MRS MODULE - UVDAR REVISION 1

David Zaitlik

Multi-Robot Systems Group, FEE CTU in Prague

February 10, 2021

Contents

List of Figures

List of Tables

1 Introduction

UVDAR is a system for mutual UAV localization using Ultra-Violet LED blinkers.

This PCB is an MRS Module providing UVDAR functionality to drones equipped by an MRS Distribution Board with a slot for a large MRS Module.

This module communicates using UART. Both communication and power are provided from the used Distribution Board.

The Board is designed to work with high-power UVLEDs ProLight Opto PM2B-1LLE, however any other similar UVLED can be used.

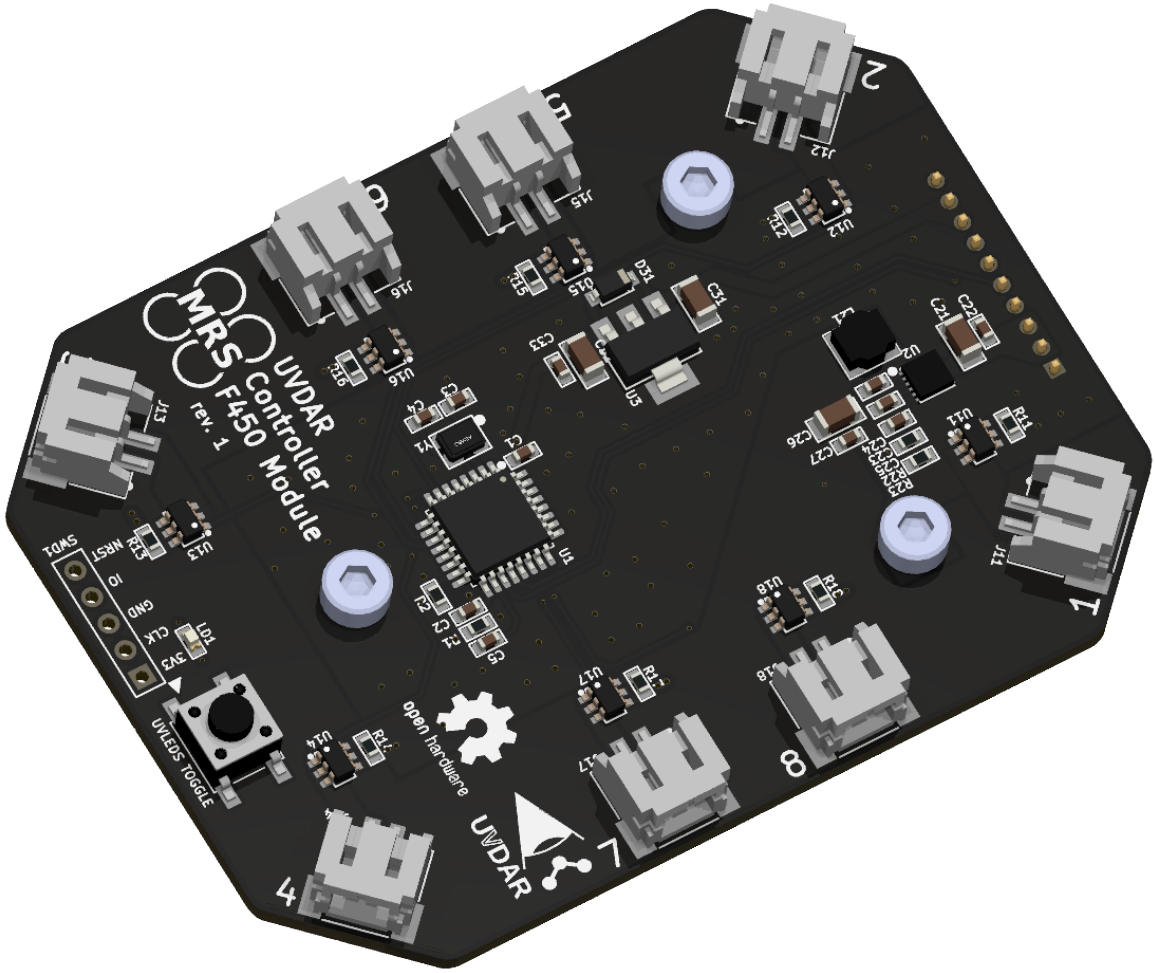


Figure 1: MRS MODULE - UVDAR: Thumbnail

1.1 Default Configuration

The default configuration used by the MRS Group drives two UVLEDs ProLight Opto PM2B-1LLE per channel. These UVLEDs are soldered to a PCB which acts as a heatsink and mounting point to the end of the drone arm. The UVLEDs are at a right angle, which can be seen in figure 2.

The voltage for UVLEDs is set to 8V, but this can be easily changed by adjusting the value of R_{22} resistor according to the following formula:
$R_{22} = 22.1 \cdot (V_{OUT} - 1)\,[k\Omega]$
The UVLEDs are driven using constant-current controllers that are set to approximately to 85mA. This value can be modified by changing the value of resistors $R_{11} - R_{18}$ according to the BCR421UW6 driver datasheet.

Channels 1-4 are used to drive the UVLEDs on the arms and channels 5-8 are used to drive the beacon UVLEDs on the top of the drone.

Figure 2: MRS MODULE - UVDAR: Photo of an F450 drone with UVDAR

2 Dimensions

The Board is 80mm long and 60mm wide with three M3 mounting holes.

The mechanical drawing can be seen in Figure 3 below.

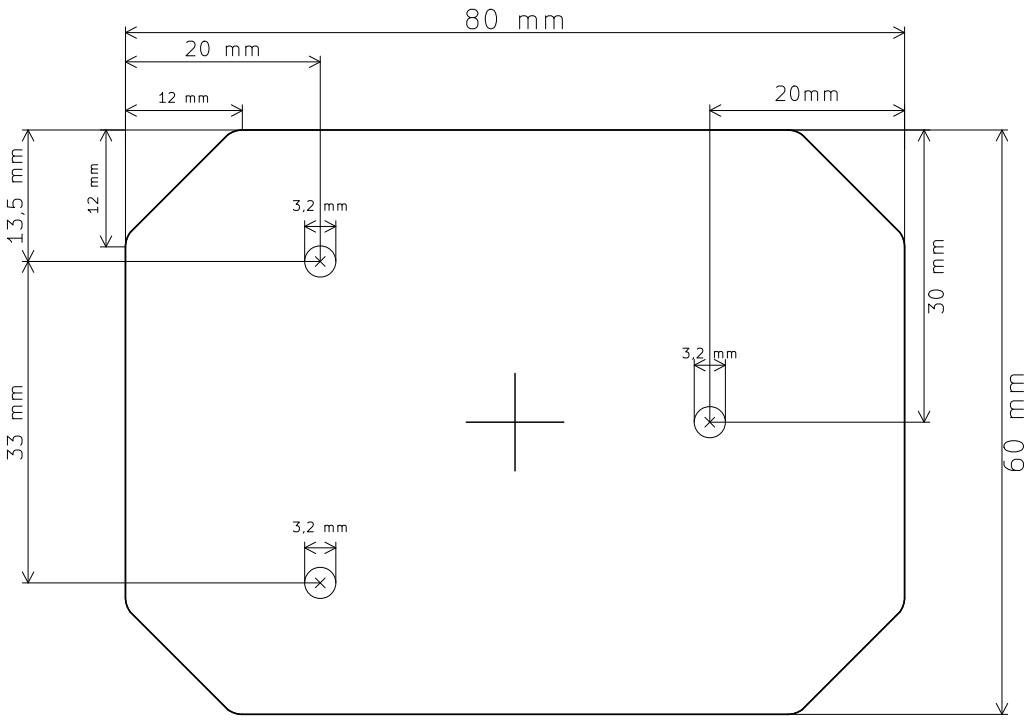


Figure 3: MRS MODULE - UVDAR: Dimensions

3 Connectors

The Board has 8 connectors numbered from 1 to 8 to connect UVLEDs. The pinout of these connectors is described in `UVLED`.
The board has a connector for connecting to a Distribution Board with a standard pinout as seen in `MRS MODULE`.
The last connector is the Serial Wire Debug (`SWD`) connector for programming the on-board STM32F042K6T. The connector placement can be seen in figures 4 and 5.

3.1 Connector Placement Drawing

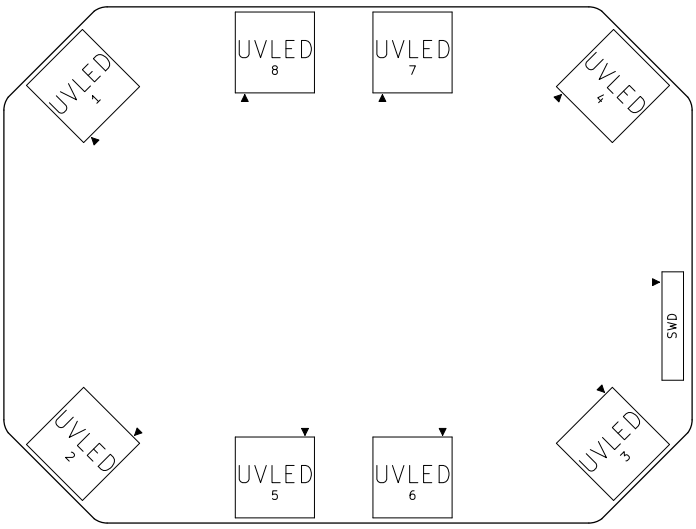


Figure 4: MRS MODULE - UVDAR: Connectors Placement, Top Layer

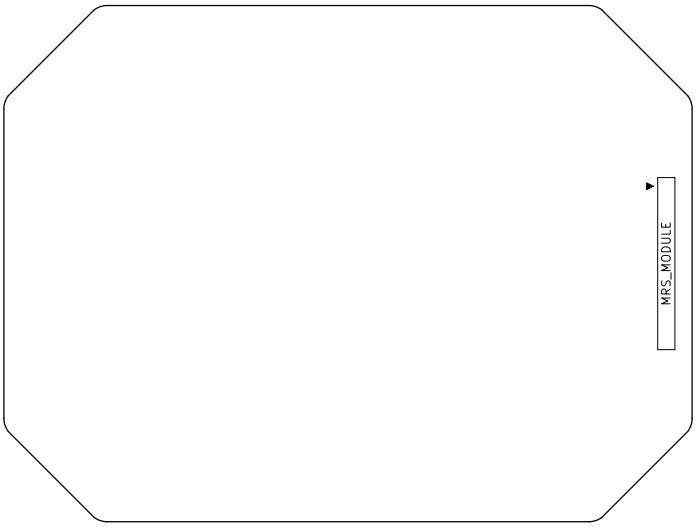


Figure 5: MRS MODULE - UVDAR: Connectors Placement, Bottom Layer

3.2 Connector Pinouts

SWD	
1	3V3
2	SWCLK
3	GND
4	SWDIO
5	NRST

UVLED	
1	LED ANODE
2	LED KATHODE

MRS MODULE	
1	GND
2	VBAT
3	VBAT
4	GND
5	UART_TX
6	UART_RX
7	GND
8	5V
9	5V
10	GND

4 MCU Firmware

The firmware receives data using UART (115200 baud); the data are in the `baca_protocol` format used by the `mrs_serial` node.
The button on the PCB serves to "mute" the UVLEDs. This button toggles global enable flag in order to stop the blinking, but does not change any other settings and does not stop the counter. This flag can be set using the UART as well.

All setting are stored in the internal FLASH memory and are retrieved on startup.

The byte structure of the `baca_protocol` can be seen in table 1. The message starts with letter 'b' (0x62 in hex) and is followed by the length of the payload (1-255). The payload then follows. The first byte of the payload is the `Message ID` and the rest of the payload contains the data (if any). The last byte is the checksum which is a simple sum of all previous bytes.

Several message types have been implemented to control the board.

Byte	0	1	2 ... (n-1)	n
Value	'b' (0x62)	Length of the payload	Payload (Message ID and Data)	Checksum

Table 1: MRS MODULE - UVDAR: Baca Protocol Message

4.1 Message Types

The available messages are described below:

- **Global Enable LEDs** - This is a software version of the HW switch on the PCB - This overrides, but saves all configuration and turns off all UVLEDs, or resumes the original settings.

Command Byte: 0x90
Payload Size: 0

- **Bulk Enable LEDs** - Enables or Disables the UVLED with the same index as the payload Byte. This will reset counters of all LEDs.

Command Byte: 0x91
Payload size: 8 B
Payload Bytes: 0x00 (UVLED Disabled), 0x01 (UVLED Enabled)
Payload Byte 0: UVLED1 enabled = 0x00 (Disabled), 0x01 (Enabled)
Payload Byte 1: UVLED2 enabled = 0x00 (Disabled), 0x01 (Enabled)
Payload Byte 2: UVLED3 enabled = 0x00 (Disabled), 0x01 (Enabled)
Payload Byte 3: UVLED4 enabled = 0x00 (Disabled), 0x01 (Enabled)
Payload Byte 4: UVLED5 enabled = 0x00 (Disabled), 0x01 (Enabled)
Payload Byte 5: UVLED6 enabled = 0x00 (Disabled), 0x01 (Enabled)
Payload Byte 6: UVLED7 enabled = 0x00 (Disabled), 0x01 (Enabled)
Payload Byte 7: UVLED8 enabled = 0x00 (Disabled), 0x01 (Enabled)

- **Bulk Set Periods** - Sets blinking freqeuncy of the LED with the same index as the payload Byte. This will reset counters of all LEDs.

Command Byte: 0x92
Payload size: 8 B
Payload Byte 0: Desired frequency of the UVLED1
Payload Byte 1: Desired frequency of the UVLED2
Payload Byte 2: Desired frequency of the UVLED3
Payload Byte 3: Desired frequency of the UVLED4
Payload Byte 4: Desired frequency of the UVLED5
Payload Byte 5: Desired frequency of the UVLED6
Payload Byte 6: Desired frequency of the UVLED7
Payload Byte 7: Desired frequency of the UVLED8

- **Enable LED** - enables the UVLED with the specified index. This will reset the counter of the specified UVLED.

Command Byte: 0x93
Payload Size: 2
Payload Byte 0: UVLED index (0-7)
Payload Byte 1: 0x00 (Disabled), 0x01 (Enabled)

- **Set Period** - Sets blinking frequency of the UVLED with the specified index. This will reset the counter of the specified UVLED.

Command Byte: 0x94
Payload Size: 2
Payload Byte 0: LED index (0-7)
Payload Byte 1: Desired frequency of the UVLED

- **Get Enabled** - Returns 8 bytes. These bytes indicate if the UVLEDs are enabled or disabled.

Command Byte: 0x95
Payload Size: 0
Return Payload Size: 8
Return Payload Byte 0: UVLED1 enabled = 0x00 (Disabled), 0x01 (Enabled)
Return Payload Byte 1: UVLED2 enabled = 0x00 (Disabled), 0x01 (Enabled)
Return Payload Byte 2: UVLED3 enabled = 0x00 (Disabled), 0x01 (Enabled)
Return Payload Byte 3: UVLED4 enabled = 0x00 (Disabled), 0x01 (Enabled)
Return Payload Byte 4: UVLED5 enabled = 0x00 (Disabled), 0x01 (Enabled)
Return Payload Byte 5: UVLED6 enabled = 0x00 (Disabled), 0x01 (Enabled)
Return Payload Byte 6: UVLED7 enabled = 0x00 (Disabled), 0x01 (Enabled)
Return Payload Byte 7: UVLED8 enabled = 0x00 (Disabled), 0x01 (Enabled)

- **Get Periods** - Returns 8 bytes. These bytes represent the frequency of the UVLEDs.

Command Byte: 0x96
Payload Size: 0
Return Payload Size: 8
Return Payload Byte 0: UVLED1 Frequency
Return Payload Byte 1: UVLED2 Frequency
Return Payload Byte 2: UVLED3 Frequency
Return Payload Byte 3: UVLED4 Frequency
Return Payload Byte 4: UVLED5 Frequency
Return Payload Byte 5: UVLED6 Frequency
Return Payload Byte 6: UVLED7 Frequency
Return Payload Byte 7: UVLED8 Frequency

4.2 Implementation notes

The Board is equipped with the STM32F042K6T6 microcontroller. Its firmware is programmed using STM32Cube Hardware Abstraction Layer Libraries.
The pinout of the MCU can be seen in the Figure **??**.

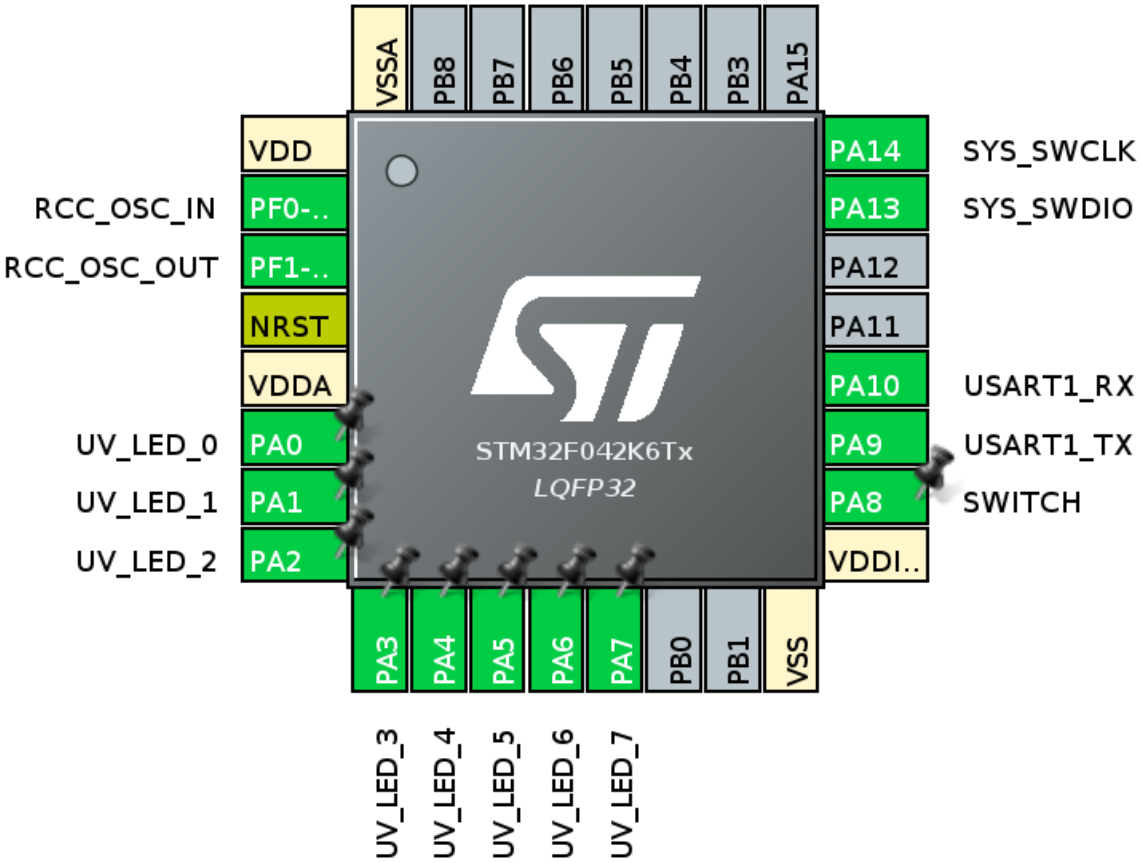


Figure 6: MRS MODULE - UVDAR: MCU pinout

The firmware is written with the need for versatility in mind and therefore is not optimal for specialised applications. The UVLED blinking is implemented using software PWM which allows us to change frequencies of individual channels, as is desired for research purposes. I suggest using hardware PWMs if the UVLEDs will have the same frequency. This is achievable only for UVLEDs 1 to 4 since these are connected to TIM2 channels 1 to 4. The other UVLED channels were moved to arbitrary pins for easier routing in the beginning and were left for legacy purposes. If hardware PWM for all channels is desired, then rearanging UVLEDs is necessary.

4.3 Software PWM for UVLEDs

The UVLED blinking is implemented in `uv_led_driver.c/h` files. To function properly the pins must be named as `UV_LED_x` where `x` is a number between 0 and 8. Of course, this can be changed by the programmer. These pins are used in the `uv_led_init( void )` function to map them to an internally used array. This function also reads the frequency values and flags for enabling UVLEDs from the internal FLASH and resets the counters.

The `uv_led_set_frequency(uint8_t led_id, uint8_t frequency)` function sets the frequency of the selected UVLED. The input argument `led_id` is index of the UVLED to be set starting from 0 and the `frequency` is an integer value of desired frequency in range 1 to 60 Hz. This function then calculates the number of ticks of the counter needed to toggle the LED (half of its total period). To calculate this value properly, the value of `TIMER_FREQ` must be set to the frequency of the hardware timer used to increment the counter.
The calculated value is then saved to the internal FLASH.

The `uv_led_enable(uint8_t led_id, uint8_t enable)` function sets the enable flag of the selected UVLED. The input argument `led_id` is index of the UVLED to be set starting from 0, with `enable` as an integer of value 1 (true) or 0(false). Any other values are invalid.
The enable flag is then saved to the internal FLASH.

The `uv_led_toggle(uint8_t led_id)` function toggles the selected UVLED. The input argument `led_id` is index of the UVLED to be set starting from 0.
This function is called if the counter value reaches the desired value. This is managed using the Timer Period Elapsed Callback Function `HAL_TIM_PeriodElapsedCallback` where the counter is increased everytime the used timer's period elapses. The used timer in this appliation is `TIM3` with a frequency of 10kHz.

4.3.1 Global Enable Flag

To be able to disable all UVLEDs on a button switch without affecting anything else, a simple flag was created. This flag toggles on with a button press (`SWITCH`) or with receiving a global enable message using UART. This flag is used in the `uv_led_toggle(uint8_t led_id)`.

4.4 UART implementation

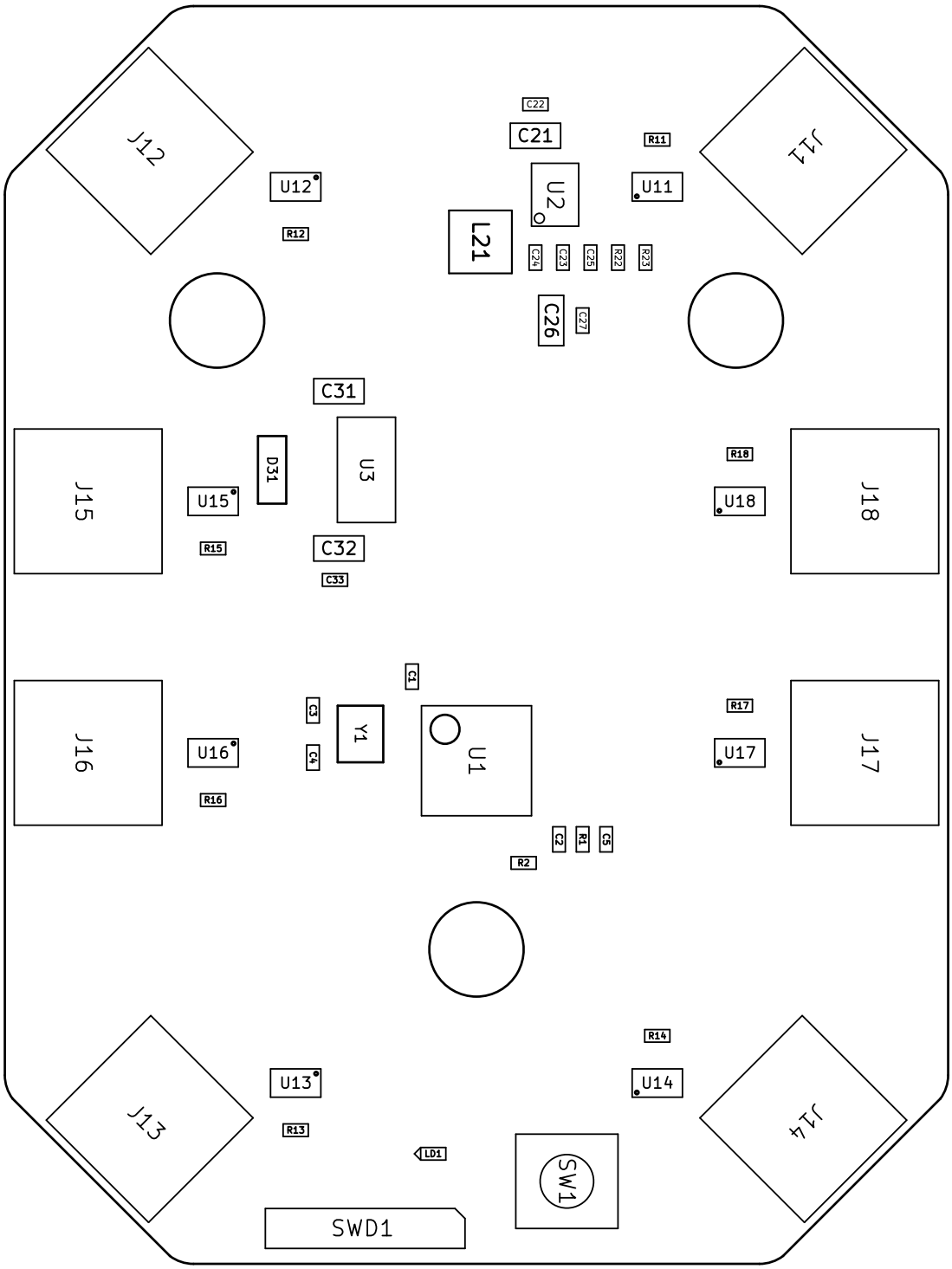
The UART communication through USART1 is implemented using DMA with enabled IDLE interrupt for variable packet size. The message is then processed in `USAR_UART_IDLECallback` where the data are passed to the `process_uart_message()` function to check if the message has the correct format (if the checksum is correct and has one of the valid message IDs) and then calls the apropriate function.
The buffer is then cleared.

5 Bill Of Materials and Assembly Drawings

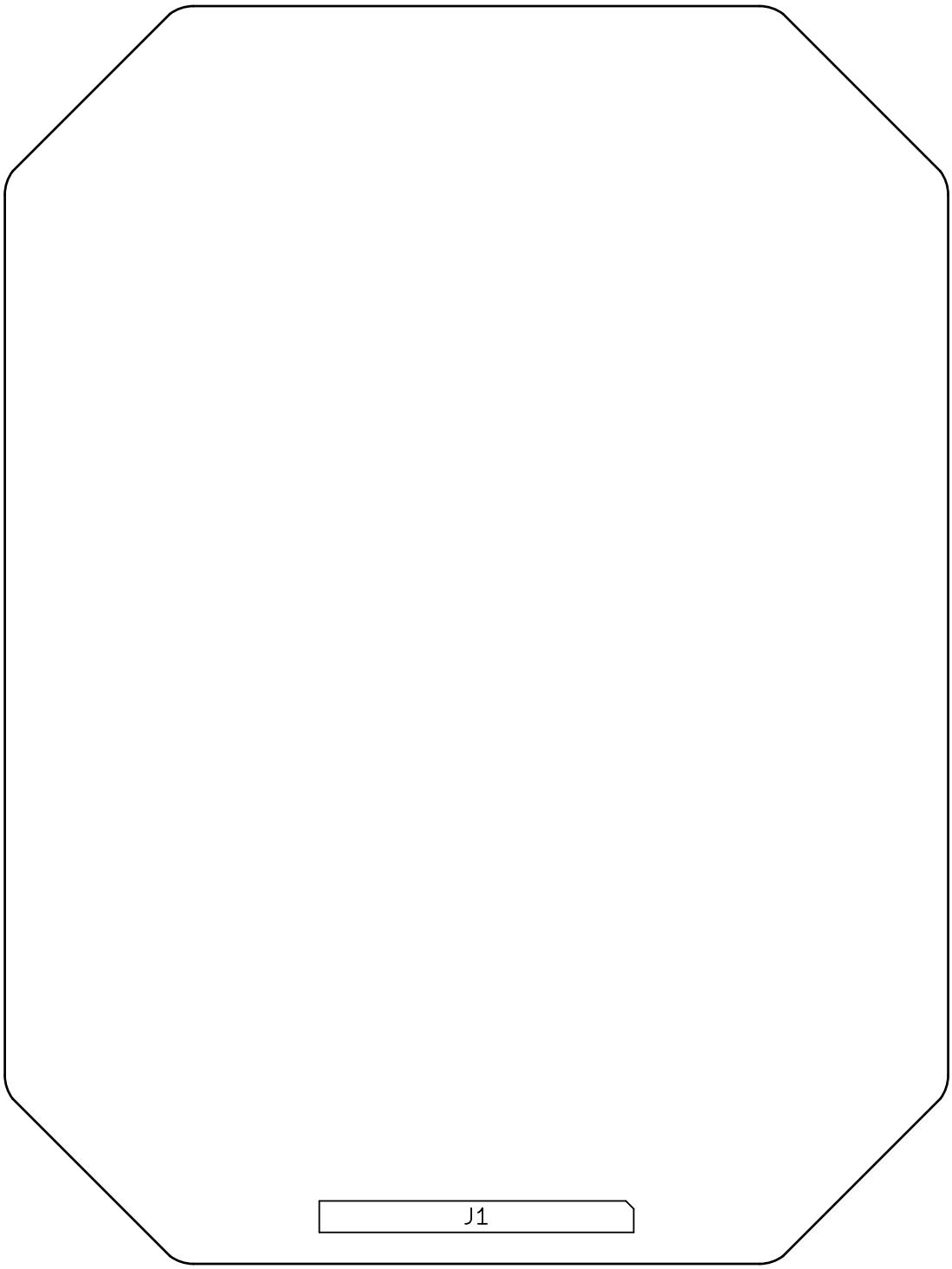
References	Value	Footprint	Quantity
C1, C28,	1uF	0603	2
C2, C33, C5, C25,	0.1uF	0603	4
C4, C3, C22,	10pF	0603	3
C23,	1nF	0603	1
C24,	10nF	0603	1
C26,	100uF/10V	2312	1
C27, C21,	22uF/25V	2412	2
C31, C32,	10uF	1206	2
D21,	B340A	SMA	1
D31,	1N4148W-E3-08	SOD-123	1
J1,	Header_1x10	Header_1x10_P2.00mm	1
J11, J12, J13, J15, J16, J17, J18, J14,	JST-PH_2	JST_PH_2	8
L21,	SRP6540-8R2M	SRP6540	1
LD1,	GREEN	0603	1
R1, R22,	10k	0603	2
R2, R23,	1k	0603	2
R11, R12, R13, R14, R15, R16, R17, R18,	3R3-0.5%	0603	8
R21,	27k4	0603	1
SW1,	Push-Switch_SMT_6x6mm	Tactile_Switch_6x6_SMT	1
SWD1,	SWO	Header_1x05_P2.54mm	1
U1,	STM32F042K6T6	LQFP-32_7x7mm_P0.8mm	1
U2,	TPS54331DR	SOIC-8	1
U3,	MCP1825S-3302E	SOT-223-3(TAB-pin-2)	1
U11, U12, U13, U14, U15, U16, U17, U18,	BCR421UW6	SOT-26-6	8
Y1,	16 MHz	ABM8G	1

Table 2: MRS MODULE - UVDAR: Bill Of Materials

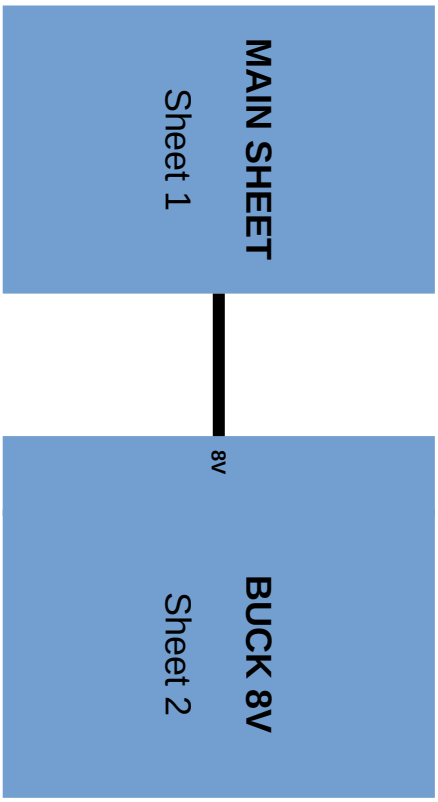
ASSEMBLY DRAWING: TOP

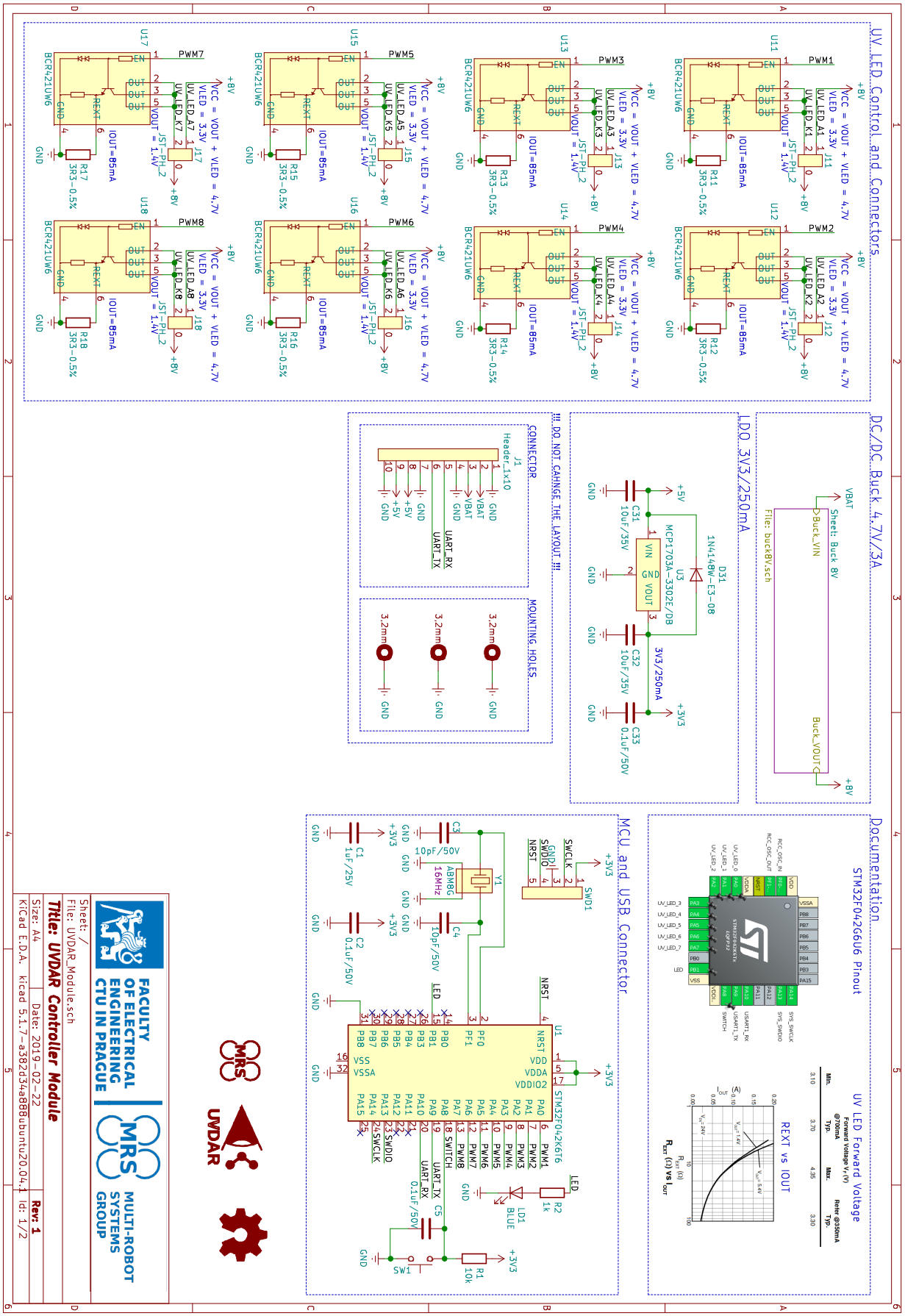


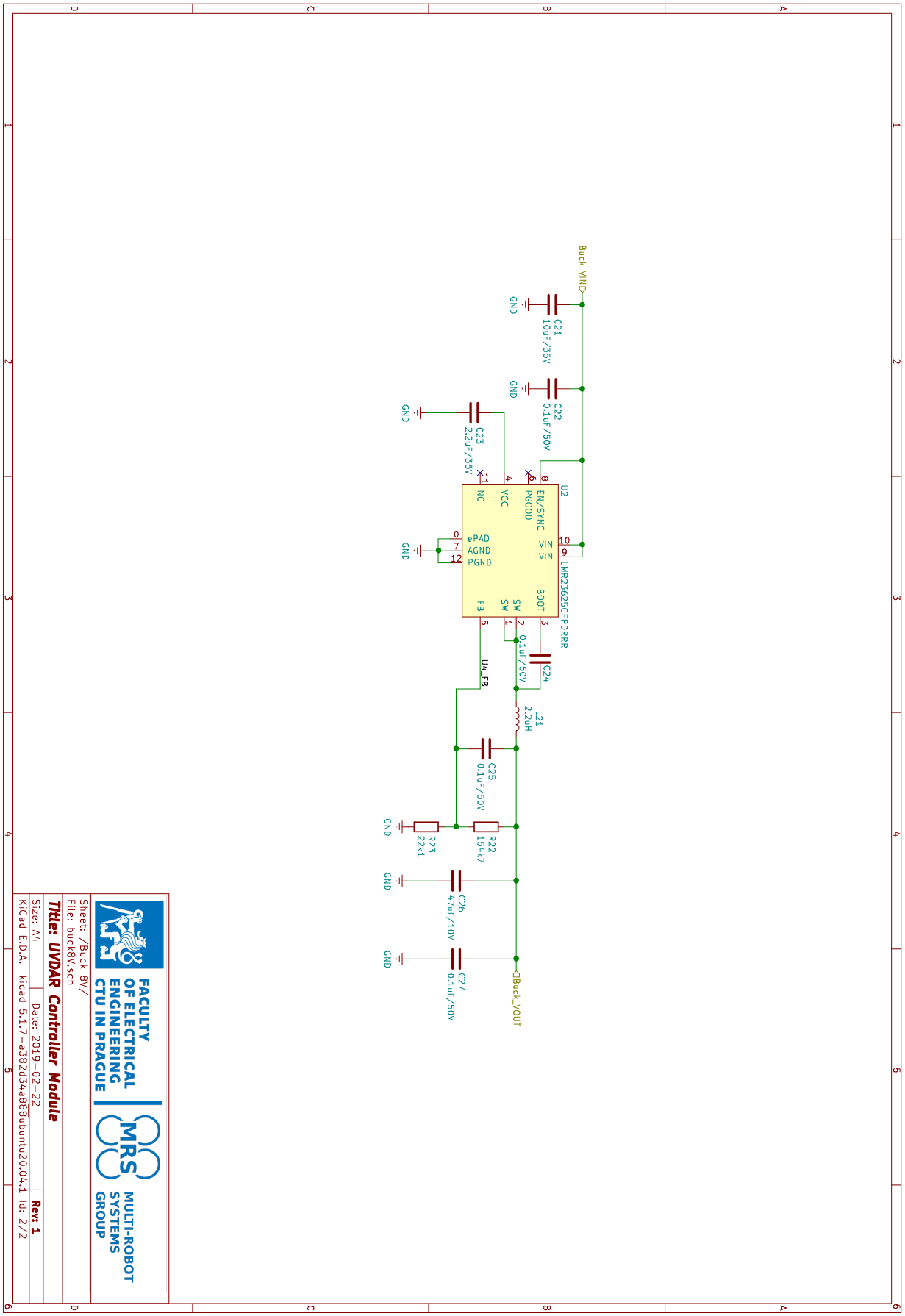
ASSEMBLY DRAWING: BOT



6 Schematic




UV_LED_Control_and_Connectors
DC/DC_Buck_4.7V/3A
LDO_3V3/250mA
!!! DO_NOT_CHANGE_THE_LAYOUT !!!
MCU_and_USB_Connector
Documentation
REXT vs IOUT
UV LED Forward Voltage
STM32F042G6U6 Pinout
FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE
MULTI-ROBOT SYSTEMS GROUP
UVDAR
Title: UVDAR Controller Module
Sheet: /
File: UVDAR_Module.sch
Size: A4
Date: 2019-02-22
Rev: 1
KiCad E.D.A. kicad 5.1.7-a382d34a88~ubuntu20.04.1
Id: 1/2


Buck_VIN
C21
10uF/35V
C22
0.1uF/50V
C23
2.2uF/35V
U2
LMR23625CFPDRRR
EN/SYNC
PGOOD
VCC
NC
ePAD
AGND
PGND
VIN
VIN
BOOT
SW
SW
FB
C24
0.1uF/50V
L21
2.2uH
U4_FB
C25
0.1uF/50V
R23
22k1
R22
154k7
C26
47uF/10V
C27
0.1uF/50V
Buck_VOUT
GND
FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE
MULTI-ROBOT SYSTEMS GROUP
Sheet: /Buck 8V/
File: buck8V.sch
Title: UVDAR Controller Module
Size: A4
Date: 2019-02-22
Rev: 1
KiCad E.D.A. kicad 5.1.7-a382d34a8~ubuntu20.04.1
Id: 2/2