a differential input capacitance C_{id} are present at the opamp inputs. Following the statements of [CARTER1, p.197] in case of a grounded noninverting input a source at the inverting input will see a parallel circuit of C_{id} and the half amount of the C_{icm}. On account of some stray capacitances it is nevertheless useful to calculate the full amount of C_{icm}. Hence the input capacitance totals up to

$C_i=C_D+C_{icm}+C_{id}$. The total capacity C_i and the infinite opamp gain lead to a frequency limit of:

$$f_{p(opamp)} = \frac{A_{OL}}{2\pi * R_F * C_i} \tag{18}$$

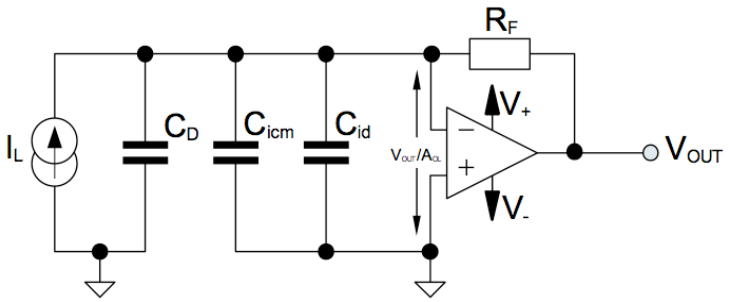


Figure 3.10: Transimpedance amplifier with its input capacitances and LED parasitic capacitance

- Phase compensation

To avoid the current-to-voltage converter from oscillating or resonate in some cases phase compensation is needed. The amplification of the input noise voltage could turn the transimpedance amplifier to oscillate. From a DC point of view the input noise voltage e_n is transferred to the output with a gain of $1+R_F/R_D$. At higher frequencies the capacitances of the circuit is responsible for the gain of e_n. Figure 3.11 helps to understand how the input noise voltage weights the current-to-voltage frequency response.

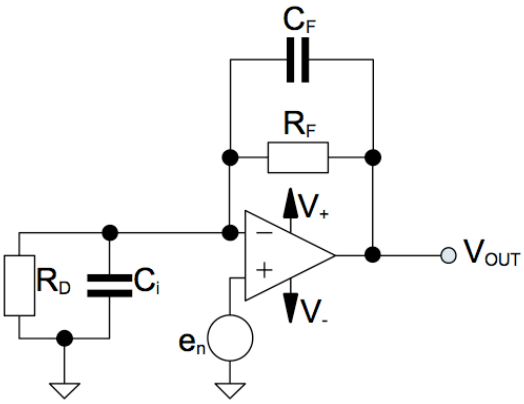


Figure 3.11: Noise amplification in the transimpedance amplifier

A calculation of the noise gain leads to Equation (19) and underlines the interest of the capacitances for higher frequencies in the circuit. The higher the frequency the higher the impedance of the feedback capacitor is.

$$A_{NOISE(f)} = \frac{V_{OUT}}{e_n} = \frac{R_D \,//\, C_i}{R_D \,//\, C_i + R_F \,//\, C_F} = \frac{R_F + R_D}{R_D} * \frac{1 + j\frac{f}{f_Z}}{1 + j\frac{f}{f_N}} \tag{19}$$

$$f_Z = f_p = \frac{1}{2\pi * R_F * C_F} \tag{20}$$

$$f_N = f_z = \frac{1}{2\pi * (R_F \,//\, R_D) * (C_i + C_F)} \tag{21}$$

As Equation (20) defines the pole frequency f_p is determined by the feedback components R_F and C_F. R_F//R_D and the sum of the two capacitances C_F and C_i define the zero frequency f_z. If $R_D >> R_F$ the DC gain can be assumed to be $A_{NOISE(DC)} = 1$ and therefore the pole frequency is lower than the zero frequency ($f_p < f_z$). For a frequency higher than f_z only the capacitances are responsible for the noise gain (22).

$$A_{NOISE(f>fp)} = \frac{C_i + C_F}{C_F} \tag{22}$$

A problem with stability of the current-to-voltage converter occurs when the pole frequency leads to a noise gain which is outside the opamp's open-loop gain. In Figure 3.12 three different pole frequencies are illustrated. f_{p1} results in a stable condition but bandwidth is wasted. However, f_{p3} leads to an unstable condition because the noise gain is higher than the open-loop gain of the opamp. The ideal case is a pole frequency f_{p2}. To achieve this perfect condition C_F has to be calculated in the context of noise gain equal to the open-loop gain of the opamp.

$$A_{NOISE(f>f_p)} = A_{OL} \rightarrow \quad \frac{C_i + C_F}{C_F} = \frac{GBW}{f_P} \tag{23}$$

$$C_F^2 - \frac{C_F}{2\pi * GBW * R_F} - \frac{C_i}{2\pi * GBW * R_F} = 0 \tag{24}$$

$$C_F = \frac{1}{4\pi * GBW * R_F} + \sqrt{\frac{1}{16\pi^2 * GBW^2 * R_F^2} + \frac{C_i}{2\pi * GBW * R_F}} \tag{25}$$

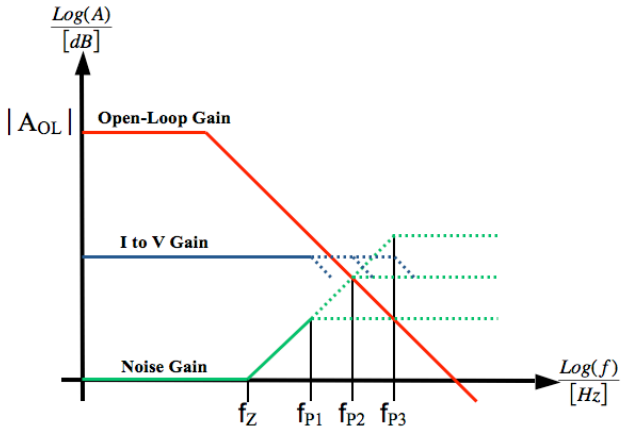


Figure 3.12: Different gains of the circuit and the resulting poles
[Adopted from [TEX1, p.2]]

As Equations (20) and (21) demonstrate the feedback resistor R_F is directly influences the frequency response and within the bandwidth of the current-to-voltage converter. On the one hand a higher value of R_F increases the signal gain but at the same time it decreases the bandwidth.

Two-stage Amplifier:

A good compromise between bandwidth and gain is the use of a two-stage amplifier. The first opamp uses a moderate feedback resistor followed by a second amplifier with voltage gain. In Figure 3.13 such kind of a two-stage amplifier is shown. The circuit is also used in the introduced embodiment. Beneath the advantage of a wider bandwidth the negative output voltage of the current-to-voltage converter is converted to a positive output voltage by the second amplifier stage. One disadvantage of the two-step structure is that the noise and offset effects of the current-to-voltage converter are amplified by the second stage.

The resulting transimpedance of the two-stage structure which complies the current-to-voltage gain of the whole circuit results from the current-to-voltage gain of the first stage (Figure 3.13 I.) and the voltage gain of the second stage (Figure 3.13 II.).

$$A_{Two-Stage} = R_F * \left(\frac{R_2}{R_1} \right) \tag{26}$$

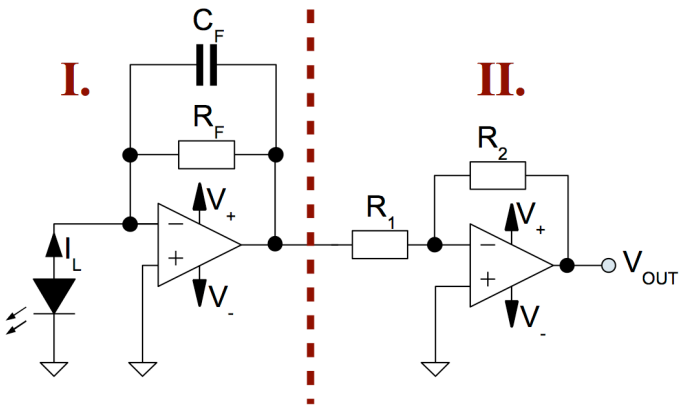


Figure 3.13: Two-stage amplifier structure

Offset:

Based on the amplifier structure in Figure 3.13 the dc noise level of the output after the two stages can be calculated. The current-to-voltage converter forces its voltage offset directly to its output. As already mentioned in Equation (16) an offset current is converted by the feedback resistor and is added to the output voltage. All offset parasites of the first stage are again amplified by the inverting opamp. The offset voltage of the second stage is also amplified by a factor $(R_1+R_2)/R_1$ and the offset current contributes over the feedback resistor R_2 of the second stage to the total offset of the system. Equation (27) sums up all offset influences to an overall output offset $V_{Off(out)}$.

$$V_{Off(out)} = \left(V_{Off\_TIA} - I_{Off\_TIA} * R_F\right) * \left(-\frac{R_2}{R_1}\right) + V_{Off\_Inv} * \left(\frac{R_1 + R_2}{R_1}\right) + I_{Off\_Inv} * R_2 \qquad (27)$$

Noise:

Especially the gain of small input signals requires the attention to noise influences. The noise of the circuit is determined by the resistance noises and the opamp noises. In Figure 3.14 the individual noise sources of the circuit are located. Equations (19) and (22) already cover the noise characteristics of the current-to-voltage converter. The important conclusion which has to be respected if a two-stage amplifier is used is that the noise of the first opamp stage is amplified by the second stage.

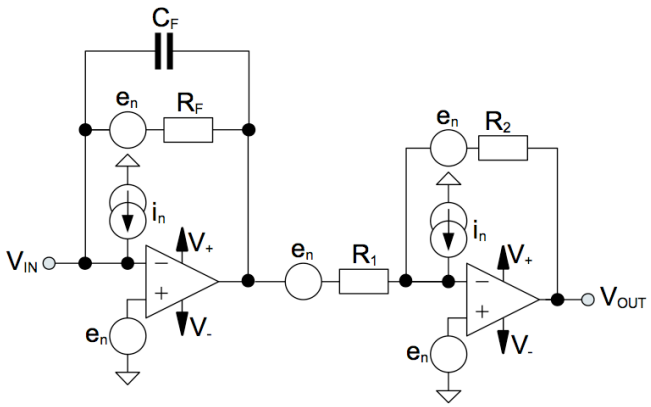


Figure 3.14: Noise sources in the two-stage circuit

3.6.2 Hardware Criteria and Dimensioning

In the present embodiment the current-to-voltage converter has to convert and amplify a very low photocurrent signal (in the range of nA) generated by LEDs. Therefore the opamps open loop gain plays a major role. The photocurrent is not constant because the LEDs toggle between detecting and emitting light hence the converter has to provide an appropriate bandwidth. As pictured in Figure 3.14 the noise densities of the opamps are further important hardware criteria. Above Equation (27) states, that the input bias current of the opamp has to be small as well as the offset voltage. The current-to-voltage converter is only a part of a system and therefore it should not be forgotten that the supply voltage for the opamp has to be in a range that can be provided in the system.

The following criteria therefore led to the right choice of opamp:

- Low input bias current
- Low offset voltage
- Low input capacitance
- Low noise densities
- Wide signal bandwidth

- High open-loop gain
- Supply voltage ≤ 5 V

Additionally a small package including two or more independent opamps but still solderable manually is a required technical feature. The selected opamp is the *AD8608* from Analog Devices. Table 1 reports the essential attributes of the *AD8608* datasheet [ANALOG1] at a supply voltage of 5V and an ambient temperature of 25 °C.

Supply Voltage	6 V	Unity Gain Bandwidth Product	10 MHz
Input Bias Current	max. 1 pA	Common-Mode Input Capacitance	8.8 pF
Offset Voltage	max. 300 µV	Differential Input Capacitance	2.6 pF
Voltage Noise Density	max. 12 nV/$\sqrt{\text{Hz}}$ (f=1 kHz)	Signal Voltage Gain	1000 V/mV
Current Noise Density	0.01 $pA/\sqrt{\text{Hz}}$ (f=1 kHz)	Supply Voltage	2.7 V to 6 V

Table 1: AD8606 attributes with 5 V supply voltage and 25 °C

Based on the choice for AD8608 and the expected input signals (see Chapter 5.1 for measurements of the LED photocurrent over incident light) a transimpedance of altogether 10 mV/nA is needed to amplify the current signal. This will be achieved by a feedback resistor R_F = 1 MΩ in the first amplifier stage and the relation of R_1 =1 kΩ to R_2 =10 kΩ in the inverting amplifier of the second stage. With the help of the subsequent simulations a feedback capacitor of C_F =10 pF completes the feedback path of the first stage.

3.6.3 Power Supply for the Transimpedance Amplifier

To achieve a broadly clean output signal it is important to provide a quiet reference potential. A possible solution is to source the opamps with dual supply voltages and to set the reference signal of the in and output signals in the middle of the opamp supply voltage. This separates the reference potentials of the signal and the opamps. The supply voltage of the opamp has to be in a range of 2.7 V to 6 V. Separate the supply voltage in a positive 2.5 V and a negative 2.5 V lead to a total voltage supply of 5 V and a center potential of 0 V (ground) for the signals. Figure 3.2 shows the connection between the supply voltage of the USB bus and the supply voltage of the opamps.

The +2.5 V can be transformed easily from the +5 V USB bus voltage in the same way as the supply voltage for the switches and the LEDs. Figure 3.15 shows the resulting circuit and values while the appropriate description can be found in subchapter 3.4.

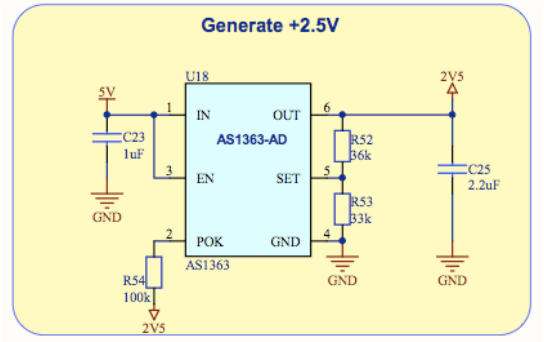


Figure 3.15: Positive voltage supply for the opamps

To generate a negative voltage out of a positive one is a little bit more complicated. The designated approach is to generate the negative supply voltage through two stages. The advantage of this solution is to generate a cleaner output voltage with a small ripple on the output voltage. First of all the +5 V are inverted by a switched capacitor voltage inverter. In the present embodiment the *MAX829* from *Maxim* is used [MAX1]. The capacitive charge pump needs three external components. A flying capacitor C_{28} is charged with the input voltage in the first half of the switching cycle. The stored charge of the flying capacitor is pumped in the reservoir capacitor C_{26} in the second half of the switching cycle to provide a negative output voltage. The switching frequency of the chip is 35 kHz and can be monitored as a ripple of the output voltage. The output ripple of the charge pump can be reduced with a large output capacitor C_{26}. The third external component is a bypass capacitor C_{24} at the input of the chip to reduce the backlash of the switching noise. A resistor in series to the charge pump input can also avoid the influences of the switching noise on the USB supply voltage. It is recommended to use capacitors with low ESR to minimize the output resistance and the voltage ripple. The more output current is needed, the more attention is needed in the generated noise.

After the capacitive voltage inverter the output voltage is still affected by a voltage ripple of 20 mV with a frequency of 35 kHz. This signal has to be downsized and straightened for the required -2.5 V with the help of a negative voltage regulator. The used regulator is the *LM337* from *National Semiconductors*. In contrast to the used LDO *AS1363* for positive voltages the *LM337* uses bipolar transistors and is able to handle negative in- and output voltages. An external voltage divider is commonly used to set the output voltage following Equation (28). Additionally three external capacitors for a rejection of transients and a reduction of the output ripple are used.

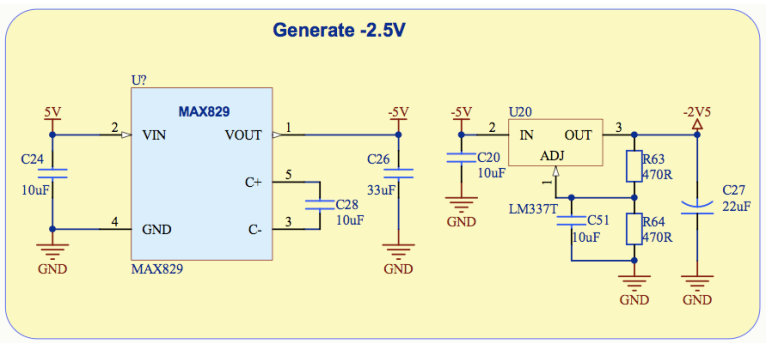


Figure 3.16: Negative voltage supply for the opamps

$$-V_{OUT} = -1.25V * \left(1 + \frac{R_{64}}{R_{63}}\right) + \left(-I_{ADJ} * R_{64}\right) \tag{28}$$

The current I_{ADJ} in Equation (28) corresponds to the output current of the ADJ Pin and is in a range of some µA (temperature dependent). It can be neglected for the calculation of the output voltage. After the second stage of the negative voltage conversion an output signal with a ripple lower than 2 mV is achieved. The measurements in Figure 3.17 and Figure 3.18 show the resulting output signals of the two conversion stages under load conditions.

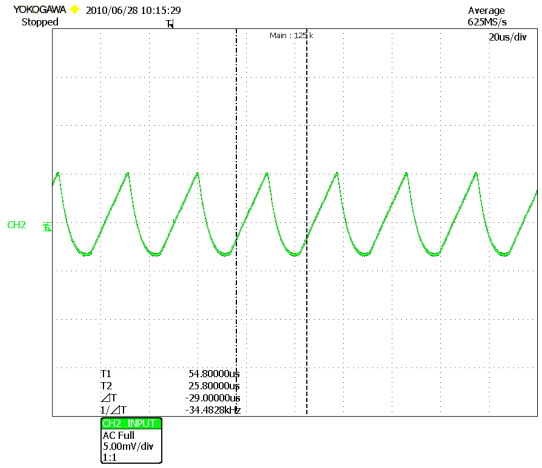


Figure 3.17: Charge pump output

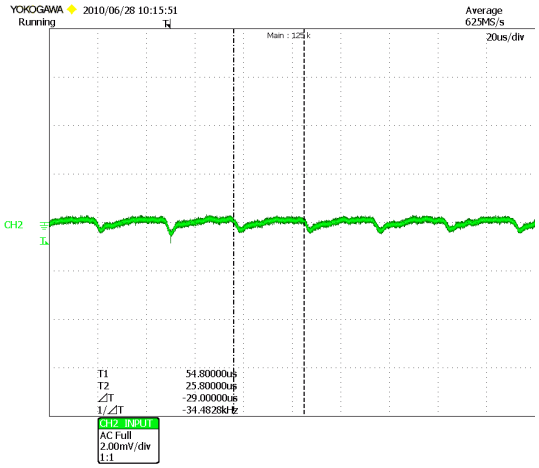


Figure 3.18: Negative LDO output

3.6.4 Simulations

With the help of circuit simulations it is possible to revise the theoretically derived context about the current-to-voltage converter and the following inverting amplifier. In addition the part values in the circuit can be estimated and, if applicable, can be changed. The amplifier stages were simulated with PSPICE. Owing to the poor possibilities in the graphical evaluation of PSPICE, the simulation data were transformed to MATLAB. In MATLAB ensued the graphical preparation of the results.

First of all a simulation of the current-to-voltage converter with an LED at its input or rather the equivalent circuit of an LED with its parasitic components underlines the discussed stability problem of the circuit (Figure 3.19).

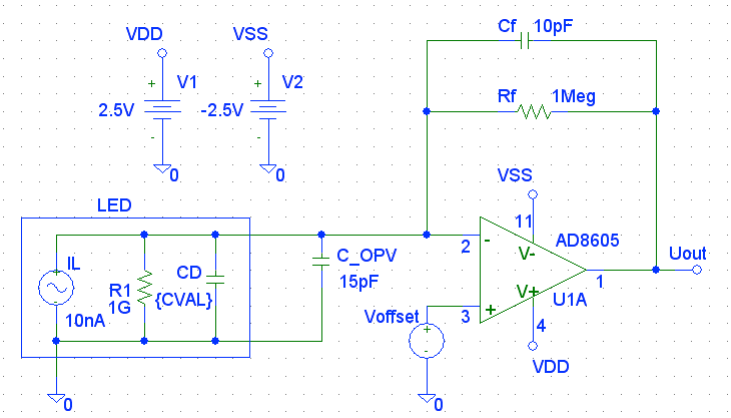


Figure 3.19: Simulation circuit for the current-to-voltage gain

A resistor R_F forms the feedback of the amplifier. For the first simulation the feedback capacity is assumed very small to eliminate its effect on the circuit behaviour. The parasitic capacitance of the LED C_D is variable in a defined range from 1pF up to 1nF. The opamp is deposed with a model of the *AD8605* which is available from the Analog Devices homepage. The amplifier is equal to the AD8608 but with only one independent opamp in its package. An exact inspection of the *AD8605* model library reveals that the common-mode input capacitance and the differential input capacitance are not taken into consideration. Therefore the input capacitances of the opamp are modelled with the capacitor C_{OPV}. An offset voltage at the non-inverting input of the opamp guarantees its necessary area of operation. The resulting diagram in Figure 3.20 represents the bode plots with four different LED capacitances.

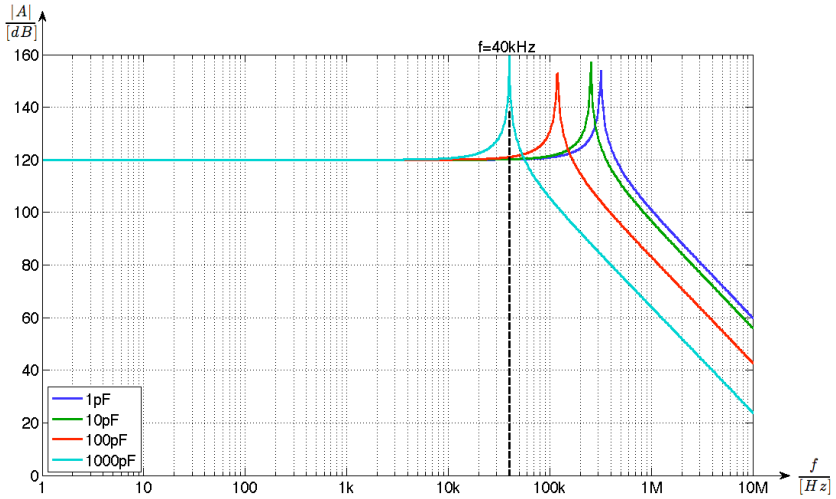


Figure 3.20: Current-to-voltage gain without a feedback capacitance

The bode plots confirm the preceding statements above: an increasing input capacitance diminishes the bandwidth of the transimpedance amplifier and supports the turn of the circuit to oscillate if it is stimulated. Concerning the waiver of a feedback capacity the pole of the frequency response is not compensated and consequently the current-to-voltage gain strongly increases at the frequency f_C where the noise gain and the open-loop gain cross. As an example the resulting frequency of the 1nF input capacitor is highlighted (40 kHz).

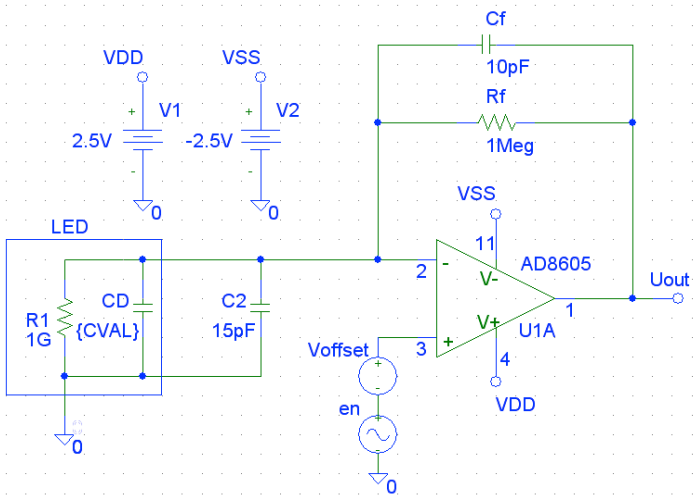


Figure 3.21: Simulation circuit for the noise gain

Another circuit with an input noise voltage e_n (Figure 3.21) is able to simulate the frequency response of the noise gain and precedes the frequencies where the current-to-voltage gain strongly increases. Figure 3.22 pictures the corresponding diagram and points out that the noise gain increases at the same frequency as the current-to-voltage gain. The dashed blue line interpolates the progression of the exemplary 1000 pF feedback capacitor and crosses the open loop gain of the opamp at the discussed frequency (40 kHz).

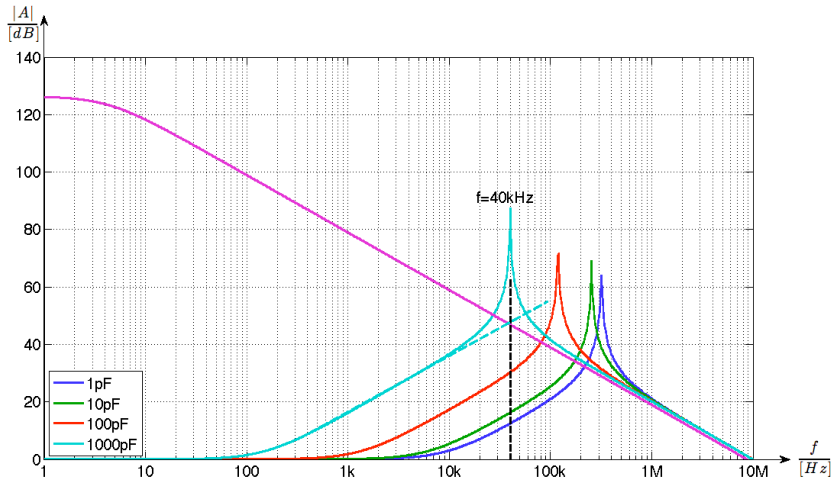


Figure 3.22: Noise gain without a feedback capacitor

The same circuit as in Figure 3.19 but now with a feedback capacitance of C_F=10 pF leads to the bode plots of Figure 3.23. The -3 dB cut-off frequency is defined by the feedback components build low-pass as mentioned in Equation (20). The division of the feedback current between the feedback resistor R_F and the feedback capacity C_F follows in Figure 3.24. With such a high value of C_F the pitch of the current-to-voltage gain is diminished and the circuits' turn to oscillate is prevented.

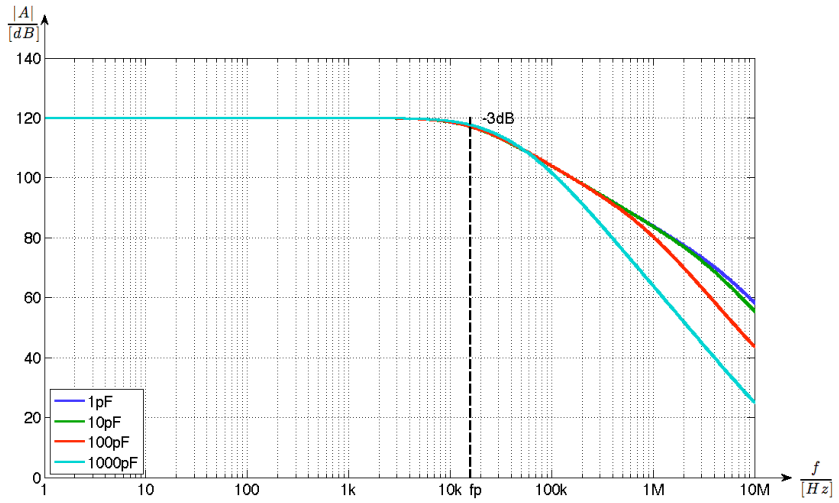


Figure 3.23: Current-to-voltage gain with a 10pF feedback capacitor

Due to the fact that all of the simulated input capacitances determine a frequency f_C which is higher than the -3 dB cut-off frequency of the feedback low-pass each input case could be compensated.

The current curves in Figure 3.24 were recorded for an input capacitance of 1nF. For low frequencies the opamp is able to provide virtual ground at the non-inverting input and the current is constrained to flow over the feedback resistor. In line with the frequency increase the impedance of the feedback capacitor increases and with it the current $I(C_F)$. In contrast an uncompensated circuit without C_F would have a significant current peak in the feedback path of the resistor R_F at the frequency f_C.

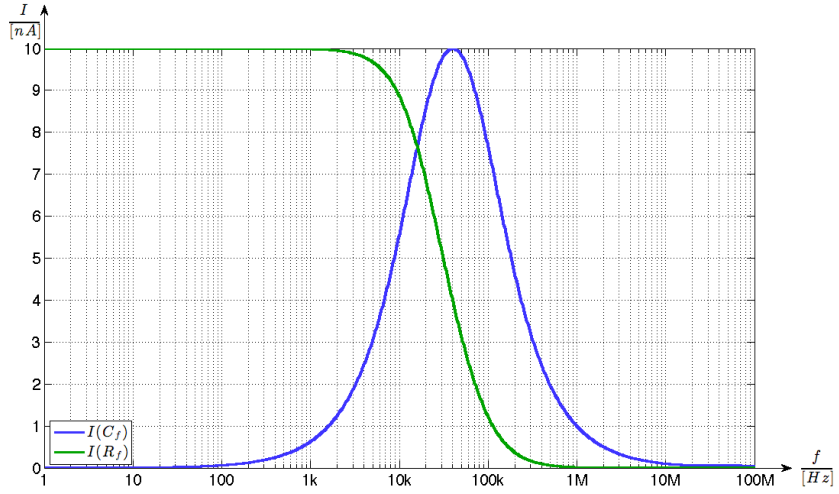


Figure 3.24: The feedback currents at a feedback of R_F=1 MΩ and C_F=10 pF and an LED input capacitance of 1nF

Once again the noise gain and the open loop gain of the opamp are pictured in Figure 3.25. The compensation of the pole can be seen at the frequency f_p given by Equation (20). Regardless of the different input capacitances the compensation always starts at the same frequency. The +3 dB cut-off frequency of the noise gain can be retraced by Equation (21). Basically the input capacitances and the feedback resistor build the responsible high-pass.

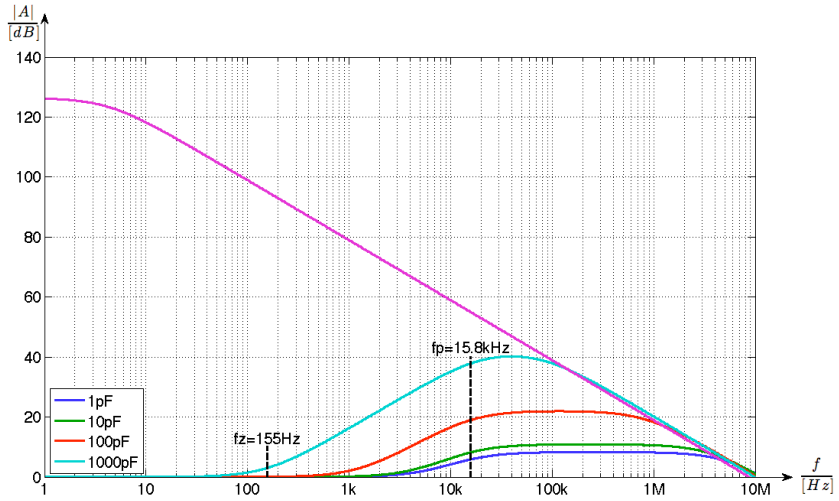


Figure 3.25: Noise gain with a 10pF feedback capacitor

Of course C_F =10 pF is not the best case value following Equation (25) but allows a strong variation of the input capacitance which strongly hinge on the used LEDs and the PCB layout and still prevent the circuit from becoming unstable.

3.6.5 Measurements

For the upcoming measurements of the output signals of the amplifier stage a separate PCB provides a setup independent from the LED matrix and the microcontroller. Only one LED is connected to the input of the two-stage amplifier and an oscilloscope directly monitors its output. Two LEDs are fixed as neighbors next to the monitored LED but they are supplied via an independent voltage source. To avoid noise influence the measurement is done in a lightproof metal box with a connection to ground.

The first picture of the oscilloscope (Figure 3.26) exhibits the output signal of the amplifier stage with light emitting neighbors. As one can see a dc-level of 208 mV can be measured. As another ambient light source acts an ordinary 40 W light bulb. In Figure 3.27 the bulb acts as single light source and produces an output level of 33.8 mV with a voltage ripple of 100 Hz.

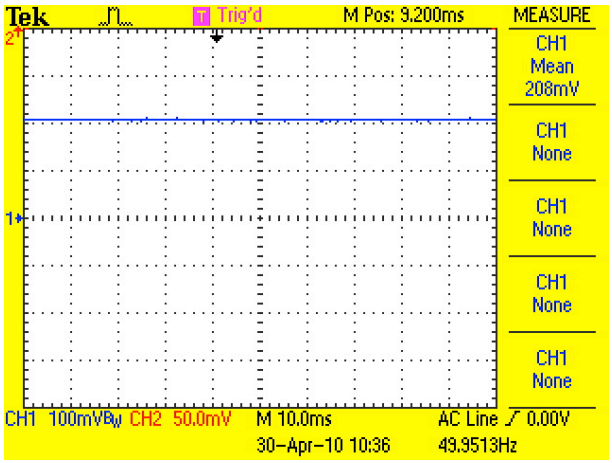


Figure 3.26: Opamp output signal; emitting neighbors

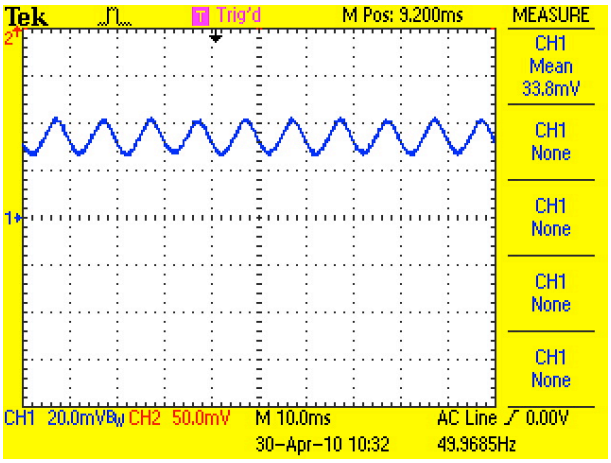


Figure 3.27: Opamp output signal; emitting light bulb

It is expected that the combined light sources (neighbors and light bulb) generate an output corresponding to the sum of each output level. Figure 3.28 confirms this assumption with an output voltage of 242 mV.

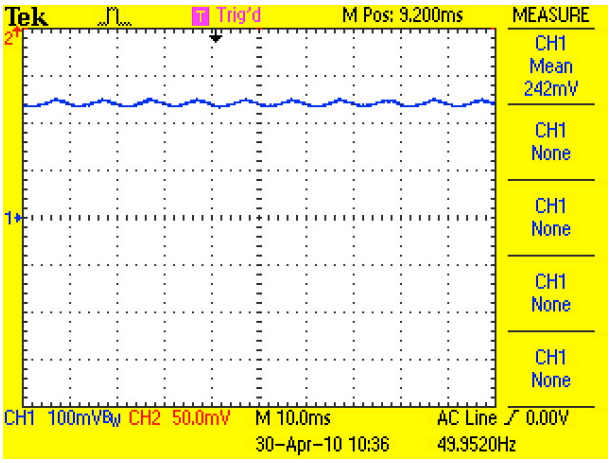


Figure 3.28: Opamp output signal; emitting neighbors and light bulb

The last measurements cover the behaviour of the opamp output signal if an object covers the LED surface and the neighbor LEDs are switched. The present measurement uses an ordinary slip of white paper to cover the LED. In Figure 3.29 the neighbors are switched. The output signal reaches again 208 mV like in the static case. Figure 3.30 now shows the effect of the paper. The output signal increases up to more than 500 mV caused by the reflected light of the white paper. In Chapter 5 further measurement results are discussed, especially the algorithm-dependent switching and the according output signals.

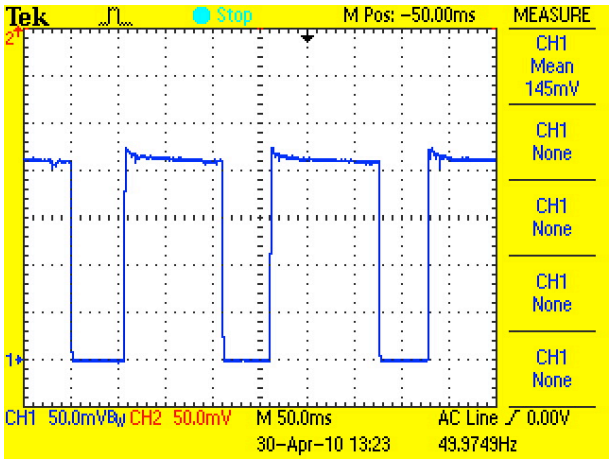


Figure 3.29: Opamp output signal; switched neighbors

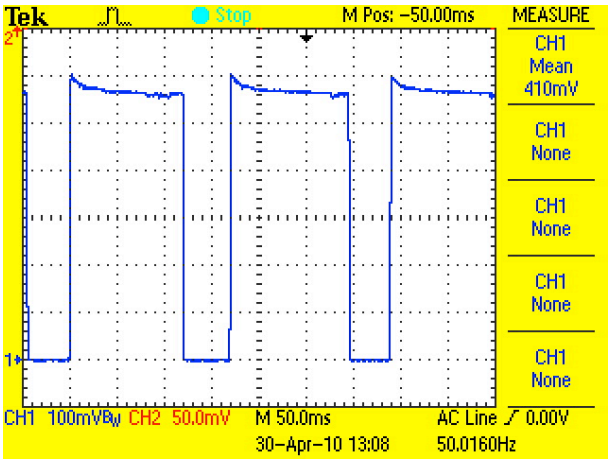


Figure 3.30: Opamp output signal; switched neighbors and covered surface

3.6.6 Schematic

With the help of the full circuit including all amplifier stages and the LED matrix at its input up to the ADC inputs it is possible to underline which signals are actually measured at the different switching positions. Two cases have to be distinguished while one of the algorithms sets the switches. The measurement without an emitting neighbor, like Figure 3.31 and the measurement while one neighbor of the monitored LED emits light (Figure 3.32).

When none of the LEDs in the matrix emit light and one column is connected to ground every amplifier stage produces an output voltage according to the ambient light. As one knows the photocurrent I_L flows towards the amplifier input. It is obvious that the value for the ambient light can be sampled for each LED of a column at the same time. The decision itself which values are sampled depends on the algorithm. The on-resistance of the column switch leads to a voltage drop and the cathodes of the LEDs are not exactly on ground potential. As the generated photocurrent is very small the voltage drop at the LED cathodes can be neglected.

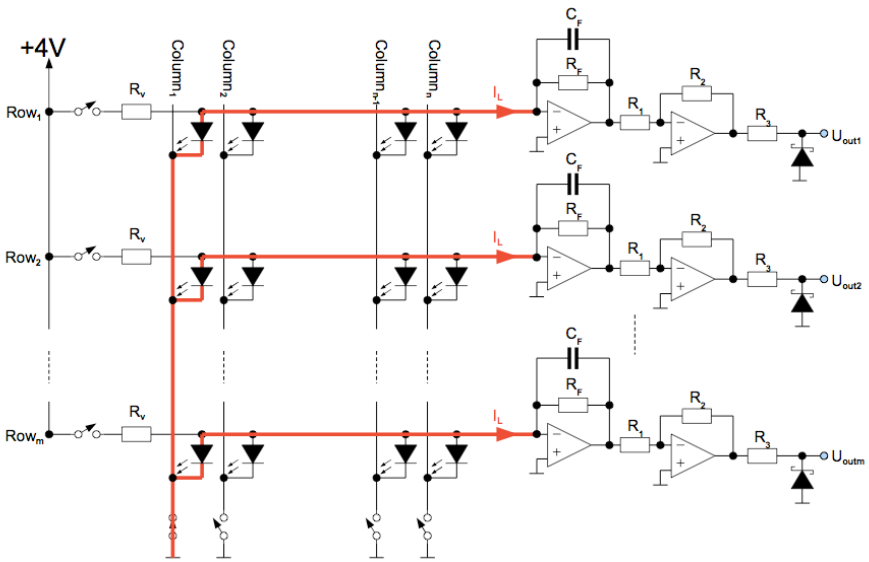


Figure 3.31: Measuring with none emitting neighbors

The second measurement case arises if one LED (or more) emits light. To allow the LED to send out light the corresponding column and row switches have to be closed. A current I_{ON} flows through the LED towards ground. Again a photocurrent is generated by all other LEDs of the column. To be able to compare the corresponding output voltage of each LED only the output voltage of the direct neighbors should be sampled but it depends again on the algorithm. The output voltage of the amplifier stage at the same row as the emitting LED is definitely unusable because of the high input the amplifier stage gets in its output rail. The same effect as described above happens at the on resistance of the column switch. Now the voltage drop is weightier due to the higher current towards ground. Subchapter 5.5 discusses the weak points of the system in detail including the voltage drop at the LED cathodes in case of emitting neighbors.

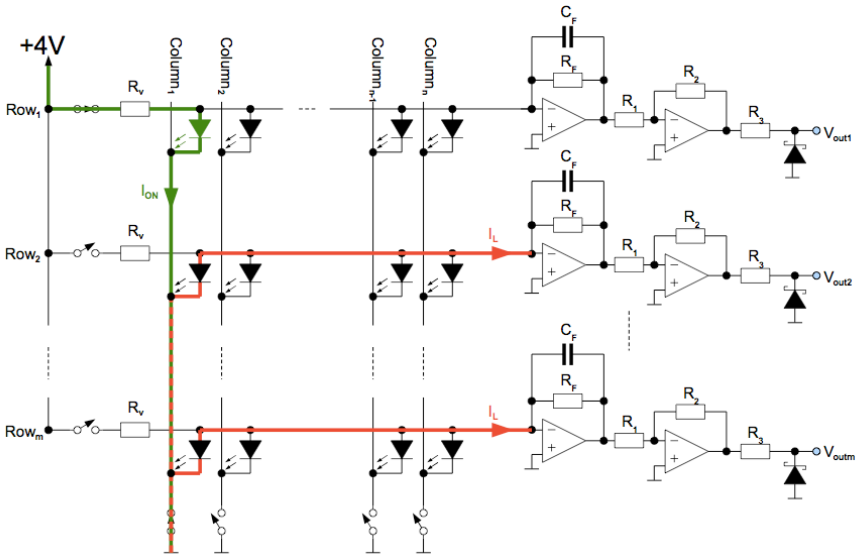


Figure 3.32: Measurement with an emitting neighbor

3.7 Microcontroller PIC24F

The intention to use a microcontroller for the present embodiment rests on some predetermined system requirements. First of all a device is needed to control the switches surrounding the matrix due to an algorithm. The output signals of the amplifiers have to be digitized by some ADCs to provide the values to a logical control unit in a short amount of time. Additionally an interface to an ordinary PC has to be provided for the possibility to display the evaluated data. Last but not least the embodiment should be working as independent as possible and should not exceed the dimension of a portable demoboard.

With a thoughtful decision for the right device a microcontroller can fit all these requirements. It provides sufficient output pins to control the switches, has an internal ADC with sufficient input channels and provides a logical platform for the algorithm to run on board. Moreover a microcontroller contributes an interface outward and allows a debugging of the algorithm during an application. Owing to the fact that not every part of the system is predefined in detail before the PCB has to be designed the microcontroller provides flexibility with its various modules.

3.7.1 Selection Criteria

In summary of the criteria mentioned above and substantiated with the detailed application requirements the selection criteria for a microcontroller are shown in Table 2:

10bit-ADC with Six Channels	18 Output Ports
USB Communication	In circuit programming/debugging
Supply Voltage ≤ 5 V	

Table 2: Microcontroller selection criteria

The choice was made in favour of a microcontroller *PIC24FJGB004* from *Microchip* in a 44-pin package. First and foremost, the decision was based on the fact that the *PIC24FJGB004* meets all criteria but also the fact that this microcontroller is widely used in the company has contributed to the choice. The support of colleagues and the fact that a development environment with an existing demoboard already exists underlines the decision. Table 3 gives an overview of the important characteristics of the microcontroller in the actual field of application. Further details can be found in its datasheet [MICRO1].

16-bit modified Harvard architecture	Fast internal RC oscillator
USB Communication	10-bit A/D Converter with 13 Channels
33 I/O-Pins	16 MIPS
In-Circuit Debugger/Emulator	8.192 Bytes Data Memory
64 Kbytes Program Memory	

Table 3: Important characteristics of the PIC24FJ64GB004

3.7.2 Power Supply for the Microcontroller

Due to the fact that the microcontroller uses USB to communicate with the outside world the advantage of an existing 5 V voltage supply can be used. The microcontroller itself needs a 3.3 V supply. With the LDO *AS1359* from *austriamicrosystems* [AMS3] which provides a fixed output voltage of 3.3 V a sufficient supply voltage can be provided. Principally the LDO works as described in subchapter 3.4 and Figure 3.33 exhibits the responsible supply circuit.

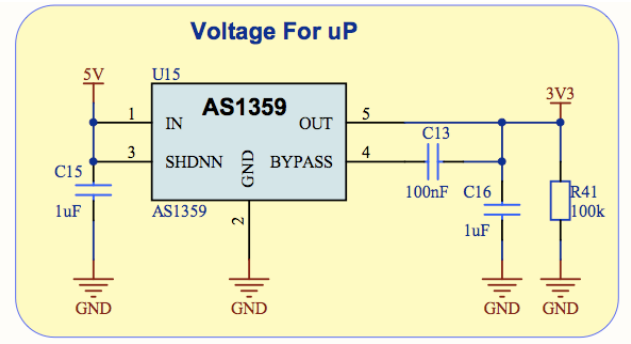


Figure 3.33: Voltage supply for the microcontroller

In contrast to the supply circuit of subchapter 3.4 a resistor with 100 kΩ is connected between the output and ground. In the case of unplugging the USB connector this resistor guarantees a reset of the USB voltage to the defined ground potential.

3.7.3 Circuitry of the Microcontroller

The implementation of the microcontroller into the embodiment does not need a big hardware effort as Figure 3.34 shows. Beneath the supply pins (VDD, VSS, AVDD, AVSS) are two other voltage inputs: the reference input for the ADC (VRef+, VRef-) and the VBUS input to recognize if an USB device is connected. In the present embodiment no external oscillator is needed as under typical conditions the internal oscillator meets the required accuracy specification of ±0.25 %. The data communication of the USB needs the D+ and D- pins. Another communication port is the debug interface which is provided through MCLR, PGED1 and PGEC1. This interface allows the programming of the microcontroller and in circuit debugging. R_{49} is important to pull up the MCLR pin (reset input) to prevent the microcontroller from unwanted resets. To enable the internal voltage regulator of the microcontroller pin DISVREG is connected to ground and in addition an external capacitor is connected to VCAP. The capacitor supports the stability of the internal voltage regulator.

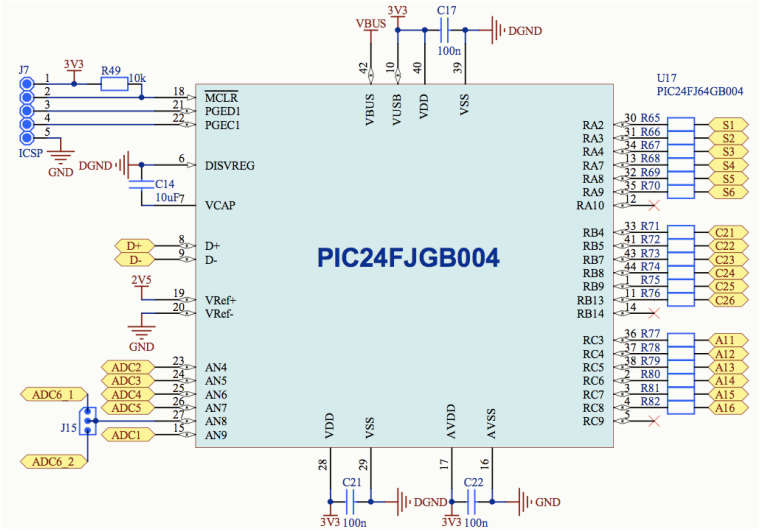


Figure 3.34: Circuitry of the microcontroller

The I/O ports (RAx, RBx, RCx) have a 100Ω resistor in series to its relative long connections to the switches. This resistor improves the EMC behaviour. The inputs of the ADC channels (ANx) are connected to the outputs of the amplifiers. To protect the inputs of the ADC channels at each amplifier output a current limit resistor is in series to the ADC input. In addition an Schottky diode is clamped against ground to avoid the ADC inputs from negative voltages. See also Figure 3.31. More details to the ADCs come up in the subsequent chapters.

3.7.4 ADC

As mentioned above, the PIC24FJ64GB004 provides a 10-bit ADC with 13 analog input channels. The ADC is a Successive Approximation Register (SAR) and reaches sample rates up to 500 ksps. Figure 3.34 exhibits that the reference voltage for the ADC is set on +2.5 V. With a resolution of 10-bit an ideal code width of 2.4 mV can be achieved which leads to an ideal current resolution of round about 0.25 nA per LSB.

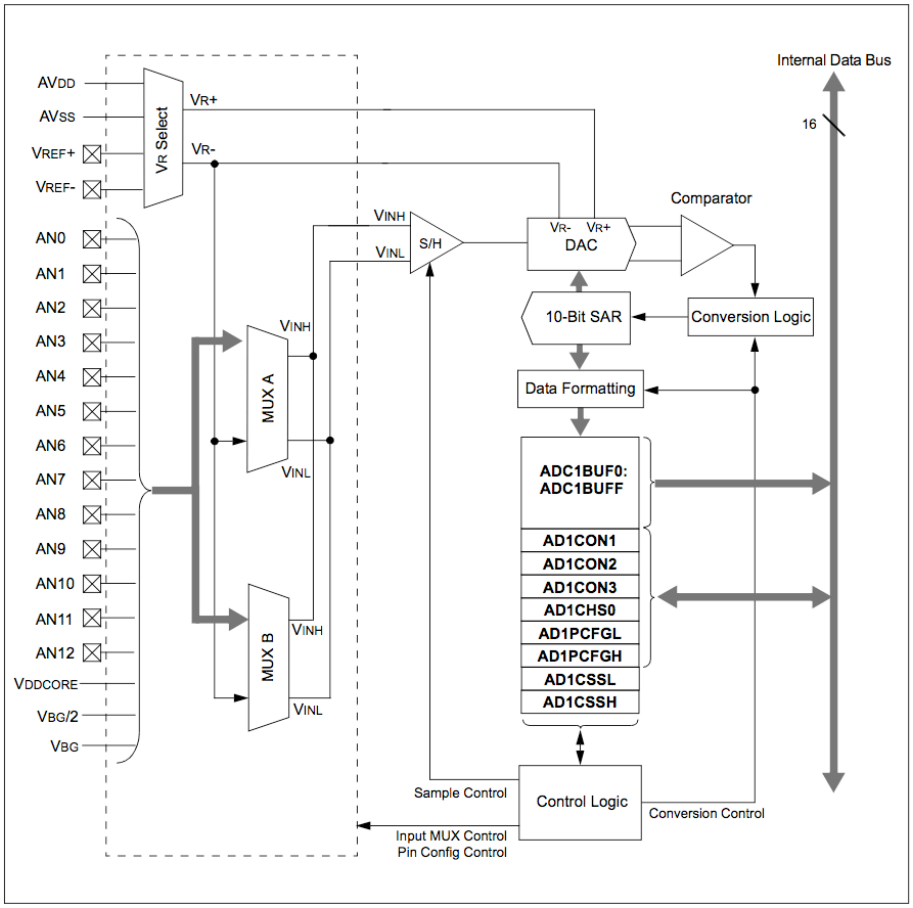


Figure 3.35: ADC block-diagram
[Extracted from [MICRO1, p.260] with kind permission of Microchip]

Figure 3.35 pictures the block diagram of the ADC. The different channels are connected to two multiplexers (predefined by the software) which forward the channels to a sample and hold amplifier. The sample and hold circuit is realized with a switched capacitor to acquire the input signal. The subsequent conversion saves the values in a 16-word result buffer and provides a link to the internal data bus. The resulting output code can be calculated via Equation (29). Eight configuration registers predefine the conversion sequence of the ADC. The preference details can be found in subchapter 4.2.2, dealing with the ADC software.

$$OutputCode = F.S.* \frac{V_{IN}}{V_{\text{Ref}}} \tag{29}$$

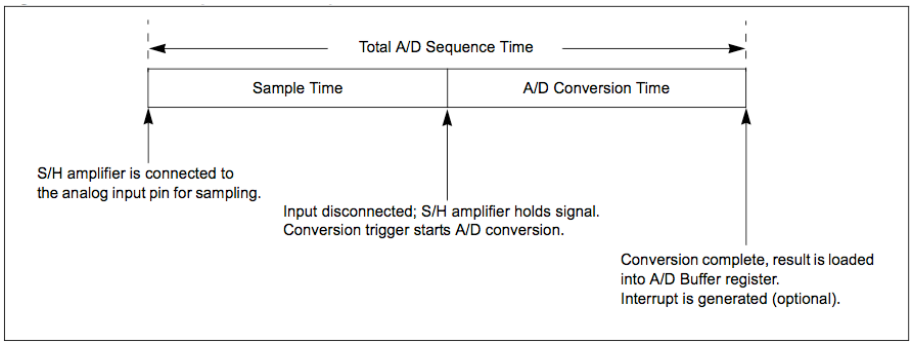


Figure 3.36: Total sequence time of the ADC
[Extracted from [MICRO2, p.4] with kind permission of Microchip]

The total conversion time for an input voltage can be seen in Figure 3.36. It consists of two periods of time: the time to sample an input signal and the time to convert the sampled signal into a digital value. Based on the configuration registers AD1CON3 the period of the conversion clock and the sampling time can be determined. The decision for a sufficient sampling time depends on the value of the holding capacitor and its surrounding impedances. The sample time must be long enough to charge the capacitor to the input voltage level. After the sampling time a trigger impulse (manual or automatic) is needed to stop the sampling and to start the conversion. The SAMC bits in the configuration register AD1CON3 are responsible for the sampling time in case of an automatic trigger impulse. Equation (30) calculates the resulting sampling time.

$$T_{SMP} = SAMC<4:0> * T_{AD} \tag{30}$$

To convert the sampled value in a 10-bit ADC result twelve conversion clock cycles are needed. Equation (31) explains how the conversion clock can be adjusted with the responsible configuration bits ADCS of the configuration register AD1CON3. A conversion clock of at least 75 ns must be provided.

$$T_{AD} = \frac{T_{CY} * (ADCS + 1)}{2} \geq 75ns \tag{31}$$

4 Software Realisation

This chapter contains the whole software of the system. That means both the firmware of the microcontroller and the graphical user interface (GUI) executed on the PC which is connected via USB. The firmware of the microcontroller is responsible for the chronology of the algorithm including the ADC timing and the switching of the matrix, the building of the USB communication with the connected PC and it decides on the basis of calculated values if a touch occurs. The GUI then again interprets the data provided by the microcontroller and presents the data graphically to the user. In addition the GUI allows the user to send some commands to the microcontroller (for example to choose an algorithm or a pattern) and also contains a code to implement the USB communication.

Apart from the technical implementation and realisation of the software on the two platforms (microcontroller and PC) this chapter explains exactly how the invented algorithm works. The chapter accents the logical structure of the algorithms and discusses the use of the generated light dependent signals in the according calculations. It defines the newly invented algorithm and underlines the differences to related approaches. Finally the handling of the GUI is explained, underlined with a screenshot of its skin.

4.1 Algorithms

The novelty of this thesis is, above all, the invented algorithm. In this context the term algorithm means the repeated toggling of the LEDs between light emitting and light detection, especially the essential switching of the LEDs in the matrix and the sampling and interpretation of the digitized output signals. For a comparison with related approaches an already used algorithm of Jefferson Han is described and implemented in subchapter 4.1.1.

The main challenges for an algorithm are the detection of a touch in random ambient light conditions and the distinction between a touch and an artificial light source. The novelty of the invented algorithm is that it additionally faces the challenge to simultaneously display a pattern. In subchapter 3.2 the different forms of incident light were already described. The LEDs have to interpret two different sources. At one hand there is the ambient light source, no matter if it is artificial or from the sun, and on the other hand the reflected light sourced by the matrix itself. Both of the light sources are detected by the LEDs. If a finger (or another reflecting object) comes near the surface of the matrix the ambient light is dimmed and the amount of reflected light increases: this has to be detected as a touch. The following Figure exhibits the behaviour of the amplifier output signal. Figure 4.1a presents the output signal of the amplifier under moderate ambient light conditions without covering the LED. The first half of the diagram indicates the output level without an emitting neighbor. In the second half a neighbor LED emits light. The output signal increases when the neighbor LED turns on due to the fact that an LED emits light in a wide angle. However in Figure 4.1b a finger (or any

object) covers the monitored LED. As a consequence the output level decreases, if no neighbor LED emits light, because the ambient light is diminished. In contrast an emitting neighbor increases the output level due to the occurring reflection at the covering object. The difference between the two phases of the output signal is significantly higher in case of a cover. Comparing these differences with a predefined threshold value allows to draw the conclusion whether a touch has occurred or not.

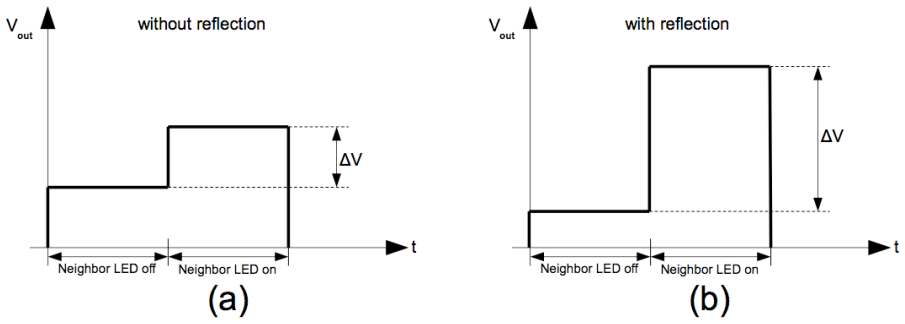


Figure 4.1: Behaviour of the amplifier output signal at typical light conditions

It may happen that a very bright light source illuminates the LED surface. Figure 4.2 deals with such ambient conditions. Figure 4.2a exhibits the output levels again without a cover. The output level of the amplifier is already in its upper limit. Therefore the additional light from an emitting neighbor is not able to increase the output level. Figure 4.2b again shows the amplifiers output level in case of a very bright ambient light but the monitored LED is once more covered. The reflection is irrelevant but when none of the neighbors emits light also the ambient light is dimmed. The difference of the two cases can again be calculated and a decision whether a touch has occurred can be made.

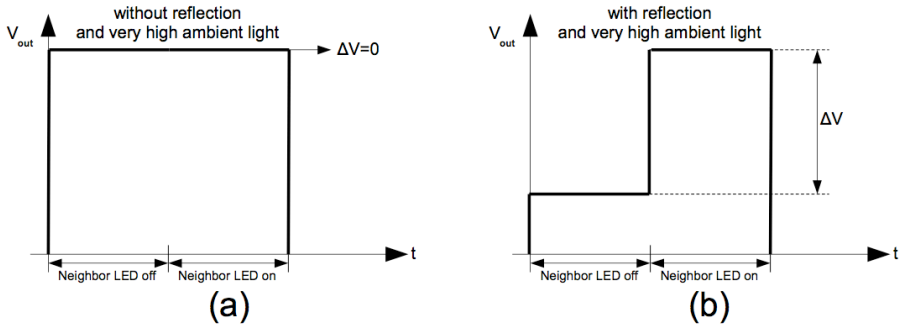


Figure 4.2: Behaviour of the amplifier output signal at bright ambient light

The algorithm has to calculate the differences of the output levels to detect a touch but first of all it has to drive the matrix in such a way that these different outputs can be measured at all. Thereby the algorithm drives the responsible switches and starts the sampling of the needed signals at a defined time. In the subsequent Chapter 5 the assumed voltage-time diagrams of Figure 4.1 and Figure 4.2 are verified by measurements.

4.1.1 Description of the “Han Algorithm”

As mentioned above one related algorithm is implemented in the microcontroller software. The description of the algorithm follows [HAN1] but not every claim and function of the patented algorithm is mentioned in detail. The algorithm for single-colour LEDs of Jefferson Han (Figure 4.3a and Figure 4.4) starts with a display scan sequence which means that LEDs

emit light depending on the desired output image given by the controller. After the scanning sequence the main part of the algorithm starts. At the beginning a counter for the columns is set to zero. The first column is connected to ground and all even rows are forward biased so each even LED of the column emits light. However, the odd LEDs now act as detectors and their output signals are sampled and stored. It is important to mention that every LED now has emitting neighbors. If a finger or an object covers an LED of this column light is scattered and reflected. The next step is to exchange the emitter and detector roles between the even and the odd LEDs of the column. The output signals of the even LEDs are also sampled and stored. Finally all LEDs of the column work as detectors without any emitting neighbors. Another array of values can be sampled and stored. The column counter increases and the next column follows the same procedure. Only when all columns have been scanned another normal display scan sequence is executed.

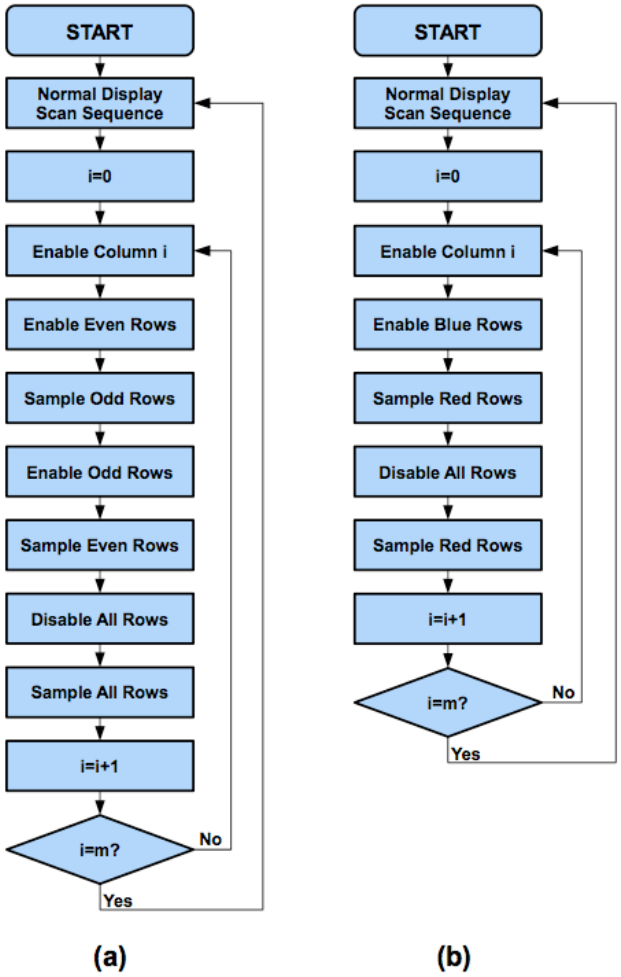


Figure 4.3: Flow chart of Han's algorithm for a) Single-colour LED b) Multi-colour LED
[Adopted from [HAN1]]

The purpose of this algorithm is that two different values for each LED are sampled and stored. One value sampled with emitting neighbors and one without separate light sources. Out of this data array it is possible to calculate if there was a finger or an object covering the matrix.

The difference of the second algorithm (Figure 4.3b) of J. Han for multicolour LEDs is that although every column must be scanned separately now there is no need to distinguish

between even and odd rows. The light is now emitted from one colour in the multicolour LED and another colour can do the detecting simultaneously.

Figure 4.4 illustrates pictorial how the single-colour algorithm works. Corresponding to the flow chart (Figure 4.3) the algorithm steps through the matrix column by column and takes three measurements each.

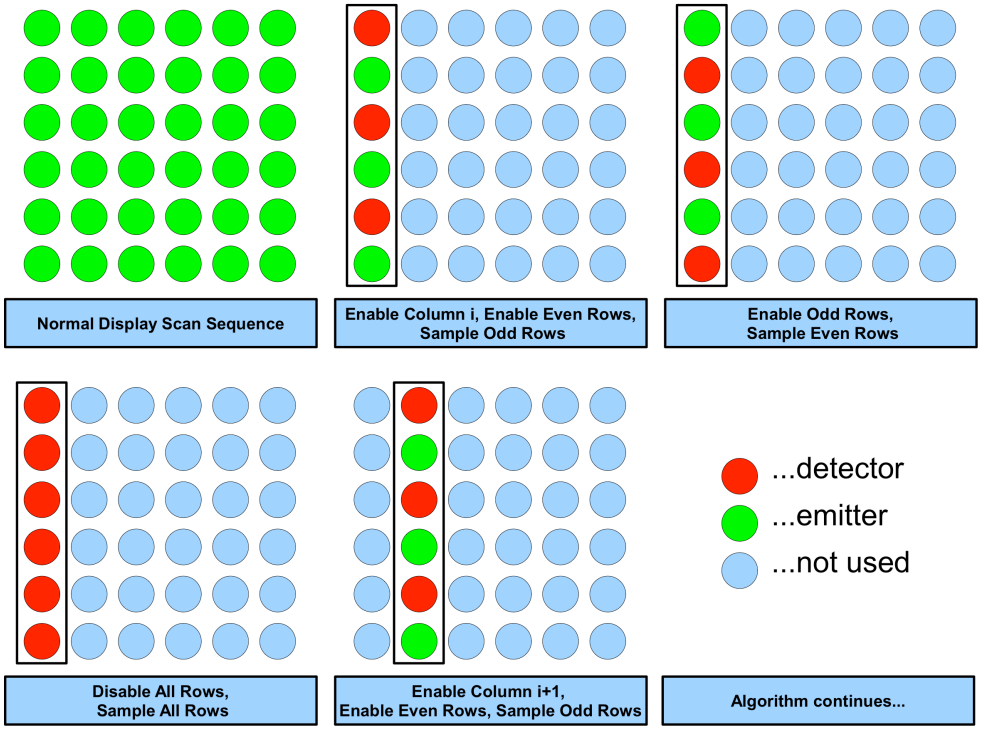


Figure 4.4: Pictorial diagram of Han's single-colour algorithm

For an even better understanding the necessary signals for the switches are shown in the subsequent diagrams in Figure 4.5. The first two diagrams correspond with the input signals of the switches between the LED anodes and the supply voltage. The following represents the switches from the cathodes to ground at each column.

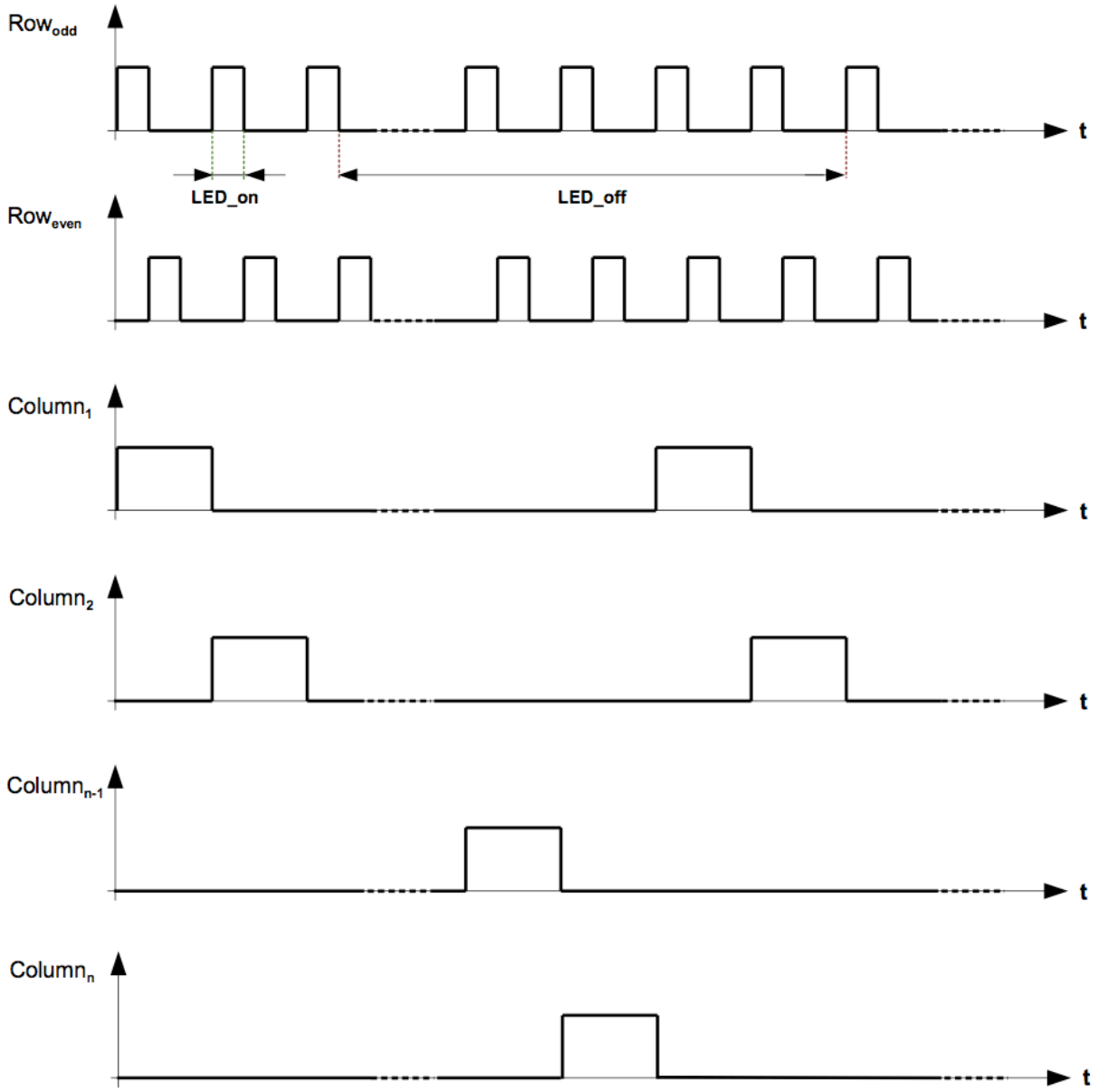


Figure 4.5: "Han Algorithm" timing sequence

The time diagrams indicate that the LEDs emit light only for a certain amount of time (responsible anode and cathode connections are set). When the LEDs do not act as emitting neighbors they stay dark. This leads to a duty cycle of the LEDs which determines the brightness of the matrix. In case of this algorithm the duty cycle depends on the number of used columns. For each column an LED emits light for a third of its active time multiplied by the number of columns. The resulting duty cycle is given by Equation (32).

$$D = \frac{1}{n_{columns} * 3} \tag{32}$$

4.1.2 Description of the "Pattern Algorithm"

In contrast to the discussed algorithms of Jefferson Han the now explained algorithm is able to show a pattern on the matrix while it simultaneously is monitoring the used LEDs. Of course the whole matrix can also be used as touch screen like in Han's algorithm. To meet this challenge the algorithm utilizes pairs of LEDs instead of a parity-based approach.

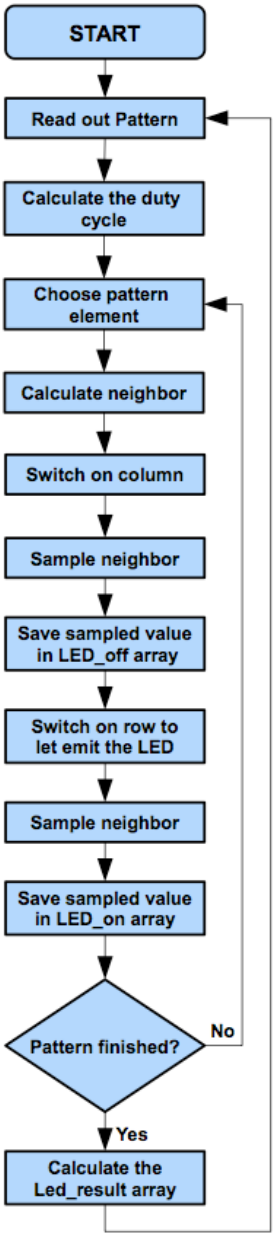


Figure 4.6: Flow diagram "Pattern Algorithm"

The flow diagram in Figure 4.6 gives a rough overview of the algorithm's timing sequence. At the start of the algorithm it is necessary that the pattern which is supposed to be displayed is saved. The pattern must be saved in any form to allow the algorithm to get information about the number of LEDs used in the pattern and which elements of the matrix exactly needed. Therefore the algorithm can calculate the duty cycle for each LED based on the number of used LEDs in the pattern. After this calculation the algorithm takes the first matrix element which is used in the pattern and calculates a possible neighbor. The criteria for being

one of the possible neighbors are that the neighbor has to be an element of the desired pattern and must be positioned in the same column as the actual LED. If these criteria are fulfilled and another LED does not already use the neighbor the neighbor-status is fixed. Based on that decision the calculated neighbor acts as detector. The column of the actual LED is connected to ground and the output signal of neighbor LED is sampled and saved in an array. Now the row which includes the actual LED is connected to the supply voltage to forward bias the LED and lets it emit light. The output signal of the neighbor is again sampled and is also saved in another array. After the value is sampled the row is again disconnected via the switches. This process repeats itself until every element of the pattern is set as active LED. The next pattern element is always the subsequent element in the same column. If the column is already finished the next column is used from top to bottom. Finally the saved values are used to calculate the difference between the two states of an emitting neighbor and non-emitting neighbor. Figure 4.7 explains the sequence again on the basis of individual states of the matrix.

Figure 4.7: Pictorial diagram of the "Pattern Algorithm"

It is obvious that the "Pattern Algorithm" needs more single steps through the matrix because only a single LED is used to emit light. To give a timing-diagram for the switches controlled by the algorithm like Figure 4.5 is a little bit more difficult because the states of the switches

strongly depend on the displayed pattern. Nevertheless, if it is assumed that every LED of the matrix is used in a pattern the minimal duty cycle of the algorithm can be calculated based on a timing-diagram (Figure 4.8).

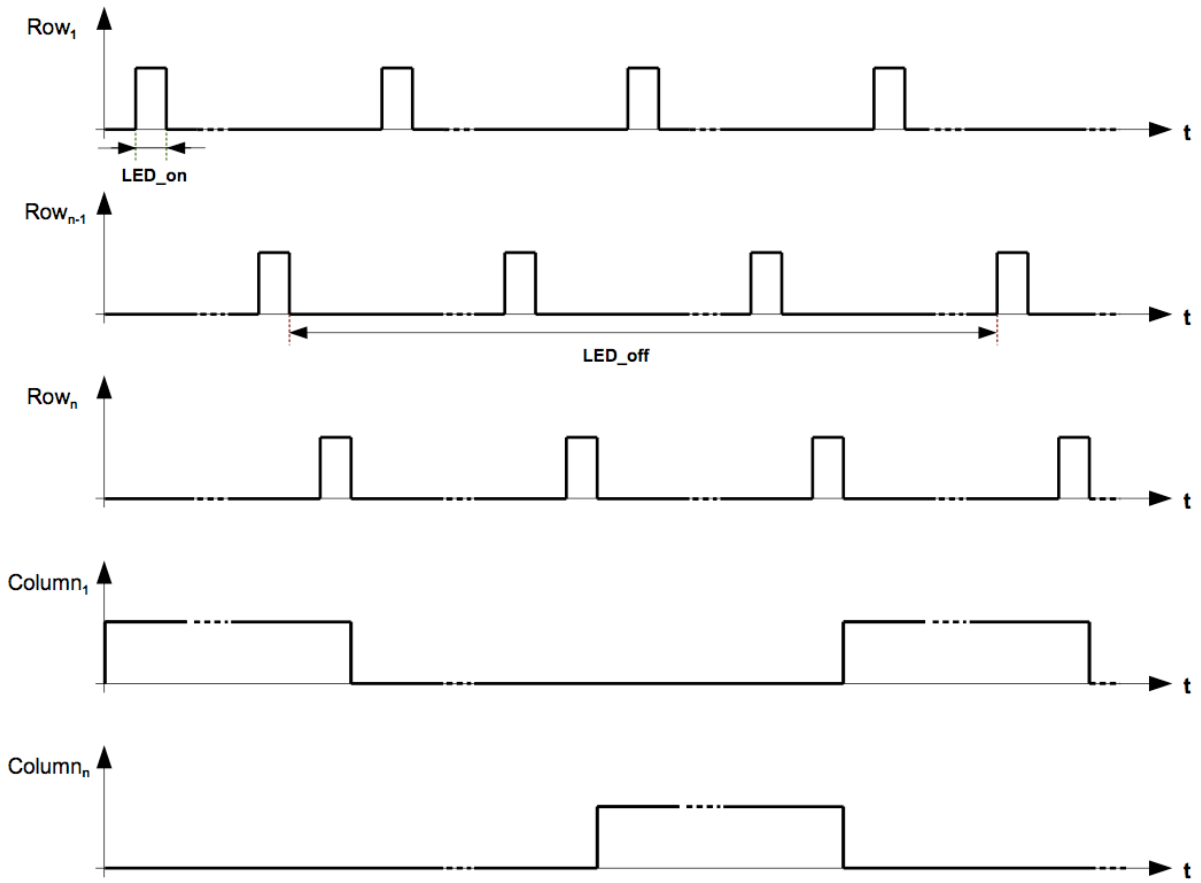


Figure 4.8: "Pattern Algorithm" timing sequence

$$D = \frac{1}{2 * n_{row} * n_{column}} \tag{33}$$

As one can see the duty cycle for each LED is really small. Equation (33) gives the exact connection of the matrix size and the duty cycle. To counter the decrease of the resulting brightness of the matrix some alterations are needed. The first possibility is an increase of the current. The LEDs are active just for a very short amount of time thus allowing a higher forward current and an increase of brightness. Another option is to reduce the sampling time of the ADC to a minimum. The application of this algorithm is not supposed to drive the full matrix but also patterns with a high number of used LEDs lead to a worse duty cycle and can use the mentioned alterations. A limitation of the number of LEDs allowed in the pattern is still reasonable.

If the whole matrix is driven with the "Pattern Algorithm" another modification can lead to an even better brightness behaviour. When the whole matrix is used the values of every row have to be sampled with and without emitting neighbors for each column. To decrease the sample time it is possible to sample the values for non-emitting neighbors for each column at the same time. This allows a much faster stepping through the matrix and thus gives a much better duty cycle. Equation (34) shows the arithmetic advantage of the increasing duty cycle.

$$D = \frac{1}{(n_{row} + 1) * n_{column}} \tag{34}$$

Despite of a smaller duty cycle the invented algorithm is still an improvement for the present embodiment. The algorithm shows any pattern and scans each of the used LEDs simultaneously. This allows, for example, for building a slider with two rows of the LED matrix and for recognizing if a user touches it and in which direction the user strikes. Another innovation used in this thesis is a touch detection related on the ambient light. The detailed implementation is explained in subchapter 4.2.5.

4.2 Microcontroller Software

Both of the explained algorithms are implemented on the microcontroller. The software is written in C-Code in the proprietary software tool *MPLAB 8.50*. The microcontroller software includes the hardware preferences for the used peripheral modules and the algorithm itself. Furthermore the communication via USB is implemented in the microcontroller software code. The hardware preferences and the algorithms have to be developed and coded from scratch. Whereas the USB communication can be superimpose on codes already created by colleagues.

To familiarize with the programming of the microcontroller the code was separated into single tasks. These tasks are the digital-to-analog conversion with the ADC module, the dealing with I/O ports and the USB communication. All tasks were programmed and implemented on a proprietary demoboard of *Microchip* to test each task separately. The individual tasks were combined and adapted to the present embodiment. The communication with the GUI has to be fixed and thenceforward the microcontroller software has to be developed in parallel to the GUI software on the PC.

Figure 4.9 gives an overview of the firmware structure executed on the microcontroller. The figure does not show all functions and connections in the software but gives a rough survey of the code structure and highlights the important functions.

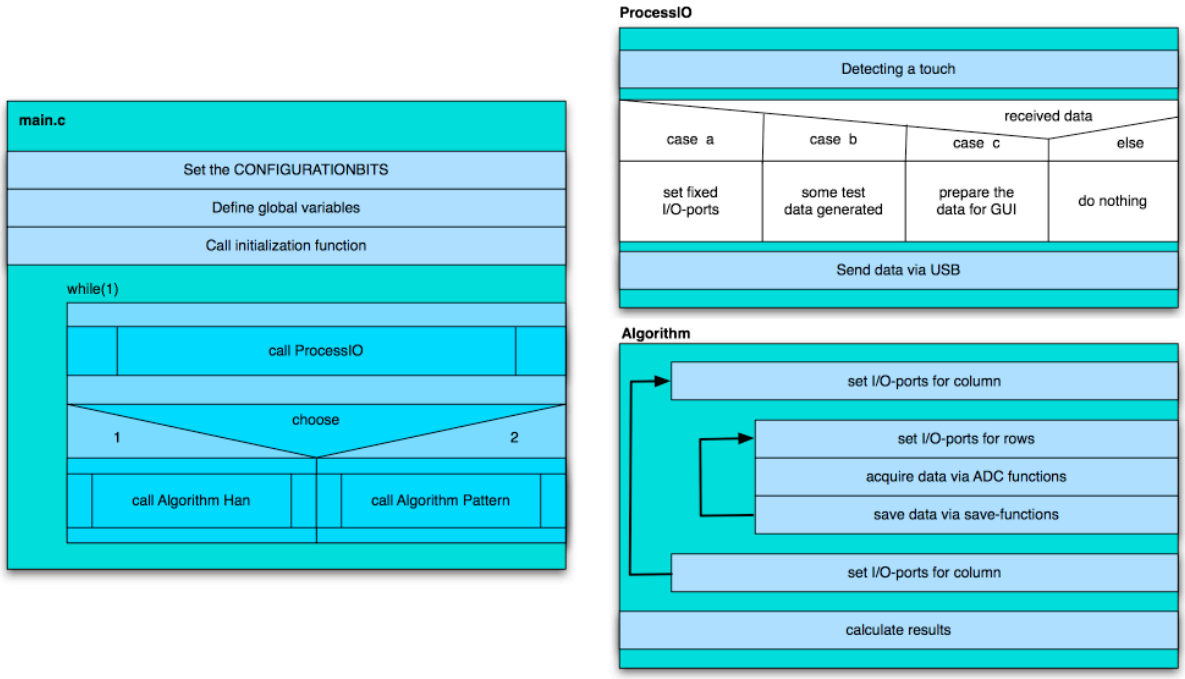


Figure 4.9: Rough overview of the important firmware parts

4.2.1 Hardware Settings

At the beginning of the microcontroller program some basic hardware settings can be determined. The PIC24FJGB004 allows these entries via configuration bits in front of the main source code. To provide the most important hardware adjustments four independent configuration registers are available. All modules of the microcontroller operate on the basis of the predefined configuration bits. For some of the used modules additional hardware settings are necessary. They will be described in the according sections.

The following configuration bits are set:

```
_CONFIG1(WDTPS_PS1 & FWPSA_PR32 & WINDIS_OFF & FWDTEN_OFF & ICS_PGx1 & GWRP_OFF & GCP_OFF &
JTAGEN_OFF)

_CONFIG2(POSCMOD_NONE & I2C1SEL_PRI & IOL1WAY_OFF & OSCIOFNC_ON & FCKSM_CSDCMD & FNOSC_FRCPLL
& PLL96MHZ_ON & PLLDIV_NODIV & IESO_ON)

_CONFIG3(WPFP_WPFP0 & SOSCSEL_IO & WUTSEL_LEG & WPDIS_WPDIS & WPCFG_WPCFGDIS & WPEND_WPENDMEM)

_CONFIG4(DSWDTPS_DSWDTPS3 & DSWDTOSC_LPRC & RTCOSC_SOSC & DSBOREN_OFF & DSWDTEN_OFF)
```

The configuration bits allow a huge number of settings but only the subsequently described settings are important for the present software. For a better readability of the code all configuration bits of each register are left in the code but most of them as default. The first configuration register defines which pins of the microcontroller are used for the debug interface (ICS_PGx1) and the fourth register is left in its default settings. In the second register the choice for the used oscillator is made. The primary oscillator is disabled (POSCMOD_NONE) and the internal FRC oscillator with a postscaler and its PLL mode is defined (FNOSC_FRCPLL). In addition the PLL96MHZ_ON bit is set to turn on the internal 96MHz PLL. Due to the use of the internal oscillator the pins to connect an external oscillator functions as normal I/O pins, set in configuration register one (OSCIOFNC_ON) and configuration register three (SOSCSEL_IO).

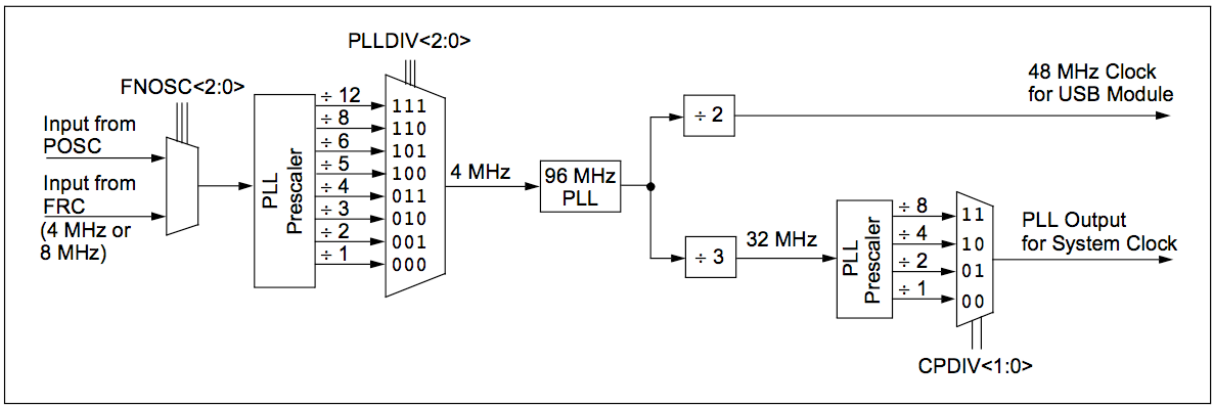


Figure 4.10: System Clock Generation
[Extracted from [MICRO1, p.114] with kind permission of Microchip]

Figure 4.10 follows the path of the internal FRC oscillator provided clock signal inside the USB PLL block. The configuration bit PLLDIV_NODIV is set in register two which forwards a 4 MHz clock to the input of the 96 MHz PLL block. The 96 MHz PLL is halved to provide a 48MHz clock for the USB module. The system clock is also generated out of the 96 MHz PLL. The postscaler CPDIV is left in its default setting which implicates a system clock of 32 MHz. Please note that the instruction cycle clock which is used for subsequent calculations has only half of the frequency of the system clock.

4.2.2 ADC Conversion

Six different ADC channels are used to provide one channel for each amplifier. On account of an easier PCB routing and in consideration of the needed I/O ports the ADC channels four to nine are used. The order of the ADC channels does not correspond to the amplifier order on the PCB. During the internal processing of the data which saved them in a memory buffer the logical order can be restored. Table 4 shows the connection of the ADC channels, the amplifiers numbered by their position on the PCB, the ADC buffer registers and the memory buffer.

ADC Channel	Amplifier	ADC Buffer	Memory Buffer
AN4	AMP 2	ADC1BUF0/ADC1BUF6	buffer[1]
AN5	AMP 3	ADC1BUF1/ADC1BUF7	buffer[2]
AN6	AMP 4	ADC1BUF2/ADC1BUF8	buffer[3]
AN7	AMP 5	ADC1BUF3/ADC1BUF9	buffer[4]
AN8	AMP 6	ADC1BUF4/ADC1BUF10	buffer[5]
AN9	AMP 1	ADC1BUF5/ADC1BUF11	buffer[0]

Table 4: Connections between ADC channels, amplifier, ADC buffer and memory buffer

Obviously the ADC is used in both algorithms. Therefore two different initialization functions are needed. Both functions set the voltage reference to an external reference and use the internal RC clock with a period of 250 ns. Each of the functions stores the ADC results in a single 16-word buffer in the integer format. The digitizing sequence of the ADC is based on the microcontrollers' ability of automatic conversion. This means a sample time T_{SAMP} and a conversion clock have to be set. Based on these settings the ADC generates the trigger signals for starting the sampling and conversion by itself.

To initialize the ADC in the desired form the following functions have to run before the ADC can be used to acquire analog values during the algorithm. The InitAllADC() function allows the ADC to sample all analog inputs from four to nine. It scans the defined input channels and generates an interrupt after the seventh sample/convert sequence to notify the end of the total conversion. The data is saved in the result buffer in the same order as the inputs are sampled (AN4…AN9, AN4). Though there are six input channels the interrupt is set until the seventh conversion sequence. This allows to dismiss the first (and always critical) sample to use the last one instead.

```
void InitAllADC()
{
        AD1CON1bits.ADON      =       0;      //switch off ADC module
        AD1CON1bits.FORM      =       0;      //set data format to integer
        AD1CON1bits.SSRC      =       7;      //auto convert
        AD1CON2bits.VCFG      =       3;      //set external voltage reference
        AD1CON2bits.CSCNA     =       1;      //scan inputs
        AD1CON2bits.SMPI      =       6;      //interrupt every 7th sample/convert sequence
        AD1CON2bits.BUFM      =       0;      //buffer in single 16-word results
        AD1CON2bits.ALTS      =       0;      //always use MUXA
        AD1CON3bits.ADRC      =       0;      //clock derived from system clock
        AD1CON3bits.SAMC      =       0x12;   //autosample time 18*Tad
        AD1CON3bits.ADCS      =       0xF;    //A/D Conversion Clock Select bits -> 8*TCY
        AD1PCFGbits.PCFG      =       0xFC0F; //set AN4-AN9 as analog inputs
        AD1CSSL               =       0x3F0;  //define inputs, which are sampled AN4-AN9
        AD1CHSbits.CH0NA      =       0;      //set negative input for MUXA on Vr-
        AD1CON1bits.ADON      =       1;      //switch on ADC module
}
```

In contrast the function InitPatternADC() samples only one input channel depending on the referred variable adcSet and uses other conversion clock settings.

```
void InitPatternADC(umword adcSet)
{
        AD1CON1bits.ADON      =       0;      //switch off ADC module
        AD1CON1bits.FORM      =       0;      //set data format to integer
        AD1CON1bits.SSRC      =       7;      //auto convert
        AD1CON2bits.VCFG      =       3;      //set external voltage reference
        AD1CON2bits.CSCNA     =       0;      //no input scan
        AD1CON2bits.SMPI      =       4;      //interrupt every 5th sample/convert sequence
        AD1CON2bits.BUFM      =       0;      //buffer in single 16-word results
        AD1CON2bits.ALTS      =       0;      //always use MUXA
        AD1CON3bits.ADRC      =       0;      //clock derived from system clock
        AD1CON3bits.SAMC      =       0x12;   //autosample time 18*Tad
        AD1CON3bits.ADCS      =       0x8;    //A/D Conversion Clock Select bits -> 9/2*TCY
        AD1PCFGbits.PCFG      =       0xFC0F; //set AN4-AN9 as analog inputs
        AD1CSSL               =       0x3F0;  //define inputs, which are sampled AN4-AN9
        AD1CHSbits.CH0NA      =       0;      //set negative input for MUXA on Vr-

        If(adcSet==1)
        ADC1CHSbits.CHOSA     =       9;      //set positive input for MUXA on AN9
        If(adcSet==2)
        ADC1CHSbits.CHOSA     =       4;      //set positive input for MUXA on AN4
        If(adcSet==3)
        ADC1CHSbits.CHOSA     =       5;      //set positive input for MUXA on AN5
        If(adcSet==4)
        ADC1CHSbits.CHOSA     =       6;      //set positive input for MUXA on AN6
        If(adcSet==5)
        ADC1CHSbits.CHOSA     =       7;      //set positive input for MUXA on AN7
        If(adcSet==6)
        ADC1CHSbits.CHOSA     =       8;      //set positive input for MUXA on AN8

        AD1CON1bits.ADON      =       1;      //switch on ADC module
}
```

After the successful initialization of the ADC module it can be used in the algorithm. As an example for the implementation two different functions which use the ADC to acquire data are shown. The first function SampleAllADC() converts the data of every amplifier stage output and saves them in the right order in the memory buffer as mentioned in Table 2. The required initialization function is InitAllADC().

```
void SampleAllADC(u16* buffer, u16 size)
{
	u16 *ADCPtr=NULL;
	ADCPtr=(u16*)&ADC1BUF0;		//set a pointer on the start of the ADC buffer

	IFS0bits.AD1IF		=	0;	//clear interrupt bit
	AD1CON1bits.ASAM	=	1;	//autostart sampling, time defined by SAMC4:0

	while(!IFS0bits.AD1IF);		//conversion done?
	AD1CON1bits.ASAM	=	0;	//yes, then stop sampling

	*(buffer+0) = ADC1BUF5;		//bring the ADC values in the right order
	*(buffer+1) = ADC1BUF6;		//take the seventh sample for ADC 2
	*(buffer+2) = ADC1BUF1;
	*(buffer+3) = ADC1BUF2;
	*(buffer+4) = ADC1BUF3;
	*(buffer+5) = ADC1BUF4;
}
```

The next function SamplePatternADC() needs the initialization function InitPatternADC() to work well. The function is very similar to the afore described function but it acquires only one amplifier output. The data is saved in the memory buffer according to the referred parameter *row*. Even this function dismisses the first sample saved in ADC1BUF0 and process ADC1BUF1.

```
void Sample_Pattern_ADC(umword row, u16* buffer)
{
	IFS0bits.AD1IF		=	0;	//clear interrupt bit
	AD1CON1bits.ASAM	=	1;	//autostart sampling

	while(!IFS0bits.AD1IF);		//conversion done?
	AD1CON1bits.ASAM	=	0;	//yes, then stop sampling

	*(buffer+(row-1))=ADC1BUF1;
}
```

4.2.3 Implementation of the Algorithms

In subchapter 4.1 the logical structure of both algorithms are described. One can see how the LEDs are switched and which output values have to be captured at a defined time. The present chapter now explains the technical implementation of the algorithms in the microcontroller firmware in detail. Especially how to set the I/O ports in the software and how to sample the output values of the amplifiers. Moreover, the calculation of the resulting values is an important part of the algorithm code. The switches addressed via PORTC provide the connection through the matrix and the supply voltage. PORTB is responsible for the matrix column connection to ground. The algorithms need always to different opamp output values for each LED and a result buffer. To save the values the two-dimensional arrays LED_on[6][6], LED_off[6][6] and LED_result[6][6] are already provided as global variables by the main function. LED_on[6][6] saves the values of the opamp outputs in case of emitting neighbors while LED_off[6][6] stores the values without an illumination of the matrix. To make the implementation as comprehensibly as possible examples of the algorithm code are quoted and explained in detail. For a better understanding it is suggested to have a look at Figure 4.9 to see how the algorithm functions are embedded in the whole firmware.

4.2.3.1 Implementation of the "Han Algorithm"

The only difference to the described logical sequence of the algorithm is that the technical implementation eschews the so called "normal display scan sequence". First of all the algorithm function secures that all switches surrounding the matrix are off. Then the first column is connected to ground and the subfunction Switch(umword, u16*, u16*) is called. The first parameter of the function is the number of the active column and the subsequent two are pointers on the arrays to save the according LED values.

```
void Algorithm3(u16* LED_off_Ptr, u16* LED_on_Ptr, u16* LED_result_Ptr)
{
	umword i;

//set all switches off
	PORTC   = PORTC & 0x207;
	PORTB   = PORTB & 0xDC4F;

	//first column
	PORTB = PORTB ^ 0x10;
	Switch(1,LED_on_Ptr, LED_off_Ptr);
...
```

Inside the Switch function the toggling between odd and even LEDs takes place. At the start the odd LEDs of the active column are switched on to emit light. A delay function provides an idle time to guarantee stable signals at the opamp inputs. The next step is to start the ADC to acquire the output signals and save them into an ADC buffer memory (adcValues). The detailed description of the ADC function can be found in subchapter 4.2.2. To save the sampled data in the responsible array (LED_on) another subfunction SaveLED_on(u8, u8, u16*, u16*) is called. The function has four parameters: One to indicate the active column, one to distinguish between even and odd LEDs and two pointers on the necessary arrays adcValue and LED_on.

```
void Switch(umword column, u16* LED_on_Ptr,u16* LED_off_Ptr)
{
	//switch on odd leds 1,3,5
	PORTC   = PORTC ^ 0xA8;

	//delay to guarantee stable inputs
	delayNMicroSeconds(200);
	SampleAllADC(adcValues, MAX_ADC_VALUES);
	SaveLED_on(column,1,adcValues,LED_on_Ptr);
...
```

The SaveLED_on function itself just copies the values of the adcValue array to the right storage position in the LED_on array depending on the active column and whether even or odd LEDs are emitting light.

```
void SaveLED_on(u8 column, u8 even, u16* adcValues, u16* LED_on_Ptr)
{
	umword i=0;

	if(even)
	{	//save the even LEDs of the column
		for(i=1;i<=5;i=i+2)
		{
			*(LED_on_Ptr+(6*i)+(column-1)) = adcValues[i];
		}
	}
	else
	{	//save the odd LEDs of the column
		for(i=0;i<=4;i=i+2)
		{
			*(LED_on_Ptr+(6*i)+(column-1)) = adcValues[i];
		}
	}
}
```

Back in the Switch function the odd LEDs are turned off and in contrast the even LEDs now emit light. Again the ADC acquires the data and the SaveLED_on function stores the data in the right storage position of the array LED_on. Now all LEDs are deactivated and the output signals are again acquired by the ADC function. At last another storage function SaveLED_off is called to save the data now in the array LED_off and finishes the Switch function.

```
void SaveLED_off_NEW(u8 column, u16* adcValues, u16* LED_off_Ptr)
{
	umword i;

	for(i=0;i<=5;i++)
	{
		*(LED_off_Ptr+((i*6)+(column-1))) = adcValues[i];
	}
}
```

In the underlying algorithm function the next column is set active and the discussed sequence starts again until every column is used. Now the software has saved two values for each LED in its arrays. Subtracting the results and saving them into a third array LED_result finalizes the algorithm function.

```
...
for(i=0;i<=35;i++)
	{
		*LED_result_Ptr = *LED_on_Ptr - *LED_off_Ptr;
		LED_result_Ptr++;
		LED_on_Ptr++;
		LED_off_Ptr++;
	}
}
```

4.2.3.2 Implementation of the “Pattern Algorithm”

The main difference in the implementation of the “Pattern Algorithm” compared to the previous algorithm is that the algorithm needs information about the desired pattern. In the default realisation the pattern is saved in a simple array. Each element of the array LED_button corresponds to an LED of the matrix. If an LED is used in the pattern the related element has the value one. The function Algorithm_Pattern3V0 which is called to execute the algorithm has five parameters altogether: three pointers on two-dimensional arrays for the measured values of the LEDs, a pointer on the LED_button array and a parameter to instuct the function which pattern has to be displayed.

Depending on the parameter *button* a switch-case statement decides which pattern has to be displayed. For an easier debugging the LED_button array is filled with the values for a desired pattern until then. The next step in the algorithm is to execute the function emit_LED3V0 for each element used in the pattern. The sequence starts with the first element of the array, finishes the column and goes on column by column.

```
void Algorithm_Pattern3V0(u16* LED_off_Ptr, u16* LED_on_Ptr, u16* LED_result_Ptr, u16*
LED_button_Ptr, umword button)
{
	umword i;

	//Initialize the delay timer
	delayInitialize();

	//depending on the value of button, one of the pattern is shown on the matrix
	switch(button)
	{
```

```
        case 0:
                {       ////////////////
                        //PLAY BUTTON//
                        ////////////////

                        //set the LEDs which are in the pattern
                        *(LED_button_Ptr+7)=1;

                        .....fill the all used elements with one....

                        *(LED_button_Ptr+31)=1;

                        emit_LED3V0(2,2,LED_off_Ptr, LED_on_Ptr, LED_button_Ptr);

                        .....execute the function for each pattern element...

                        emit_LED3V0(4,4,LED_off_Ptr, LED_on_Ptr, LED_button_Ptr);
```

The emit_LED3V0 function has the column of the element and its row as parameters. The pointers on the arrays of the measured values and the LED_button array are also parameters of the function. Inside the function the calculation of the neighbors starts. First of all the position of the referred pattern element is calculated and the potential neighbors over and under the element are set. Now the neighbors are tested whether are elements of the pattern or whether they are already used. When one neighbor fits all requirements the corresponding LED_button element is set on the value two.

```
void emit_LED3V0(umword row, umword column, u16* LED_off_Ptr,u16* LED_on_Ptr, u16*
LED_button_Ptr)
{
        umword neighbor_row=0;
        umword neighbor_pos_down,neighbor_pos_up,position=0;

        //calculate the array position of the emitting LED
        position=((row-1)*6)+(column-1);

        //calculate the next potential neighbors
        neighbor_pos_down=position+6;
        neighbor_pos_up=position-6;

        //which of the potential neighbors is in the pattern and not used
        if(*(LED_button_Ptr+neighbor_pos_up)>0)
        {
                //is the upper neighbor already used?
                if(*(LED_button_Ptr+neighbor_pos_up)==1)
                {       //neighbor is not used
                        neighbor_row = row - 1;
                        *(LED_button_Ptr+neighbor_pos_up)=2;
                }
                //neighboor was already used
                else
                {
                        //is there a neighbor down?
                        if(*(LED_button_Ptr+neighbor_pos_down)==1)
                        {       //yes theres a neighbor down
                                neighbor_row = row + 1;
                                *(LED_button_Ptr+neighbor_pos_down)=2;
                        }

                        //no there is no neighbor down
                        else
                        neighbor_row=7;
                }
        }
        else if(*(LED_button_Ptr+neighbor_pos_down)==1)
        {
                neighbor_row = row + 1;
                *(LED_button_Ptr+neighbor_pos_down)=2;
        }

        //no neighbor in the pattern available
        else
        neighbor_row=7;
....
```

After the neighbor is chosen the column of the referred pattern element is connected to ground and the actual value of the neighbor is sampled and saved into the LED_off array. To get also the value for the LED_on array the row of the element is connected to the supply voltage. Now the referred element emits light and the value of the neighbor can be sampled and saved. Finally the switches set back to disconnect the emitting LED.

```
...
//connect according column
switch(column)
{
	case 1:
	{
		PORTB   =	(PORTB & 0xDC4F) |    0x10;
	}
	break;
			....
}

if(neighbor_row!=7)
{	//measure LED_off value
	Init_Pattern_ADC(neighbor_row);
	Sample_Pattern_ADC(neighbor_row, adcValues);
	SaveLED_off_Pattern2V0(neighbor_row,column,adcValues, LED_off_Ptr);
}

//connect according row
switch(row)
{
	case 1:
	{
		PORTC   =	(PORTC & 0x207) | 0x8;
	}
	break;
			.....
}

if(neighbor_row!=7)
{
	//measure LED_off value
	Init_Pattern_ADC(neighbor_row);
	Sample_Pattern_ADC(neighbor_row, adcValues);
	SaveLED_on_Pattern(neighbor_row,column,adcValues, LED_on_Ptr);
}
```

At the end of the algorithm function the difference between the LED_on and LED_off array elements are calculated and saved in LED_results. The only contrast to the previous result calculation is that every unused element of the pattern is set to zero manually.

As mentioned in the discussion about the fundamental sequence of the “Pattern Algorithm” a second approach can be made if the whole matrix is used. In this case the calculation of the neighbors is much easier. A function similar to emit_LED3V0, so called emit_LED, gets the same parameters except the LED_button array and the variable *button*. In this case both of them are minor. The neighbor calculation inspects if the referred element is even or odd. Even elements get the nearest matrix element above in the column as neighbor and the odd elements get one underneath it.

```
void emit_LED(umword row, umword column, u16* LED_off_Ptr,u16* LED_on_Ptr)
{
	umword neighbor_row=0;

	//select the right neighbor
	if(row%2==0)
	neighbor_row = row - 1;
	else
	neighbor_row = row + 1;
```

The subsequent code is again very similar except for the sampling of the LED_off values. They can be sampled and saved for all LEDs in one column at the same time. This is the corresponding opportunity to improve the duty cycle of the LEDs in the matrix as mentioned above.

4.2.4 USB Communication

The USB communication is based on the USB Human Interface Device Class (USB HID class). This is a class defined by the USB-IF and used by many devices like mice, keyboards and many others. The big advantage of using the USB HID class is that the device driver is already implemented in most operating systems. The microcontroller acts as HID device and the PC as USB host. The software running on the PC sends interrupt requests to the microcontroller in a defined interval. The microcontroller software recognizes the request and answers. Of course there are a lot more details of how the USB communication works but during the realisation of this thesis a stable USB communication code already existed in the company and could be used. The thesis deals in more details what kind of data have to be sent via the USB connection and how these data have to be prepared for sending.

Guideline for the available USB communication is that a host (PC) asks for data with a 64-byte array (this array can also contain some commands) and the microcontroller answers equally with a 64-byte array. The data sent back by the microcontroller are boxed in a 64-byte array separated in 8-bit values. The first position of the array is reserved by the communication protocol, hence 63-bytes can be filled up with data and sent back each transmission. In the microcontroller software the received data is so called ReceivedDataBuffer and the sent array is so called SendDataBuffer. The challenge for the microcontroller software is to put all necessary data for the GUI into the 64-byte SendDataBuffer.

An infinite loop iterates in the main function of the microcontroller firmware, after all variables are declared and all initialize functions are executed. Inside this repeating loop the algorithm is executed as well as a function ProcessIO(). The ProcessIO() recognizes if a host sends data to the microcontroller via USB. In case of incoming host data (ReceivedDataBuffer) the function inspects the incoming array, executes according commands (e.g. set output ports) and may send back an array (SendDataBuffer).

As mentioned in the discussion of the algorithms the calculated values are saved in a two dimensional array, so called LED_result. The following part of the ProcessIO() function code is responsible for splitting up the LED_result array and repackaging the data into the SendDataBuffer array.

```
...
//get the LED values out of the 2D array and split it into upper and lower bits
	for(i=0;i<=35;i++)
	{
		UpperBitsADC[i] = (*LED_result_Ptr >> 8) & 0xFF;
		LowerBitsADC[i] = (*LED_result_Ptr) & 0x00FF;
		LED_result_Ptr++;
	}

//split the LED values again in lower and upper bits
	for(i=0;i<=35;i++)
	{
		LowerUpperBitsADC[i] = UpperBitsADC[i] & 0xF;
		LowerLowerBitsADC[i] = LowerBitsADC[i] & 0xF;
		UpperLowerBitsADC[i] = LowerBitsADC[i] >>4;
	}

//for transfering the ADC values in one package, cut out the unused upper 6 bits from the ADC
	for(i=0;i<=53;i++)
	{
		Send[3*i]   = (LowerUpperBitsADC[2*i]<<4)   | UpperLowerBitsADC[2*i];
		Send[3*i+1] = (LowerLowerBitsADC[2*i]<<4)   | LowerUpperBitsADC[2*i+1];
		Send[3*i+2] = (UpperLowerBitsADC[2*i+1]<<4)| LowerLowerBitsADC[2*i+1];
	}

//repackage the data in the SendDataBuffer
	ToSendDataBuffer[0]   = 0x13; //send back the received command first

	for(i=0;i<=53;i++)
	{
		ToSendDataBuffer[i+1] = Send[i];
	}
...
```

As a result of the code segment above and its including shift operations the SendDataBuffer is filled with 12bit of data for each LED. The content of each element of the LED_result array which contains the data in the integer format is split. This data is again divided to be able to cut out the elements which do not contain needed values. This can be done because there are six unused bits filled up with zero in front of the 10bit integer values. The remaining data is packed into the SendDataBuffer. This totals up to 432bits for 36 LEDs and is completed to 440bits by hooking up the received command as first element of the SendDataBuffer. Table 5 pictures the transferred data package to give an idea which kind of data is available for the receiving USB host (PC).

Command	Value LED 1	Value LED 2	...	Value LED 35	Touch detection
xxxx xxxx	00xx xxxx xxxx	00xx xxxx xxxx	...	00xx xxxx xxxx	0x00 or 0xAA

Table 5: SendDataBuffer

4.2.5 Detection of a Touch

Another important code section of the ProcessIO() is the detection of a touch. Inside the function the average of all 36 LED_result values of the matrix are built. A subsequent request asks if one LED value of the LED_result array is significantly higher than the average. When the request is complied a touch must have happened. With the help of this request it is possible to reduce the influence of ambient light once again. As an additional safety check another request asks if the according element of the LED_off array is small enough to draw the conclusion that an object covers the LED. This second safety check assures that no light source spuriously initiates a touch. To transfer the indication of a touch to the GUI the sent data package is enhanced with two command-bits (Table 5).

```
        u16* startAdr;
        startAdr=LED_result_Ptr; //save the start adress of the Led_result array


        //calculate the average of the matrix values
        for(i=0;i<=35;i++)
        {
          sum = sum + *LED_result_Ptr;
          LED_result_Ptr++;
        }

        average = sum/36;

        LED_result_Ptr = startAdr; //set back the pointer to the first element of the array

        for(i=0;i<=35;i++)
        {
          if(*LED_result_Ptr > (average + 0x64) && *LED_off_Ptr < 0x10)
          {
               touch = 1;
          }

          LED_result_Ptr++;
        }
```

With the help of recognizing a touch it is possible to switch between patterns. An interactive button can be built. That means a pattern is displayed on the matrix and after a touch is recognized the pattern changes. It is necessary to avoid the interactive button from bouncing. To ensure this two global variables are needed. The help variable count_com and the variable play_count_com which are responsible for the displayed pattern are used. The following code deals only with the pointer on these variables because they are global. If no touch is detected the count_com variable saves the pattern status. In case of a touch an additional request asks if the status has changed. Only if the status has not changed the pattern switches. Therefore a finger touching the matrix's surface and staying there does not lead to a toggling of the pattern.

```
...
//no touch?
if(touch==0)
{
        *count_com=*play_count_com;
}
//new touch?
if (touch==1 && (*count_com==*play_count_com))
{
        //change pattern
        *play_stop_com=!(*play_stop_com);
        //debouncing
        delayMilliSeconds(100);
}
...
```

4.3 Graphical User Interface

The logical structure of the GUI is programmed in Visual C++ on the basis of a graphical interface for the C++ class library QT from Nokia. The main software structure for the GUI was adapted from already existing projects in the company. Thence the USB communication was ready for use and the rough structure of the graphical interface in QT. The main tasks of the GUI are to request and to import data via USB. The GUI processes the received data for an appealing representation of them and informs the user if a touch occurs. The user is also able to use the GUI as a back channel to the microcontroller to send some commands.

For a better understanding it is good to know about the procedure of how QT and Visual C++ work together. First of all the graphical interface is designed in QT. In QT itself it is possible to design the ancestor of the interface like buttons, titles and so on. The interface has to be implemented into an existing Visual C++ project. This project is not described in every detail because it was already available from former projects of colleagues. Please note that the existing project provides a lot of communication possibilities, predefined class structures, a main window for the graphic interface, register maps and a logger to deliver interesting feedback of the actual executed code. For the present GUI one of the communication channels (USB) is important as well as the main window of the interface where a newly created tab can hooked up. The chronological of the GUI follows the important aspect that QT works with so called signals and slots. Signals can be generated in nearly every section of the QT program. For example a signal can be forced via an interaction of a user or a timer which is passed by. Predefined slots in the QT program code then again capture the signals. A rough periphrasis could be the comparison of this implementation with jumps in ordinary C-code. The slots themselves which capture the generated signals execute a related program code.

4.3.1 Internal Succession

The origin of the GUI is the already existing Visual C++ project. At the beginning (main function) the communication is instantiated and the main window as well as the logger are implemented. Inside the code for the main window where some superior signal-slot connections are created and the graphical interface is created with QT is implemented as a new tab via the main window. More details about the superior project code are not relevant for this thesis. All subsequent code discussions apply on the newly created tab. The code for the tab includes all further interesting code segments for the present application.

First of all a new thread is initialized and the connection between used signals and slots are built. In addition some variables are set to zero at the start of the thread and the communication class is available via the former instantiated instant. The signals updateAllADC() and updateLED() are used for some test applications and play only a tangential role. Hence the signal updateButton() is important for the upcoming software and are discussed in detail. The associated slot which catches the updateButton() is doButtonUpdate(). The code within this slot is executed every time the signal is generated.

```
MaSTab::MaSTab(AMSCommunication* com, QWidget *parent) : QWidget(parent)
{
	int i;
	this->setupUi(this);

	//communication instant
	this->com = com;

	//Thread init
	this->masThread = new MaSTabThread();

	connect(this->masThread, SIGNAL(updateAllADC()), this, SLOT(doAllADCupdate()));
	connect(this->masThread, SIGNAL(updateLED()), this, SLOT(doLEDUpdate()));
	connect(this->masThread, SIGNAL(updateButton()), this, SLOT(doButtonUpdate()));

	this->masThread->start();

	//initialize variables with zero
	for(i=0;i<=5;i++)
	{
		this->row[i]=0;
		this->column[i]=0;
	}

	for(i=0;i<=35;i++)
	{
		this->oldADCValue_three[i]=0;
		this->LED_dif[i]=0;
	}
}
```

Another important function of the related GUI code is the paintEvent (QPaintEvent *). It is responsible for additional drawings inside the tab. In the actual case the function draws two matrices in the tab based on the values of the according variables. The first matrix shows the status of LED matrix when using the test functions of the GUI to set some switches static. The second matrix is more interesting because it displays the status of the sensed LED matrix, in case of a running algorithm. Every time a function calls the function update() the paint event is executed and the matrices are redrawn.

Back to the succession of the GUI after the first execution of the paintEvent() the function run() is started. The function includes an infinite loop. Inside the loop every 100 ms the signal updateButton() is generated. As described the slot doButtonUpdate() is responsible to catch it. The slot doUpdateButton() forces an USB interrupt and processes the incoming data. The upcoming subunit 4.3.3 discusses the slot in detail.

```
void MaSTabThread::run()
{
	while(1)
	{
		emit updateButton();

		//wait for 100ms before generating a new signal
		msleep(100);
	}
}
```

4.3.2 USB Communication

As mentioned above the communication structure already exists from former projects and as explained in the discussion about the microcontroller software the basis of the USB communication is the HID-class. To demand new data at the necessary positions in the GUI an USB interrupt has to be generated. This interrupt is caught in the microcontroller software and answered with a 64byte data package (look at Table 5).

For example the subsequent code section generates an interrupt and sends two commands to the microcontroller. In this case a command complies a simple four-bit value interpreted by the microcontroller software.

```
command=”SetOutputGetInputCombinationReport_Interrupt(00130003)(00)”;
this->com->sendCommand(command, &answer);
```

The received data package is stored as a string and can be addressed via the variable answer. For the present GUI software the received data is useless in the string format therefore further processes with the input data are needed. Subchapter 4.3.3 explains the ongoing steps and how the data is displayed.

4.3.3 Data Evaluation

To use the data which are provided by the microcontroller first of all an interrupt has to be generated to request the desired data as described in the former subchapter. The signal updateButton() is generated and caught by the slot doButtonUpdate(). The slot itself executes the code for further process of the data.

Due to the possibility of the GUI to decide which kind of algorithm shall run on the microcontroller some if-requests interpret the user-choice and sends the USB-HID interrupt with the according values to the microcontroller.

```
void MaSTab::doButtonUpdate()
{
	QString command, answer;
	QString valueADC[36];
	int i=0,j=0,k=0, l=0;

	//depending on user's choice, a bitstream according to an algorithm will be sended

	if(this->choose_algorithm==1)
	{
		command="SetOutputGetInputCombinationReport_Interrupt(00130001)(00)";
		this->com->sendCommand(command, &answer);
	}

..... ulterior if-requests.....

	else if(this->choose_algorithm==4)
	{
		command="SetOutputGetInputCombinationReport_Interrupt(00130004)(00)";
		this->com->sendCommand(command, &answer);
	}
....
```

After the interrupt is sent to the microcontroller its answer is received and can be addressed via the string *answer*. If a touch is detected by the microcontroller firmware two additional command bits are transferred via USB. The subsequent code section inspects the incoming data to indicate a possible touch. In case of a transferred command bit the GUI reports the indication of a touch by displaying the word “TOUCH” in big red letters.

```
....
	//look if there was a touch detected, and if the uP sends the according value
	if(answer.mid(112,2).toInt(0,16)==170) //yes?
	{
		le_touched->setText("TOUCH");    //tell me about the touch
		label_touched->setFixedHeight(50);

	}

	//no touch?
	if(answer.mid(112,2).toInt(0,16)!=170) //no touch detected?
	{
		le_touched->setText("NO TOUCH");//tell me about "no touching"
		label_touched->setFixedHeight(0);
	}
....
```

The next step is to extract the LED values out of the *answer* string and save them in an integer array. To provide an output of the values in form of text the now available integer values have to be multiplied by the factor 2.5 to convert them into mV. If a user clicks on the calibration button some additional calculations are performed for smoothing the graphical output of the LED data. A detailed discussion about the calibration can be found in the upcoming subchapter. Another opportunity to improve the display of the LED values follows. To decrease the toggling of the output the already displayed values are stored in an array. Only if the fetched data differs more than 100 mV from the actual displayed data the graphical output is updated. For the graphical output the LED values are saved in an two-dimensional array led_value[l][j].

```
....
	//get the values out of the bitstream
	for(i=0;i<=35;i++)
	{
		valueADC[i]=answer.mid((4+(i*3)),3);
		//calculate the Values of each LED in mV
		value_mV[i]=valueADC[i].toInt(0,16)*2.5;
	}

	//substract the difference to the average value
	for(i=0;i<=35;i++)
	{
		value_mV[i]= value_mV[i] - LED_dif[i];
	}

	//stop the toggeling of the display by just show differences higher than 100mV
	for(i=0;i<=35;i++)
	{
		if(abs(value_mV[i] - this->oldADCValue_three[i]) > 100)
		{
			le_LED11->setText(QString::number(value_mV[0]));
			le_LED12->setText(QString::number(value_mV[1]));

			....

			le_LED65->setText(QString::number(value_mV[34]));
			le_LED66->setText(QString::number(value_mV[35]));

			k=0;
			for(l=0;l<=5;l++)
			{
				for(j=0;j<=5;j++)
				{
					this->led_value[l][j]=(valueADC[k].toInt(0,16));
					k++;
				}
			}
....
```

Finally the paintEvent() is again executed to update the displayed output. If the values are displayed they are saved for comparison with the input data fetched next.

```
...
                        //draw the tab again
                        update();

                        //save the LED values, if they are already displayed
                        for(i=0;i<=35;i++)
                        {
                                this->oldADCValue_three[i]=value_mV[i];
                        }
}
```

Let's take a look at the code of the paintEvent(). Note that the subsequent code section is not the full paintEvent() but mirrors the important part for the graphical output of the LED values. The values of the LEDs are saved in a two-dimensional array. Only if one of the LED values is 100 mV higher than the average of all values the displayed data is updated. In plain language this means only if an object covers the matrix a graphical output starts. Otherwise the whole matrix is greyed. For the graphical output the LED values are converted into the HSV colour space. The value of each LED now corresponds to the hue of the displayed matrix element.

```
...
for(i=0;i<=5;i++)
{
        for(j=0;j<=5;j++)
        {
                if(led_value[i][j]>=(this->average+100)/2.5)
                {
                        for(i=0;i<=5;i++)
                        {
                                for(j=0;j<=5;j++)
                                {
                                        if(led_value[i][j]<=((this->average+70)/2.5))
                                        greenOff.setHsv((this->average/2.5),255,255);

                                        else
                                        greenOff.setHsv(this->led_value[i][j]+60,255,255);

                                        painter.fillRect(led[i][j],greenOff);
                                }
                        }
                }
        }
}
```

4.3.4 Calibration

The GUI provides the user with a calibration button. The button represents a method to improve the graphical output. If the button is clicked the average of all values is calculated. The average is subtracted from each value to get an array with the differences between average and actual value (at the time the user presses the button).

```
void MaSTab::on_pB_Calibration_clicked()
{       int i;

        this->average=0;

        //build the sum of all LED values
        for(i=0;i<=35;i++)
        {
                this->average = this->average + this->value_mV[i];
        }

        //calculate the average value of all 36 LEDs
        this->average = this->average/36;

        //display the average value
        le_Calibrate->setText(QString::number(this->average));

        //calculate the difference between each LED value and the average
        for(i=0;i<=35;i++)
        {
                this->LED_dif[i]=this->value_mV[i] - this->average;
        }
}
```

In the formerly discussed slot doButtonUpdate() is the calculated LED_dif array already used. The incoming LED values are subtracted by the corresponding LED_dif array. The procedure smoothen the whole output matrix. This can be very helpful if the LED matrix is covered by a transparent wafer which may diffuse the self-emitted light of the matrix and may lead to unwanted reflections. The code segment of the slot doButtonUpdate() follows again to show the influence of the calibration sequence on the output data. The subtraction is done two times: once for the text-output of the values and a second time for the graphical output.

```
//substract the difference to the average value
        for(i=0;i<=35;i++)
        {
                value_mV[i]= value_mV[i] - LED_dif[i];
        }

....

k=0;
        for(l=0;l<=5;l++)
        {
          for(j=0;j<=5;j++)
          {
            this->led_value[l][j] = (valueADC[k].toInt(0,16))-(this->LED_dif[k]/2.5);
            k++;
          }
        }
```

4.3.5 GUI Manual

The skin of the GUI is pictured in Figure 4.11. The focus of the interface is still on the development of algorithms and provides a lot of test applications and output for engineers. The best way to describe the interface is to handle through its in- and output possibilities. The user can choose between four different algorithms as described above. After the algorithm is selected via one of the checkboxes the corresponding application starts immediately. The matrix in the lower left corner provides a graphical output of the LED values calculated by the algorithm. The colour of the matrix elements pictures their value. The nearer an object comes to the surface of the matrix the higher the calculated value mirrored in the colour (especially the hue of the HSV colour space) of the element. In addition a text-based output matrix follows on the right side. The output values correspond to the calculated LED values in mV.

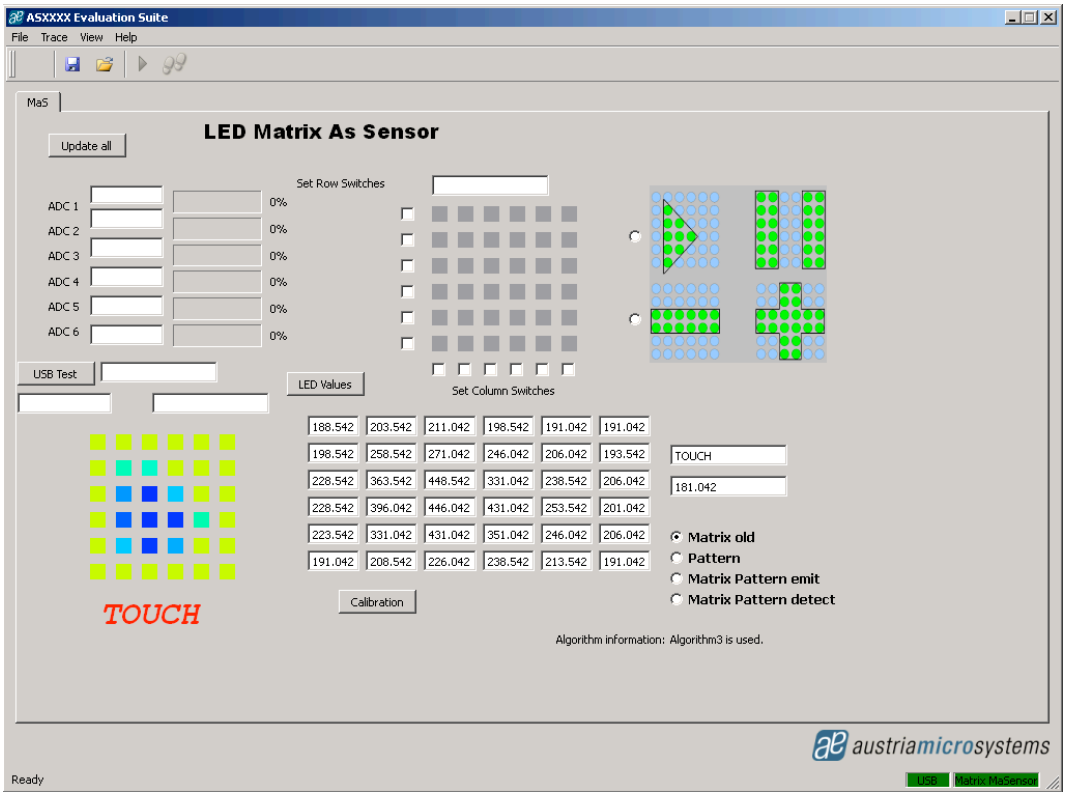


Figure 4.11: Skin of the Graphical User Interface

The calibration button allows an improvement of the graphical output. The values in the text-based matrix which are displayed after some calibration do not correspond to the measured values. This output can be neglected in case of a calibration. A detected touch is visualized in big red letters under the matrix and in a textbox. The surrounding checkboxes of the upper matrix provide a static access to the switches of the matrix if no algorithm is running. This may support some measurement applications on the PCB. The upper matrix itself only shows if an LED is switched on statically. The buttons “USB Test” and “LED Values” are also used for test applications. The textboxes in the upper left corner are able to show the actual captured ADC values after a click on the “Update All” button and in addition a progress bar for each ADC underlines the monitored values graphically.

5 Results

This chapter underlines the discussions above with relevant measurements and shows interesting signals of the present embodiment. At the beginning a measurement of the used SMD LED [KING1] underlines the postulated connection between incident light and generated photocurrent. The next section provides a deep inspection of the microcontroller output signals which control the surrounding switches of the matrix. The measurement results are completed with the output signals of the opamp stages and the voltage level at the cathodes of the matrix columns. The amplifier output signals are important on that score; they have to be sampled by the ADC of the microcontroller. At the end of the chapter the advantages of the novel algorithm are summarised and finally remaining weak points of the system are addressed.

5.1 Photocurrent Depending on Illumination

The basis of the whole system is the ability of LEDs to generate a photocurrent out of incident light. As postulated in the theory chapter, the more light impinges on the LED surface the more photocurrent is generated. A measurement setup with a defined light source (40 W light bulb) fixed in a distance of 20 cm to an LED surface in a lightproof box provides the general conditions to measure the characteristic curve of the LED's photocurrent corresponding to the illumination level. The measurement exhibits a linear relationship between the photocurrent and the illumination level.

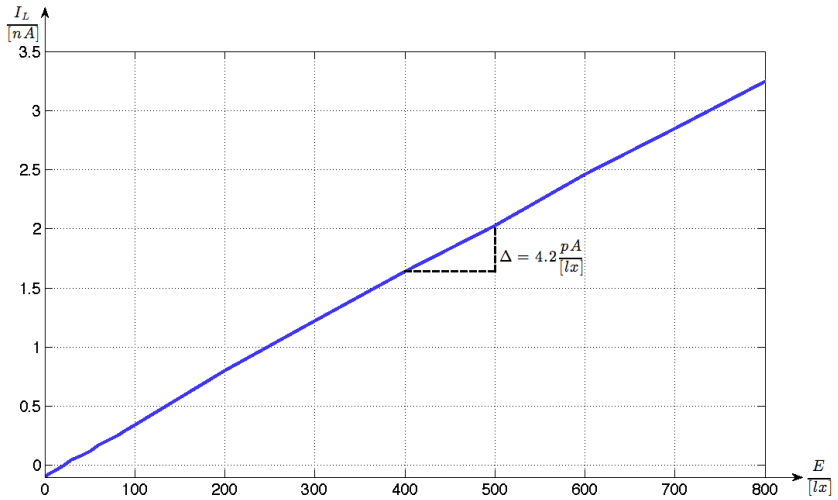


Figure 5.1: Current to illuminance level

The characteristic curve in Figure 5.1 shows the photocurrent only for illuminations up to 800 lux. According to [THEISS1] illuminance levels up to these values are common for indoor lighting like offices or living areas. Due to the fact that the present embodiment is also composed for outdoor operations also much higher illuminance levels may occur. Daylight on

a sunny day can reach levels of several thousands of lux. Assuming a linear progress of the measured curve and including the maximum output voltage of the amplifier stage leads to a maximum converted illumination level of round about 60.000 lx. The arithmetic derivation of this assumption is given by Equation (35).

$$\Delta = 4.2\,{}^{pA}\!/_{lx} \;\; V_{out,max} = 2.5V \rightarrow E_{convert,\max} = \frac{250\,nA}{4.2\,pA} = 59524\,lx \tag{35}$$

Further measurements with the same setup but with additional emitting neighbors lead to the same slope of the photocurrent and a parallel move of the whole curve by the amount of the neighbor's illuminance. As one can see in Figure 5.2 the emitting neighbors lift up the curve round about 20 nA.

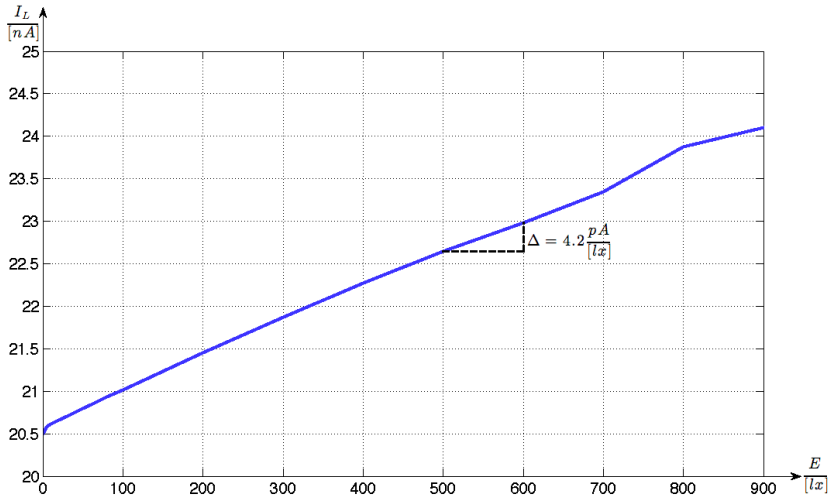


Figure 5.2: Current to light illuminance level; emitting neighbors

5.2 Switching Signals

The algorithms discussed in subchapter 4.1 are underlined with according timing diagrams. To confirm these diagrams and point out the actual timing of the switching signals the upcoming measurements show the input signals of the switches provided by the microcontroller. As explained the timings of the algorithms are different therefore measurements for both algorithms are done and pictured.

The first measurements display some of the switches' input signals of the "Han Algorithm". The subsequent Figure 5.3 shows the input of the first column switch against ground (CH1) and for the last column switch (CH2). The curves underneath are the inputs of the even and odd rows (CH3-even; CH4-odd). Every time a column is active the even rows are connected followed by an activation of the odd rows and finally both rows are disconnected.

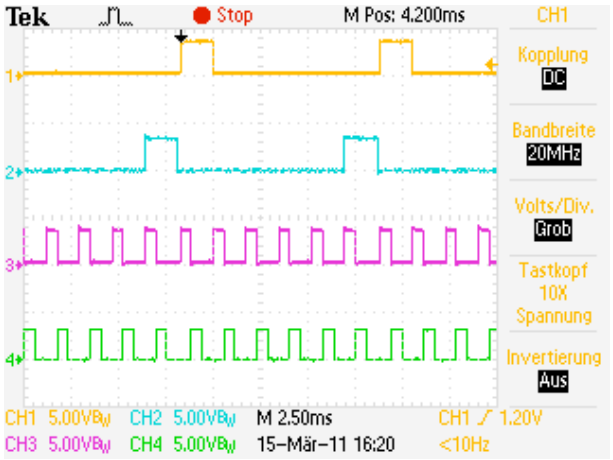


Figure 5.3: Actual timing of the "Han Algorithm"

Figure 5.4 zooms out one cycle of an active column. The column is high for about 1.73 ms. Each row is active for 575 µs and both are disconnected for the same amount of time. Inside the algorithm code a delay of 400 µs is set before the ADC samples its input signals to allow a discharge of the parasitic LED capacitance and guarantee stable input signals. This leads to a sample time of round about 175 µs for all six input channels. Each channel is therefore sampled in about 30 µs. This timing corresponds with the configuration of the ADC module in subchapter 4.2.2. As every column has to be sampled the algorithm needs 10 ms to fetch the data of the whole matrix.

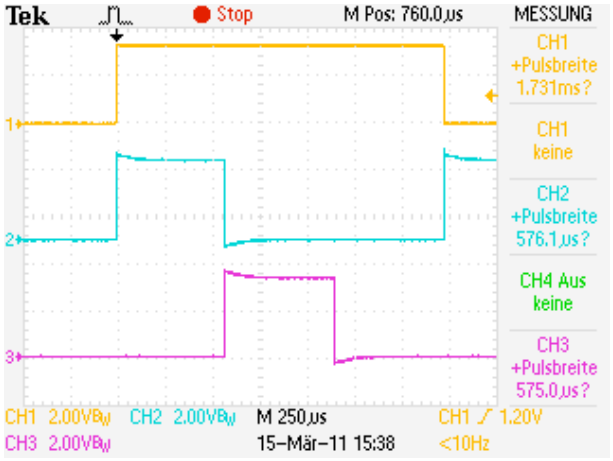


Figure 5.4: One active column cycle of the "Han Algorithm"

The timing diagram of the "Pattern Algorithm" strongly depends on the desired pattern shown on the matrix. To allow a comparison to the "Han Algorithm" the whole matrix is used for the pattern so again every LED emits light. In Figure 5.5 the inputs of the first (CH1) and the last (CH2) column switches towards ground are displayed. The other channels are the first (CH3) and the last (CH4) row switch inputs. As mentioned above a worse duty cycle is the consequence of using the whole matrix for the pattern.

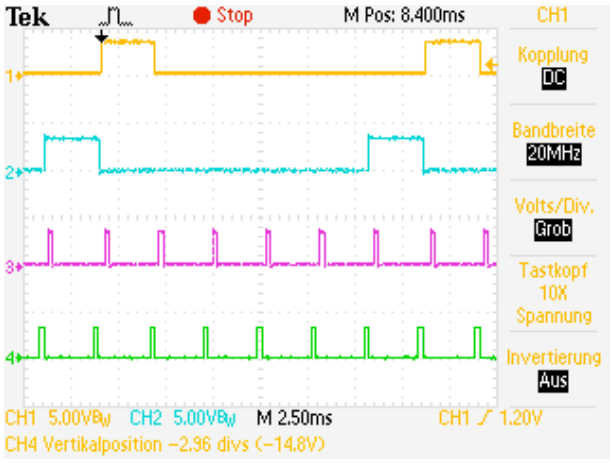


Figure 5.5: Actual timing of the "Pattern Algorithm"

Figure 5.6 zooms out the time of one active column (CH1). While the column is active every LED of the column is active for one time to emit light. As an example the first (CH2) and the second (CH3) rows are measured. While a row is disconnected the value of its neighbor can be sampled. After that, the switch is connected to let the LED emit light and the neighbor LED is again measured. In this algorithm the ADC samples only one channel at each execution. A delay of 200 µs is set before the ADC samples the value to guarantee again stable input signals. This leads to a total sample time of 34 µs per channel. The timing corresponds to the desired configuration in subchapter 4.2.2. Owing to the fact that now only one channel is sampled the fetching of the whole matrix values totals up to 17 ms. Please consider, that this is the worst case for the "Pattern Algorithm". The less LEDs are used for the pattern the better the duty cycle is.

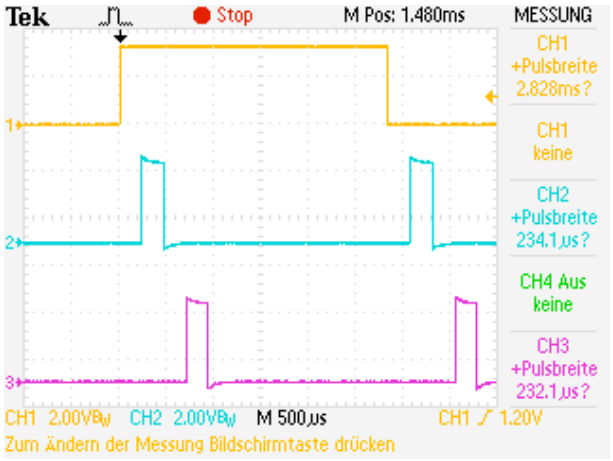


Figure 5.6: One active column cycle of the "Pattern Algorithm"

It is a little bit complicated to show the timing of the "Pattern Algorithm" due to its strong dependence on the displayed pattern. In case of a pattern which uses every LED of the matrix a good compliance to the "Han Algorithm" can be achieved. The actual timing and of course the time needed for the sampling of the values is shown and explained graphically.

5.3 Output Signal Behaviour

Signals of great interest are the outputs of the amplifier stages. The small photocurrent is converted into an appropriate output voltage with a transimpedance of 10 mV/nA. These output signals coincide with the inputs of the six ADC channels of the microcontroller. In Figure 3.26 to Figure 3.30 the output signals are measured in case of static input signals or at least with a very slow switching clock. In contrast if one of the algorithms drives the matrix the switching is unequally faster. The control signals of the signals are discussed above but in this case the transient behaviours of the amplifiers output signals are the issue at stake.

Figure 5.7 shows the output signal of amplifier four (CH1). Additionally the control signals of the even (CH2) and odd (CH3) rows are part of the figure. None of the matrix LEDs are covered by an object; only a small amount of ambient light impinges on its surface. If the control signal of the even switch is high all even LEDs are emitting light. This also connects the input of the monitored transimpedance amplifier to the supply voltage and forces the output of the amplifier in its rail. After the even rows are disconnected the parasitic capacitance of the monitored LED has to be discharged. The odd rows are now connected to supply and the LED at the amplifier's input detects the light. Therefore the amplifier output signal increases up to approximately 300 mV (depending on the total amount of incident light). If all LEDs are disconnected from the supply voltage the amplifier measures only the ambient light (a few mV).

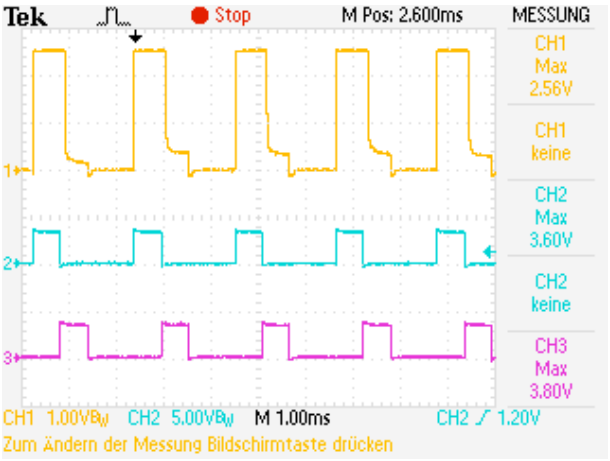


Figure 5.7: One amplifier stage output during an algorithm controlling the matrix

The following measurement pictured in Figure 5.8 shows the same signals but with an object covering some of the matrix LEDs. As its consequence the output signal increases to 600 mV if the according neighbors are active. Hence, the subsequent part (no LED emits light) of the output signal decreases because of the ambient light diminished by the covering object.

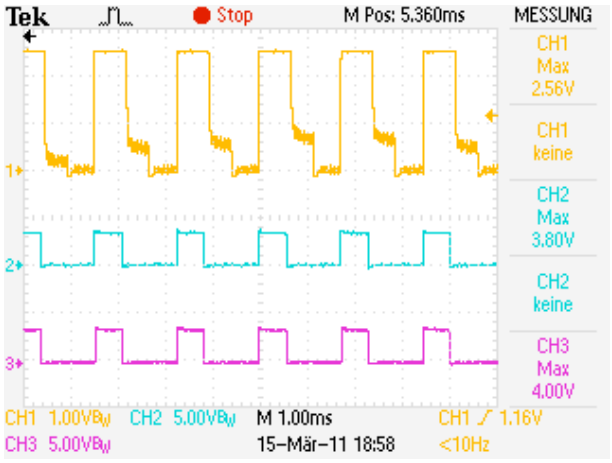


Figure 5.8: One amplifier stage output during an algorithm controlling the matrix with covered LEDs

Figure 5.9 shows the output signal (CH1) in more detail. The time axis and the voltage axis are zoomed to get a better view of the transient behaviour of the signal. The exponential discharge function of the LEDs' parasitic capacitance can be seen clearly. The additional curves are the input signals of the even (CH3) and odd (CH2) rows. Note that the output signal of the amplifier stage does not fit on the figure due to the setting of 200 mV per division. The highlighted voltage level of 360 mV is generated by the ambient light and the emitting neighbors. As the measured ambient light is very small the resulting voltage level (a few mV) in case of none emitting neighbors cannot be seen noticeably using this plotting scale.

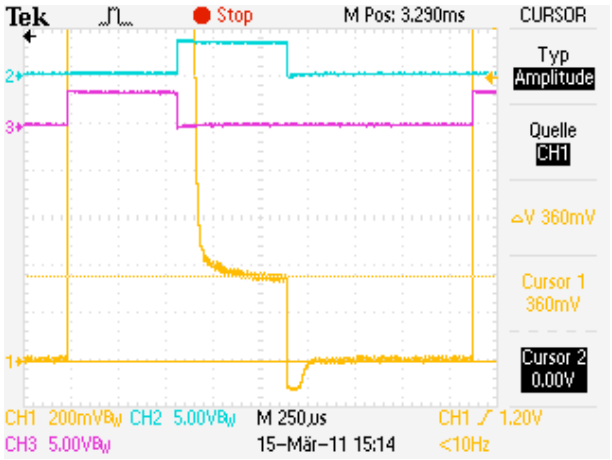


Figure 5.9: Detailed amplifier stage output signal

Apart from the output signals of the amplifiers also the voltage levels of the common cathodes of each column are interesting. As mentioned in prior chapters the on-resistance of the switches is not zero and therefore a voltage drop occurs. This voltage drop leads to a voltage over the LED. In Figure 5.10 the voltage level of one column (CH1) is measured according to the control signal of the switch (CH2). The measurement shows the signals while the "Han Algorithm" drives the matrix. In this case three LEDs are connected to the supply voltage if emitting neighbors are needed and the current flowing toward ground totals up to 60 mA. The total current leads to a voltage drop of round about 180mV at the common cathodes of the LEDs (highlighted in the figure). In the last third of the active column none of the LEDs emit light and only a small photocurrent flows through the switch. No measurable

voltage drop occurs at the common cathodes of the column. After the column is disconnected from ground the voltage drop increases to 800 mV. This is caused by the next step of the algorithm. The subsequent column is connected to ground and three of its LEDs emit light. Through the common anode connections at each row a leakage current flows into the disconnected columns, too. The voltage drop itself can be explained by the threshold voltage of the internal clamp diodes of the switch against ground which flow through the leakage current.

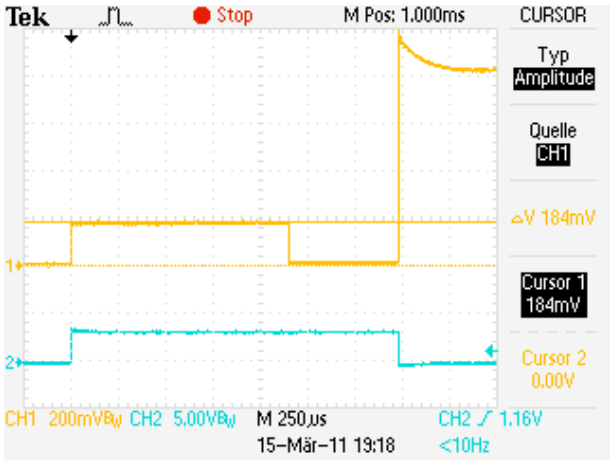


Figure 5.10: Voltage level at the common cathodes of one column

5.4 Algorithm Innovations

As one learned in the process of this thesis the newly developed algorithm is a novel approach to use an LED matrix as a light sensitive input device. In contrast to related works it is possible to show patterns on the LED matrix and to measure it simultaneously. The resulting brightness of the LED matrix is independent from the pattern size because of an accommodation of the duty cycle just in time of displaying the pattern. With this kind of algorithm it is possible to create an interactive button out of the LED matrix. The LED matrix displays a pattern and if a touch is recognized the pattern changes. Up to now there is no algorithm available which is able to handle this case of application. A lot of other possible fields of applications can be put into practise using this kind of algorithm. The algorithm provides the needed signals and calculations for an evaluation software to realize multifarious applications. For a wide range of ambient light the algorithm works fine due to its fast measuring sequences and calculations. Neither natural light sources nor artificial light sources are a limit for the algorithm. It is even conceivable to display animated patterns on the matrix during a simultaneous monitoring of the LED signals.

5.5 Weak Points

The present embodiment works well under typical light conditions and in the darkness but it is not the last word on the subject. Some weak points of the system are listed below and possible solutions are given to improve the performance of the embodiment.

Parasitic Capacitance:

If an LED of the matrix is used to emit light before it is used to detect light it is necessary to take into account that the parasitic capacitance of the LED is charged. This capacitance has to be discharged before the LED is able to detect light (Figure 5.9). The current to discharge the capacitance must flow through the feedback resistor of the transimpedance amplifier. The high value of the feedback resistor leads to a long discharging time. An additional switch to bypass the feedback path of the current-to-voltage converter with a smaller resistor would decrease the discharging time. Such an upgrade of the circuit allows the algorithm to switch faster between the emit- and detect-status of each LED.

External Oscillator:

Actually the microcontroller uses its internal RC-oscillator to generate the system clock and drive the 96 MHz PLL for the USB clock. Under typical conditions the RC-oscillator works with an accuracy of ± 0.25 %. This is sufficient for the present applications including the USB communication. In more critical conditions the accuracy of the internal oscillator can decrease to +1 % and -1.25 %. To avoid problems with the clock it would be better to use an external oscillator to guarantee an accuracy of ± 0.25 %.

Very Bright Ambient Light Conditions and leakage current:

Owing to the common ground connection of each column and the common supply connection of each row the LEDs of the matrix are interconnected at their anodes and cathodes. This results in many possible paths for leakage currents within the matrix. In typical conditions with moderate light levels this is not a problem for the function of the embodiment. However, under very bright light conditions the leakage currents within the matrix can disturb the operation of the system. If a very bright light strikes the surface of the matrix every LED produces a significant photocurrent apart from its connection to ground. If a switch connects the LED cathodes to ground a photocurrent flows towards the amplifier and evokes a positive output voltage at the amplifier stage. An LED without a ground connection also produces a photocurrent but the photocurrent flows in the opposite direction sourced by the amplifier stage connected to its anode. This leads to a current which flows against the monitored photocurrent and influences the measurement. The worst case takes place if the LED values are measured without emitting neighbor. In this case every LED is irradiated by a similar amount of light and the opposite current leads to massive irritations. Experiments have shown that even negative output signals are produced if only the non-active LEDs irradiated. To avoid these problems an additional switch at each row is recommended between LEDs an amplifier stage. These switches can interrupt the responsible current path and thereby inhibit an adulteration of the measurement.

6 Conclusion and Outlook

The thesis has shown how an ordinary LED can be used as a light-sensitive sensor. Not only single LEDs, also the use of LED matrices have been presented and a new kind of algorithm was invented to fully utilise the LED matrix ability to detect and emit light simultaneously. The knowledge about the light-sensitive ability of LEDs is based on a wide literature survey summarized in Chapter 2. This chapter contains the fundamental basics of LEDs and compares them with photodiodes which lead to the addressed application as sensor. Additionally it deals with the benefits and negative effects of using a whole matrix of LEDs in this field of application. The concrete implementation starts in Chapter 3 with a detailed explanation of the invented hardware platform. The chapter shows that it is of vital importance to set up a sophisticated embodiment to achieve usable output signals and discusses critical parts in detail. Moreover, the hardware constrained the substructure of Chapter 4: the software implementation. Apart from the mere technical implementation the fourth chapter shows the logical structure and observations behind the algorithms. Also the newly invented algorithm as a related algorithm is explained in detail. Chapter 5 completes the thesis by comparing the outcome results (underlined with a lot of measurements) with the postulated coherences of the theory and the expected performance of the embodiment. Last but not least this chapter does not keep weak points of the embodiment secret and shows possible solutions for improving its performance. This closing conclusion, together with a short overview of future challenges finalizes the thesis. All this is complemented with the according engineering data in Appendix A and an enclosed CD-ROM with the actual software source code.

The thesis ends at this point but the project will continue. As mentioned there is still a high potential to improve the system to make it still more independent from ambient conditions. The hardware is designed to act as a demoboard to present the systems to customers. Therefore austriamicrosystems owned parts should replace all parts of external companies to show the customer: we are able to face all of the challenges made by the system! Concrete use cases will be defined and put into practise by an extension or adaption of the present software to take account of customer requests. Furthermore, based on the present thesis, new master theses stuck in the wings with special attention to introduce the technology under the influence of covers over the LED matrix and the newly invented algorithm stands on the verge of being patented.

7 Bibliography

[AMS1] AUSTRIAMICROSYSTEMS: *AS1744, AS1745 High-Speed, Low-Voltage, Dual, Single-Supply, 4Ω, SPDT Analog Switch Datasheet.* http://www.austriamicrosystems.com/eng/content/download/1300/7259/583

[AMS2] AUSTRIAMICROSYSTEMS: *AS1363 500mA, Low-Dropout Linear Voltage Regulator Datasheet.* http://www.austriamicrosystems.com/eng/content/download/12507/219217/12422

[AMS3] AUSTRIAMICROSYSTEMS: *AS1358/AS1359 150mA/300mA, Ultra-Low-Noise, High-PSRR Low Dropout Regulators Datasheet.* http://www.austriamicrosystems.com/eng/content/download/12507/219217/12422

[BURRB1] BURR-BROWN: *Photodiode Monitoring with OP Amps.* Application Bulletin, Tucson, 1995 http://focus.ti.com/lit/an/sboa035/sboa035.pdf

[CARTER1] CARTER Bruce, MANCINI Ron: *Op Amps for Everyone – Third Edition*. Newnes, Burlington, 2009

[DIETZ1] DIETZ Paul H. et al.: *LED With Controlled Capacitive Discharge for Photo Sensing.* U.S. Patent 6,870,148 B2, Mar. 22, 2005

[GRAEME1] GRAEME Jerald G.: *Photodiode Amplifiers – Op Amp Solutions*. McGraw-Hill, New York, 1996

[HAN1] HAN Jefferson Y.: *Multi-Touch Sensing Light Emitting Diode Display and Method for Using the Same*. U.S. Patent 7,598,949 B2, Oct. 6, 2009.

[KASAP1] KASAP Safa O.: *Optoelectronic Devices and Photonic: Principles and Practices*. Prentice Hall, New Jersey, 2001

[KINGB1] KINGBRIGHT: *SMD Chip LED Lamp APHHS1005SURCK Datasheet*. http://www.us.kingbright.com/images/catalog/SPEC/APHHS1005SURCK.pdf 2009

[MAX1] MAXIM: *MAX828/MAX829 Switched-Capacitor Voltage Inverters Datasheet.* http://pdfserv.maxim-ic.com/en/ds/MAX828-MAX829.pdf

[MICRO1] MICROCHIP: *PIC24FJ64GB004 Family Data Sheet.* http://ww1.microchip.com/downloads/en/DeviceDoc/39940d.pdf

[MICRO2] MICROCHIP: *PIC24FJ64GB004 Family Referenz Manual, Sect. 17 10-Bit A/D Converter.* http://ww1.microchip.com/downloads/en/DeviceDoc/39705b.pdf

[MIMS1] MIMS Forrest M. III: *LED Circuits & Projects*. H. W. Sams, Pennsylvania, 1973

[NATIONAL1] NATIONAL SEMICONDUCTORS: LM137/*LM337 3-Terminal Adjustable Regulator Datasheet.*
http://www.national.com/ds/LM/LM137.pdf

[SCHUBERT1] SCHUBERT E. Fred: *Light Emitting Diodes – Second Edition*. Cambridge University Press, New York, 2006

[SINGH1] SINGH Jasprit: *Semiconductor Devices – Basic Principles*. John Wiley & Sons, Inc., New York, 2001

[STINY1] STINY Leonhard: *Handbuch aktiver elektronischer Bauelemente – Aufbau, Strukturen, Wirkungsweise, Eigenschaften und praktischer Einsatz diskreter und integrierter Halbleiter-Bauteile*. Franzis Verlag GmbH, Poing, 2009

[SZE1] SZE S. M., KWOK K. Ng: *Physics of Semiconductor Devices – Third Edition*. John Wiley& Sons, Inc., Hoboken - New Jersey, 2007

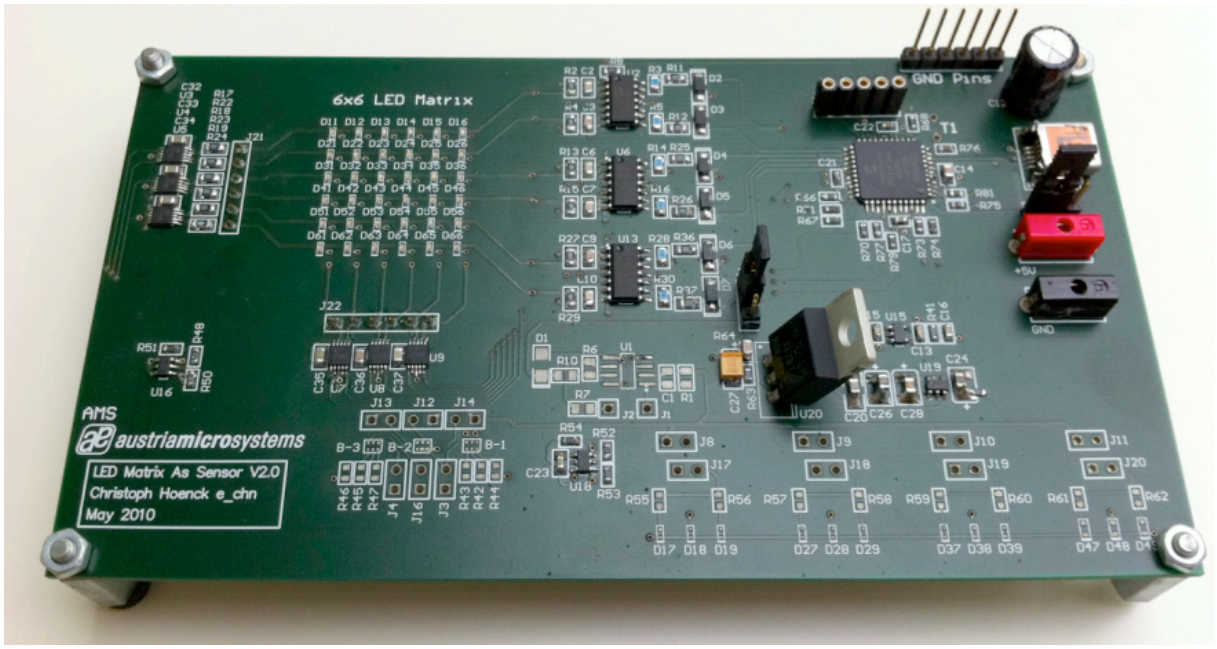
[TEX1] WANG Tony, ERHMAN Barry: Compensate Transimpedance Amplifiers Intuitively. Texas Instruments – Application Report, 2005
http://focus.ti.com/lit/an/sboa055a/sboa055a.pdf

[THEISS1] THEISS Erik: *Beleuchtungstechnik: neue Technologien der Innen- und Außenbeleuchtung.* Oldenbourg Industrieverlag, München, 2000

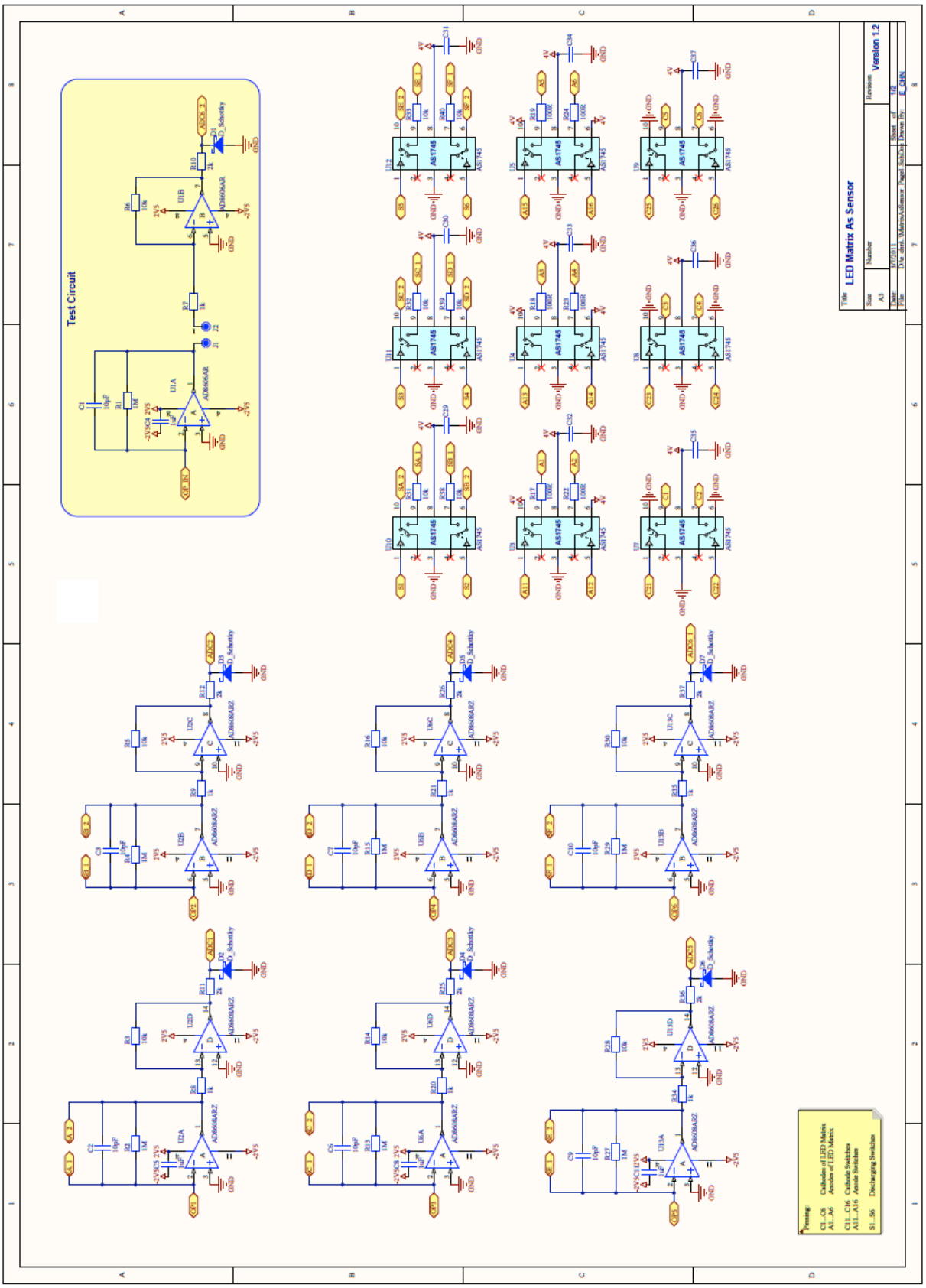
8 Appendix A

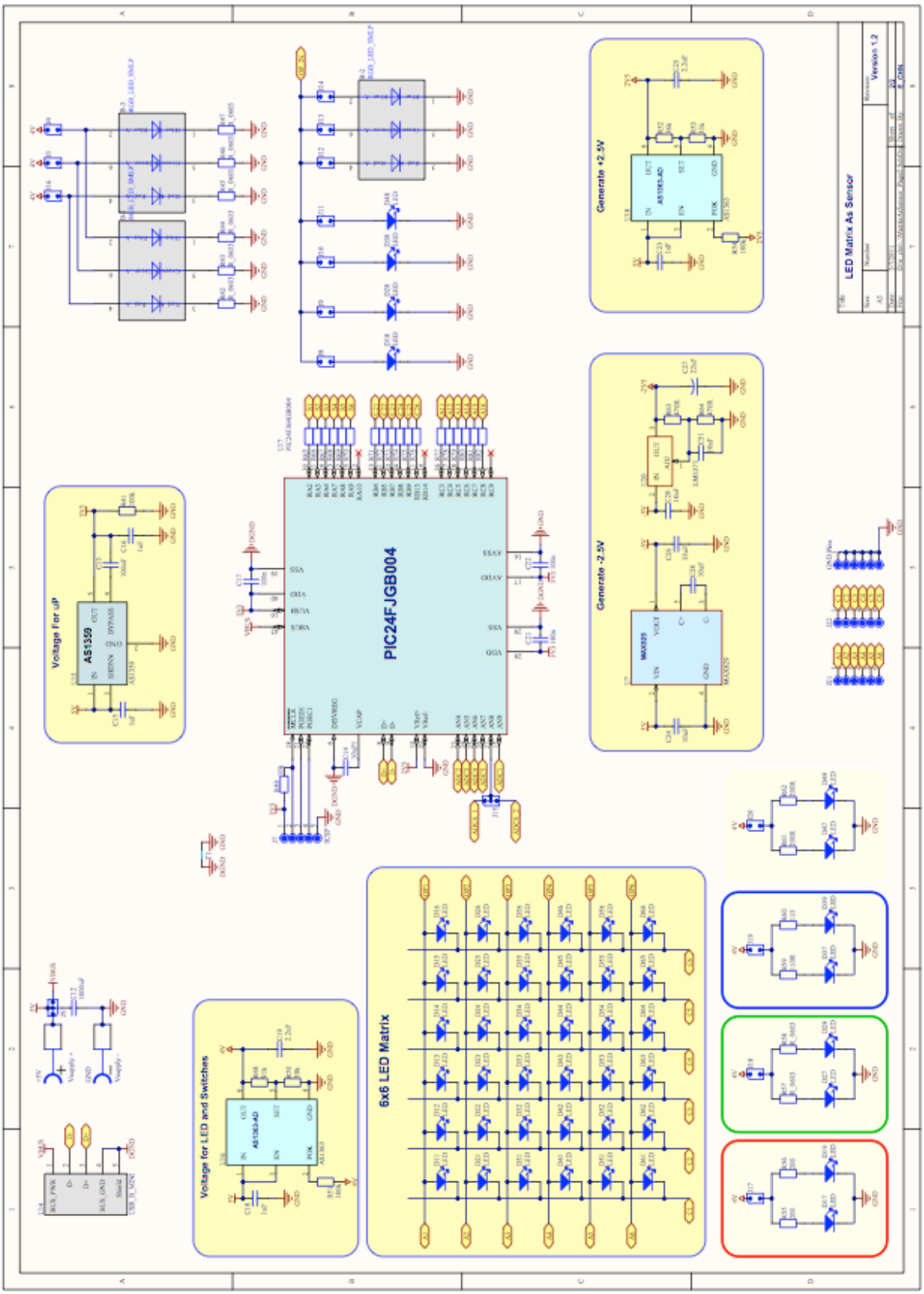
The appendix includes all relevant engineering data of the present hardware embodiment and a picture of the finished hardware as well the schematic and the layout of the designed four layers PCB. Not every part on the PCB is discussed in detail in the present thesis. This includes some test circuits and additional features for subsequent applications. Every part on the PCB which is not mentioned in the thesis does not contribute to the main characteristic of the described embodiment and is therefore not part of the master thesis.

8.1 Finished Hardware



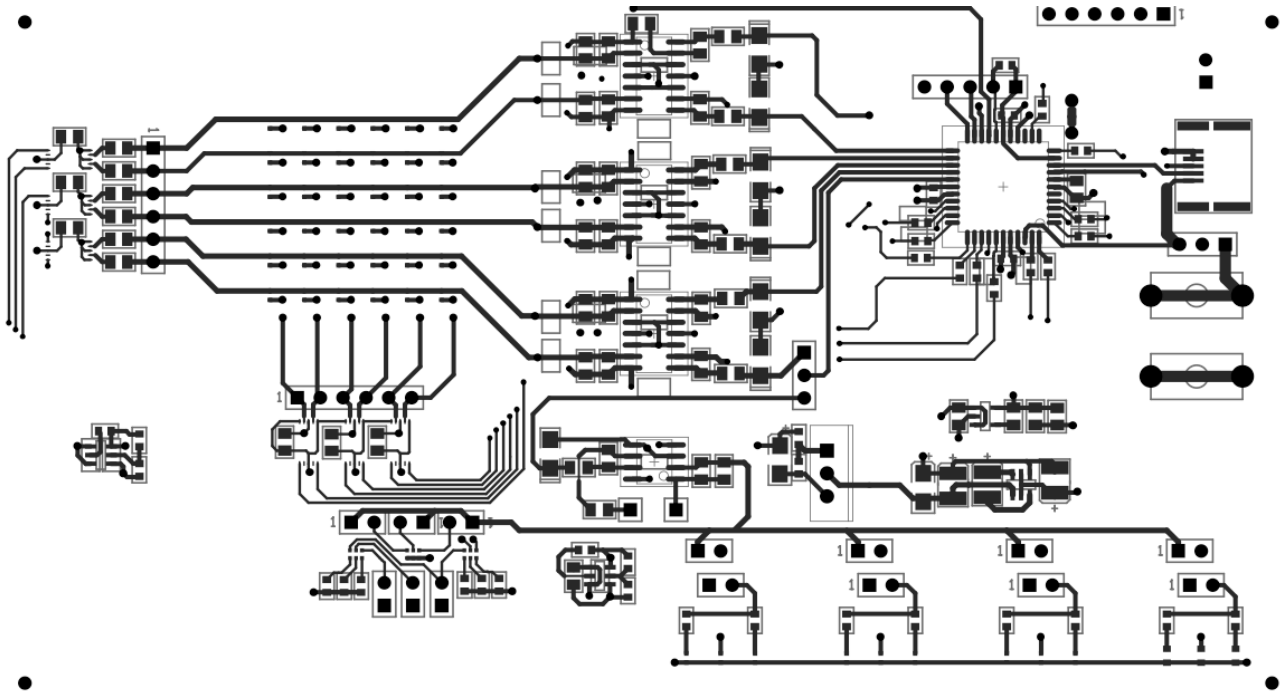
8.2 Schematic



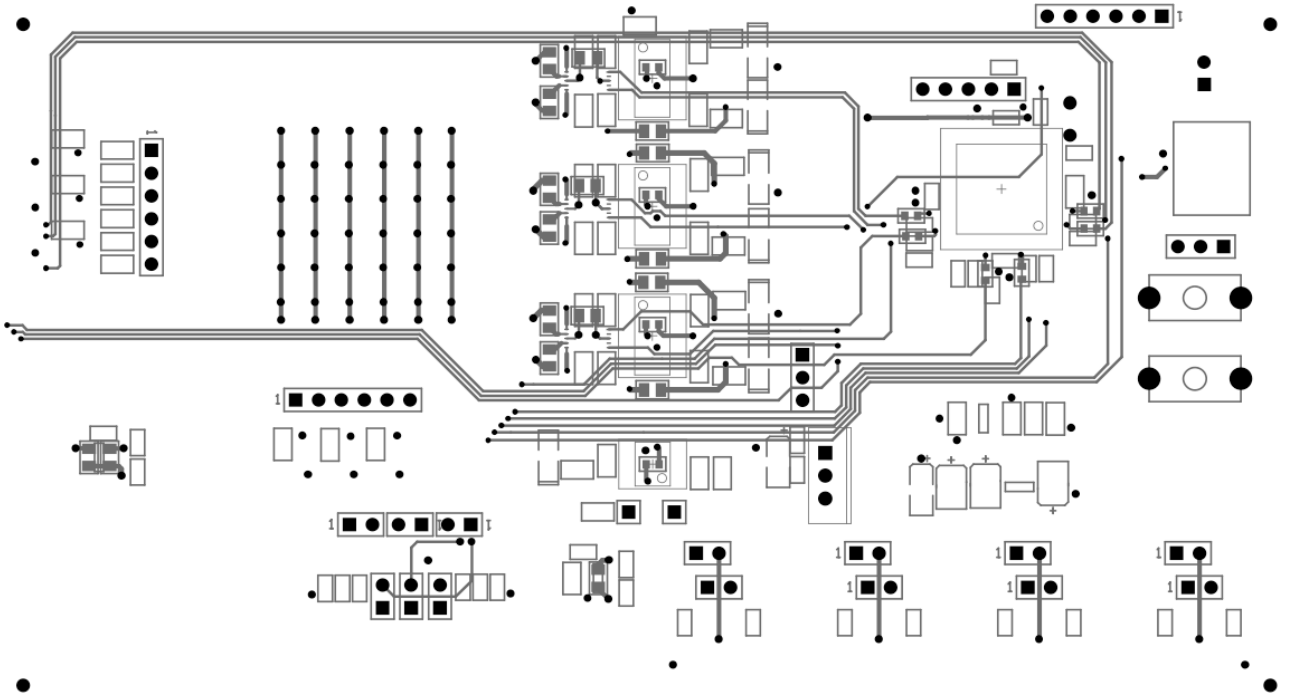

LED Matrix As Sensor
Generate +2.5V
Generate -2.5V
Voltage For uP
PIC24FJGB004
6x6 LED Matrix
Voltage for LED and Switches

8.3 Layout

Top Layer:



Bottom Layer:



9 Appendix B

Appendix B contains the kindly permission from Cambridge University Press to reprint some of their figures published in [SCHUBERT1] and an email response from Microchip including a link to their legal page which also contains the kind permission to reprint their figures published in [MICRO1] and [MICRO2].

9.1 Cambridge University Press

CAMBRIDGE
UNIVERSITY PRESS

Christoph Hönck
Heinrichstrasse 8/DG4
8010 Graz
Austria

The Edinburgh Building
Shaftesbury Road
Cambridge CB2 8RU, UK

www.cambridge.org

Telephone +44 (0)1223 312393
Fax +44 (0)1223 315052
Email information@cambridge.org

February 2, 2011

Dear Christoph Hönck

Figure 4.2, page 62 and Figure 4.5, page 65-67, from E. Fred Schubert, Light-Emitting Diodes, (2006).

Thank you for your recent permission request, to include the above extract/s in:
your forthcoming MSc thesis provisionally entitled *Design of a Touch Screen Using Only a LED Matrix*, for non-commercial publication.

Non-exclusive permission is granted free of charge for this specific use on the understanding that you have checked that we do not acknowledge another source for this material.

Please ensure full acknowledgement (author, title, publication date, and Cambridge University Press).

Yours sincerely

C. Taylor

Claire Taylor
Publishing Assistant
email ctaylor@cambridge.org

9.2 Microchip

Hello Christoph:

Thank you for your email. Here is a link to our legal page concerning your request. http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=99

Under "Copyright Usage Guidelines", note the section: "Educational and Non-Profit Use of Copyrighted Material"

If you have any questions, please contact our legal department at legal.department@microchip.com.

Regards,

Marc McComb
Academic Program Sales Engineer
2355 West Chandler Blvd., Mailstop 9-B
Chandler, AZ 85224-6199
office: 480-792-4391
mobile: 480-478-5676

Need more information on Microchip's Academic Program or would like to become a Partner? Visit ***www.microchip.com/academic*** for more information.