Swakeup

PROJECT REPORT

within the lecture of
Programming Embedded Systems

at Uppsala University
in the Departement of Information Technology

Elmar van Rijnswou (Elmar.Vanrijnswou.9818@student.uu.se)
and Maximilian Stiefel (Maximilian.Stiefel.8233@student.uu.se)

Deadline: 2017-05-31 24:00

Processing Period:	January 2017 - May 2017
Supervisor:	Philipp Rümmer (philipp.ruemmer@it.uu.se)

Contents

List of Figures

List of Tables

1 Introduction

It is a well-known fact, that Sweden is a country with big variations of the daily hours of sun throughout the year. Far up north the sun does not set anymore in January. Even in Stockholm in January, the most dark month of the year, the sun rises at 8:47 am and sets at 2:55 pm (cf.[1]). According to *Sveriges Radio* "many Swedes suffer from the winter blues or seasonal affective disorder" (cf. [2]). In a strong winter every source of light is a source of happiness. This is why a wakeup light, which is based on a strong light source (at least 10 W RGB LED), is able to give one the optimal start into a dark winter day with an artificial sunrise as bright as a real sun shining through the window.

This report describes the Swakeup (from engl. "Swedish Wakeup Light"), a device communicating to the user not only through light. It does not simply wake one up, but also gives one information about social media, latest mails, calendar and weather. The user interface consits besides of a high-power LED of an OLED screen. Swakeup is also part of the IoT as it has the ability to communicate via IEEE 802.11. This of course enables a lot of possibilities e.g. connecting your phone to the wakeup light. A lot of effort has been put into the designing maxim, that everything should be as small as possible. The whole electronics fit on an base area of 5 cm x 4 cm. So the Swakeup fits smoothly on the bedside table. And honestly: What is the last thing people are doing before they go to sleep? Right! They look on your phone. That is why Swakeup comes with a USB charger for your e.g. phone as well. Another design maxim of this product is cheapness. Everybody should be able to buy one. As all engineering work is available online, it gives people (with the corresponding knowledge) the opportunity to build a wakeup light themselves, look up what this device is doing with their personal data or even contribute to the product.

2 Background and Analysis

3 Design

3.1 System Architecture and Required Functionality

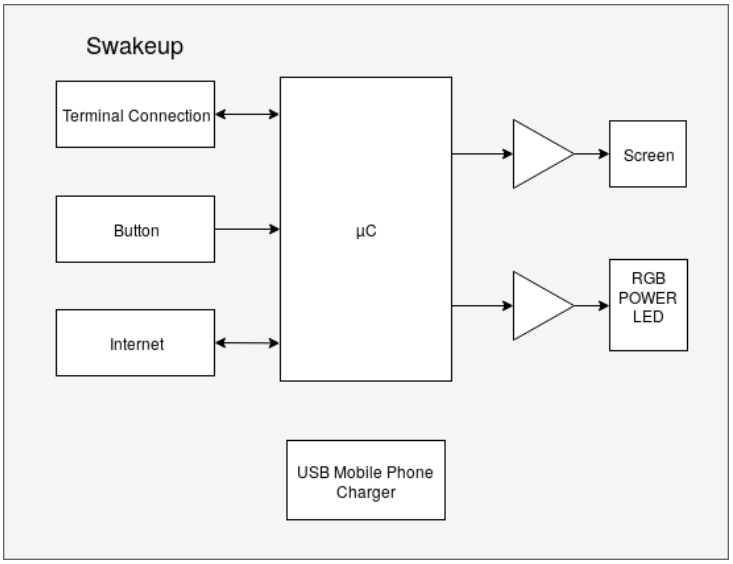


Figure 3.1: Simplified blockdiagram of Swakeup.

In fig. 3.1 a simplified system architecture from an abstract (e.g. customer) point of view is displayed. The heart of the system is a microcontrolller. This microcontroller has to procide the necessary peripherals and enough RAM and flash for the necessary funtionality, which has to be implemented in software.

A terminal connection is crucial to transport data to a computer. One needs this to do debugging and development. The idea is, that a USB cbale can simply be plugged in and then the developer can start working without additional complications.

A button is not only usefull for the programmer to interact with the code to set the system to a certain state, but it is also absolutely needed for deployment of the systems in the real world. Obviously, it should be extra convenient and simple for a sleepy user to snooze (e.g. short button pressing) or to turn off the alarm (e.g. long button pressing).

The idea of a IoT device is of course, that it is connected to the internet, that is why a component has to be included into the design to, which enables the system to receive a IP address. As the final product will be standing most likely on a bedside table of the customer a wired internet connection (e.g. Ethernet) does not make any sense. Instead a wireless connection is more convenient (e.g. IEEE 802.11).

USB charging circuitry i.e. a USB dedicated charging port has to be implemented to charge phones with a current of at leas 1.5 A.

As screen provides information to the user, which goes beyond the current date and time (e.g. weather, social media and email). It is reasonable, that this screen needs to be powered somehow. Also the software developer or the user should have controll over the screen brightness. This means, that some

time has to be invested in the contruction of a powerful software-steerable screen driver.

To light up a multi-color LED with a sufficient brightness to emulate a sunrise circuitry has to be implemented to drive a high current. This current has to be steerable in magnitude by the microcontroller to mix colors together according to the RGB color model.

3.2 Design Choices

Block	**Component(s)**	**Cost**	**Justification of the Choice**
Terminal Connection	SILICON LABS CP2102	24 SEK	Simple chip. Drivers for different operating systems are usually out-of-the-box available (plug and play). Available in QFN package.
Internet	ESP8266	25 SEK	ESP-E12 comes as ready to solder SMD module. No antenna design required. Easy connection to microcontroller via UART. Sming framework makes it easy to program chip in C++.
USB Mobile Phone Charger	TS30012 and TPS2514	40 SEK	TS30012 is a relatively cheap chip to break down 20V to 5V and to provide up to 2A output current. USB designated charging ports need signaling for different proprietary/open standards. TPS2514 is a quite nice chip to provide this signaling for different brands.
RGB LED and Driver	Custom Step-Down Converter	20 SEK per Stage	Fast, powerful LED driver built up with discrete components can be tailored to the requirements of the device and is therefore cheaper and more performant. Necessary feedback circuitry is also hard to find as COTS.
Screen and Driver	Custom Opamp Amplifier	30 SEK	One channel of the LM324 (four channel operational amplifier) was unused a simple circuit made it a cheap and easy-to-use DAC screen driver.

4 Implementation

4.1 Hardware

4.2 Software

5 Testing

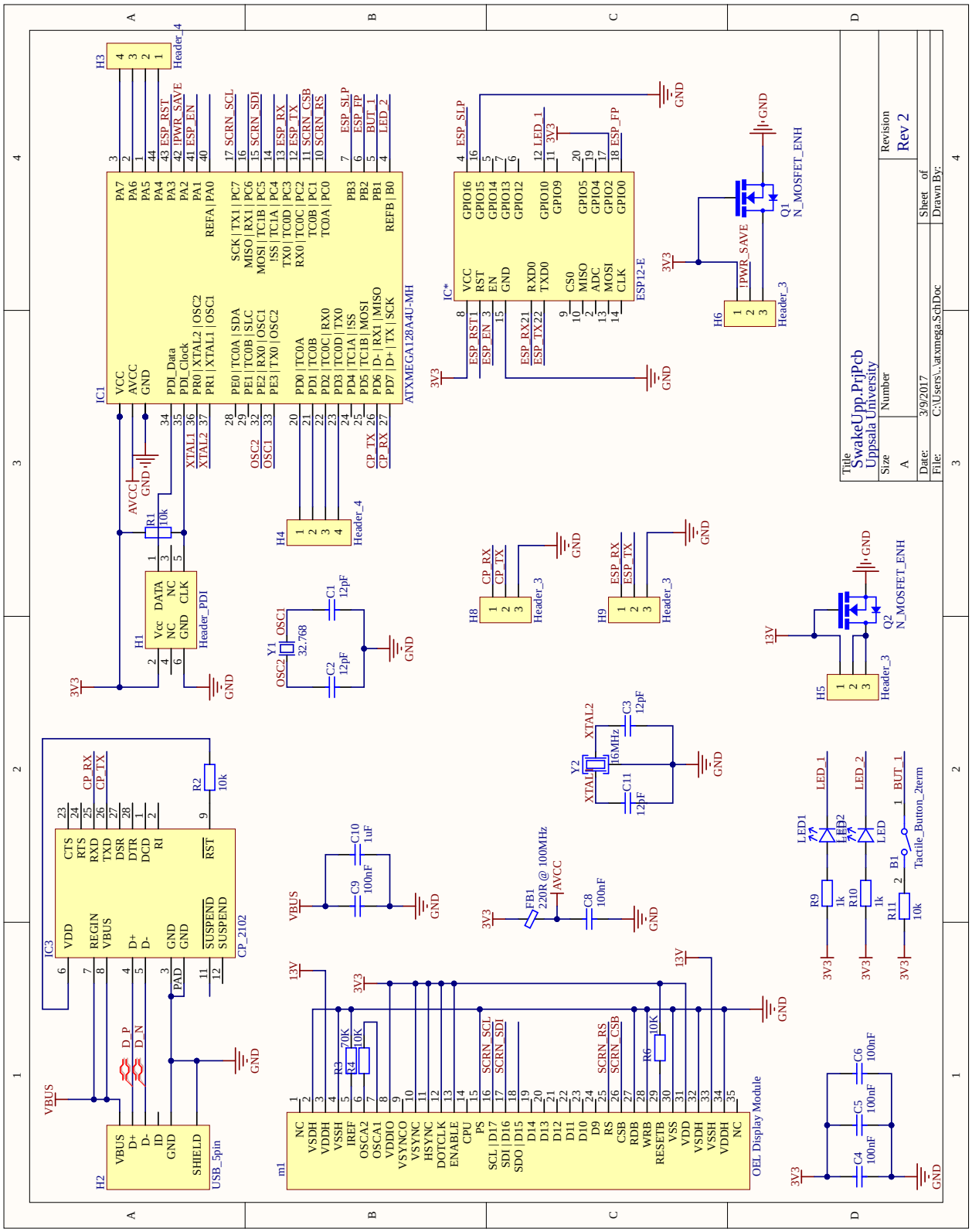
6 Conclusion

Bibliography

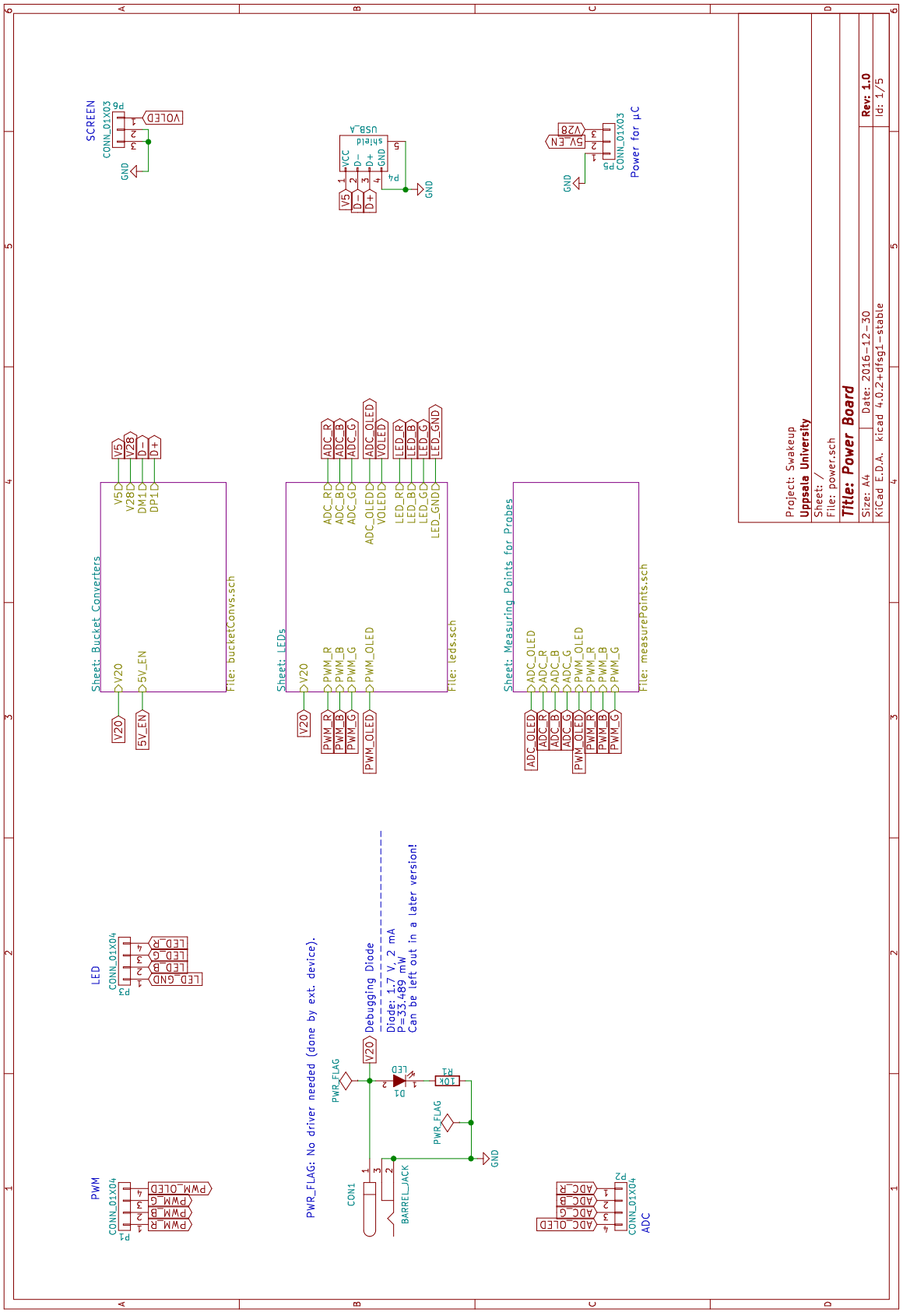
[1] *Visit Sweden*. (2017). Time and daylight in Sweden, [Online]. Available: https://visitsweden.com/time-and-daylight-hours/.

[2] *Sveriges Radio*. (2014). Are Swedes more suicidal than most? [Online]. Available: http://sverigesradio.se/sida/artikel.aspx?artikel=5924063.

A Schematics Logic Board

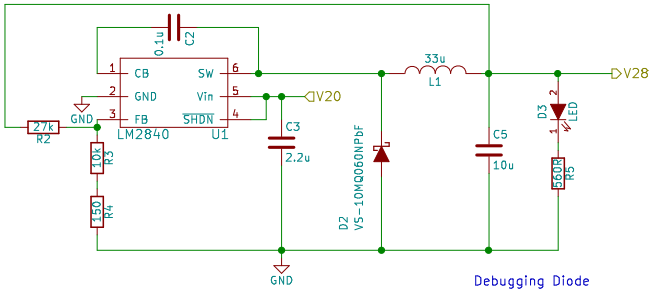


B Schematics Power Board



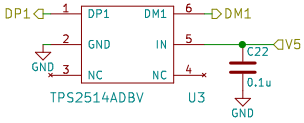
2.8 V for µC

Vout=0.765 V (1 + (27k/ 10.15k))=2.8 V
27k/10.15k= 2.66



Debugging Diode

Diode: 1.7 V, 2 mA
P=2.161 mW
Can be left out in a later version!

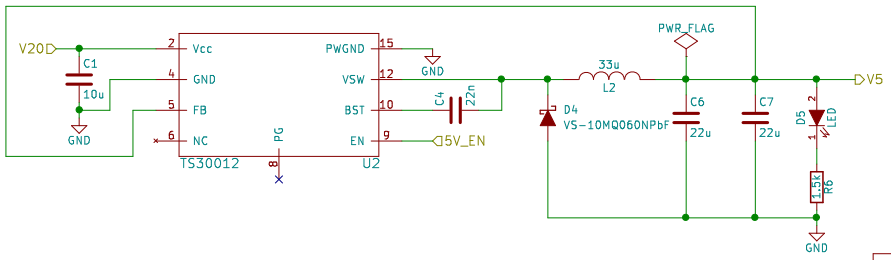


USB Dedicated Charging Port Control.

Simple SOT-23-6 IC for detecting propriatary and open standards used by a device and providing the corresponding electrical signature at the data lines (voltage or impedance).

5V for Mobile Phone

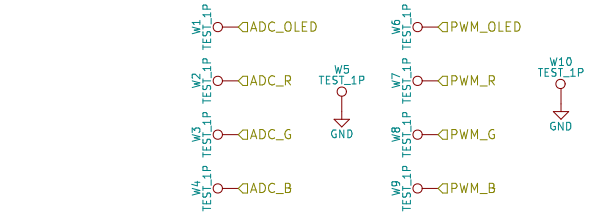
Vout,max=5 V
Aout,max = 2A



Debugging Diode

Diode: 1.7 V, 2 mA
P=7.26 mW
Can be left out in a later version!

Uppsala University		
Sheet: /Bucket Converters/ File: bucketConvs.sch		
Title: Bucket Converters for 5 V and 2.8 V		
Size: A4	Date: 2016-12-30	**Rev: 1.0**
KiCad E.D.A. kicad 4.0.2+dfsg1-stable		Id: 2/5



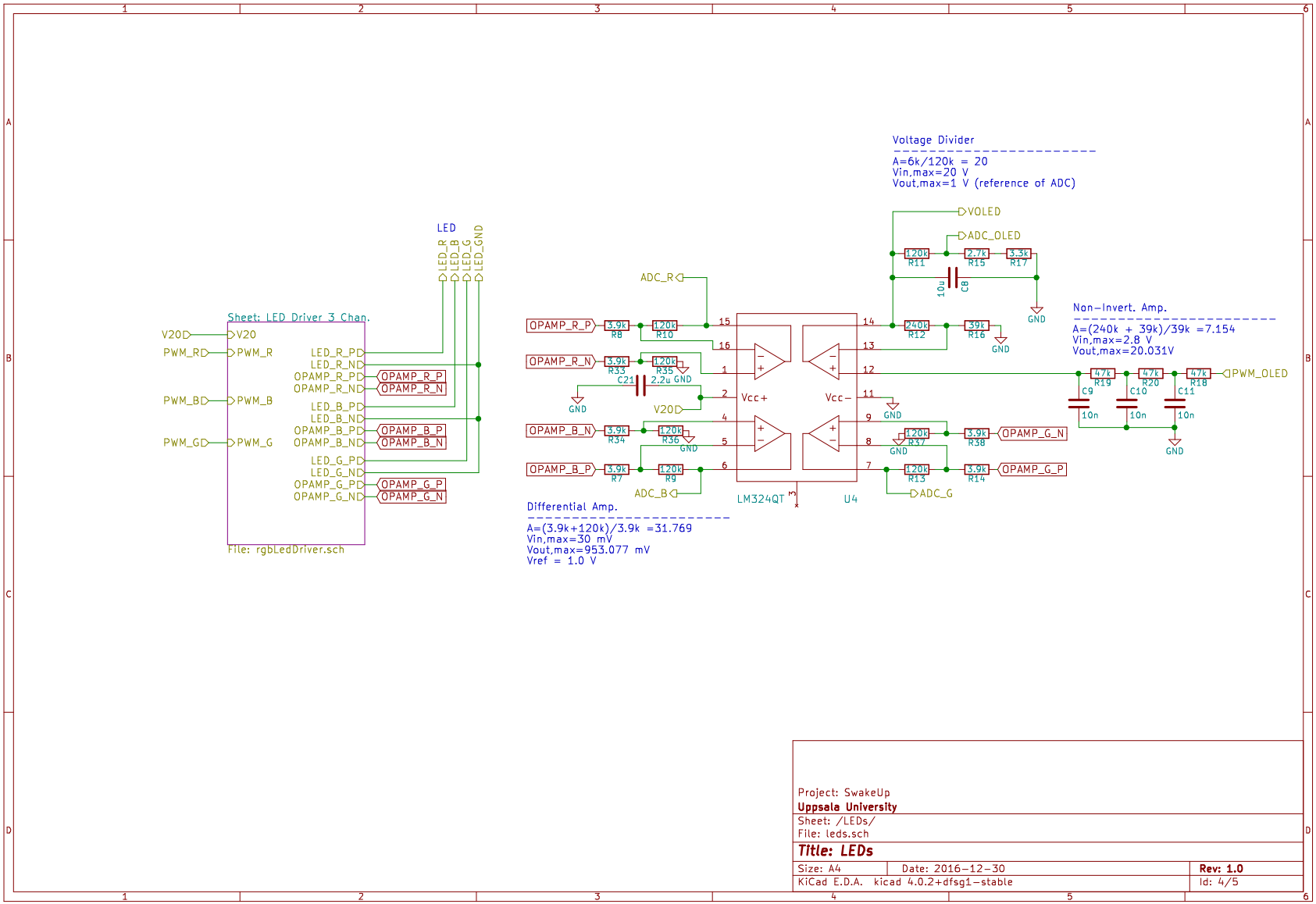
Test Points for the FBs.
Can be left out in a later version!

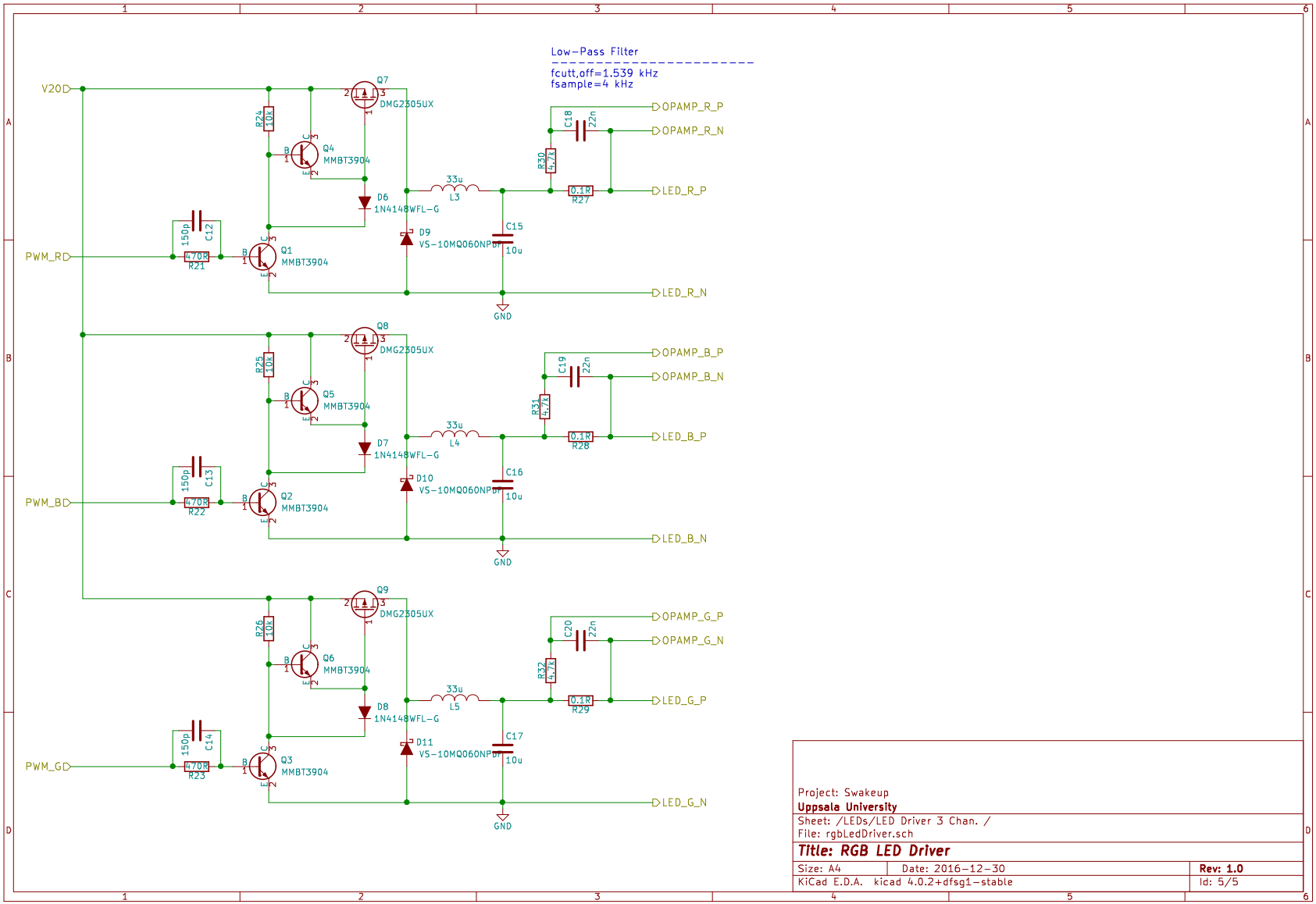
Test Points for the PWM signals.
Can be left out in a later version!

Project: Swakeup
Uppsala University
Sheet: /Measuring Points for Probes/
File: measurePoints.sch

Title: Test Points

Size: A4	Date: 2016-12-02	**Rev: 1.0**
KiCad E.D.A. kicad 4.0.2+dfsg1-stable		Id: 3/5


LED
LED_R
LED_B
LED_G
LED_GND
Sheet: LED Driver 3 Chan.
V20
PWM_R
PWM_B
PWM_G
LED_R_P
LED_R_N
OPAMP_R_P
OPAMP_R_N
LED_B_P
LED_B_N
OPAMP_B_P
OPAMP_B_N
LED_G_P
LED_G_N
OPAMP_G_P
OPAMP_G_N
File: rgbLedDriver.sch
Voltage Divider
A=6k/120k = 20
Vin,max=20 V
Vout,max=1 V (reference of ADC)
VOLED
ADC_OLED
ADC_R
ADC_B
ADC_G
120k R11
2.7k R15
3.3k R17
10u C8
GND
240k R12
39k R16
Non-Invert. Amp.
A=(240k + 39k)/39k =7.154
Vin,max=2.8 V
Vout,max=20.031V
47k R19
47k R20
47k R18
PWM_OLED
C9 10n
C10 10n
C11 10n
OPAMP_R_P
OPAMP_R_N
OPAMP_B_N
OPAMP_B_P
3.9k R8
120k R10
3.9k R33
120k R35
C21 2.2u
3.9k R34
120k R36
3.9k R7
120k R9
Vcc+
Vcc-
V20
LM324QT
U4
120k R37
3.9k R38
120k R13
3.9k R14
OPAMP_G_N
OPAMP_G_P
Differential Amp.
A=(3.9k+120k)/3.9k =31.769
Vin,max=30 mV
Vout,max=953.077 mV
Vref = 1.0 V
Project: SwakeUp
Uppsala University
Sheet: /LEDs/
File: leds.sch
Title: LEDs
Size: A4
Date: 2016-12-30
Rev: 1.0
KiCad E.D.A. kicad 4.0.2+dfsg1-stable
Id: 4/5



Project: Swakeup
Uppsala University
Sheet: /LEDs/LED Driver 3 Chan. /
File: rgbLedDriver.sch

Title: RGB LED Driver

Size: A4	Date: 2016-12-30	**Rev: 1.0**
KiCad E.D.A. kicad 4.0.2+dfsg1-stable		Id: 5/5

C USART interrupt generation

```
#define USARTRXCISR(NAME, PORT, USART_ID, REC_FC)            \
ISR(NAME##_RXC_vect) {                                  \
        uint8_t read = PORT.DATA;                        \
        if (writeInBuf(read, &PORT)) {                   \
        REC_FC(read);                                       \
        uint8_t i = 0;                                  \
        for (; i < UART_MAX_DELIMITERS; i++) {      \
                if (delimiters[USART_ID][i].delimiter != 0) {  \
                        delimiters[USART_ID][i].length++;          \
                        if (read == delimiters[USART_ID][i].delimiter) {   \
                                delimiters[USART_ID][i].port = &PORT;       \
                                event_fire(&EVENT_UART_DELIMITER,        \
                                SYSTEM_ADDRESS_CAST (&delimiters[USART_ID][i])); \
                                }   \
                        }   \
                }   \
        } else {/*buffer full */     \
                CP_PORT.CTRLA &= ~(USART_RXCINTLVL_LO_gc);  \
        }   \
}

#define USARTDREISR(NAME, PORT, USART_ID)\
ISR(NAME##_DRE_vect) {              \
        uint8_t size = uartStatus[USART_ID].outBuffer_size;    \
        if (size > 0) { \
                if (softlock(USART_ID)) {\
                        uint8_t tail = uartStatus[USART_ID].outBuffer_tail;\
                        PORT.DATA = outBuffer[USART_ID][tail];  \
                        uartStatus[USART_ID].outBuffer_size--; \
                        tail++; \
                        if (tail >= UART_MAX_OUT_BUFFER) tail = 0;\
                        uartStatus[USART_ID].outBuffer_tail = tail;\
                        unlock(USART_ID);  \
                }\
        } else {\
                sending[USART_ID] = 0;\
                PORT.CTRLA &= ~(USART_DREINTLVL0_bm);\
        }\
}
```