STANDALONE UVDAR CONTROLLER REVISION 3

David Zaitlik

Multi-Robot Systems Group, FEE CTU in Prague

FACULTY
OF ELECTRICAL
ENGINEERING
CTU IN PRAGUE

MRS

MULTI-ROBOT
SYSTEMS
GROUP

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 a standalone UVDAR controller designed for simple development of the UVDAR system.

The controller communicates using USB. The power for the PCB is provided by the `J3` connector.

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

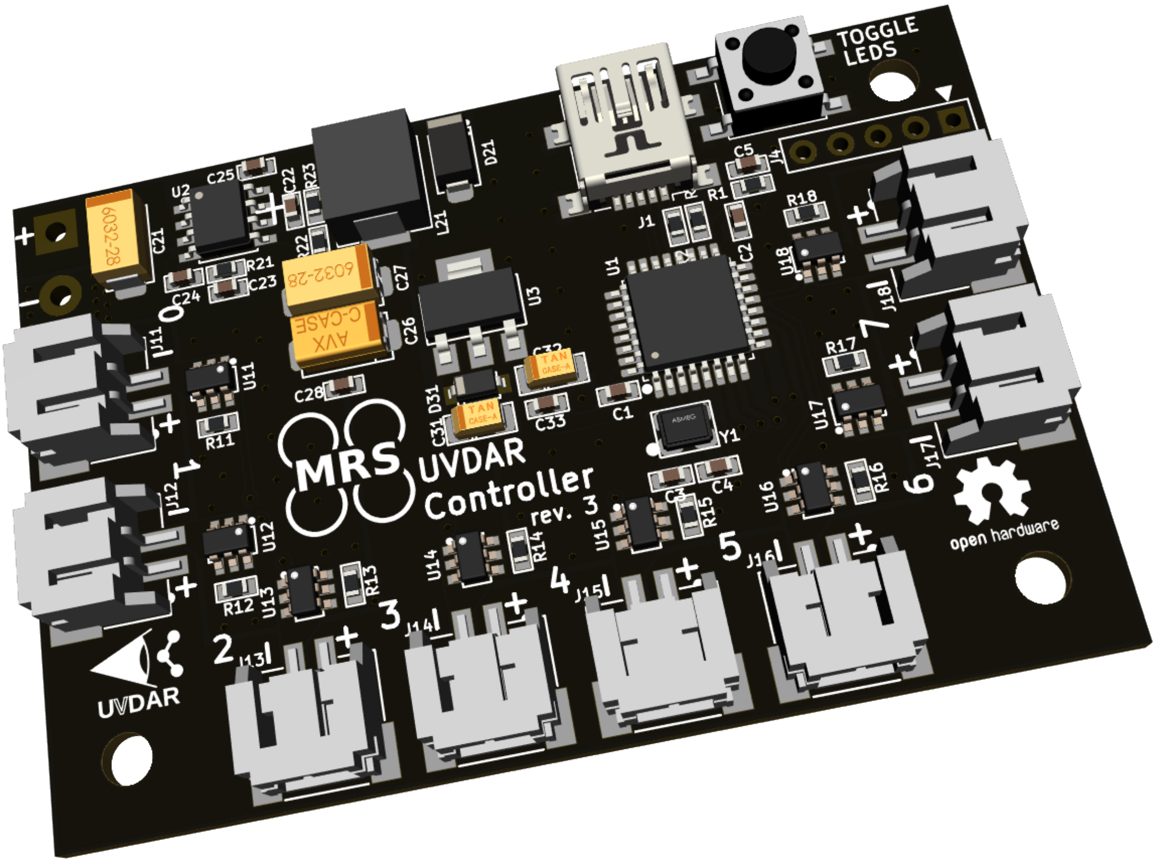


Figure 1: STANDALONE UVDAR CONTROLLER: Thumbnail

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.

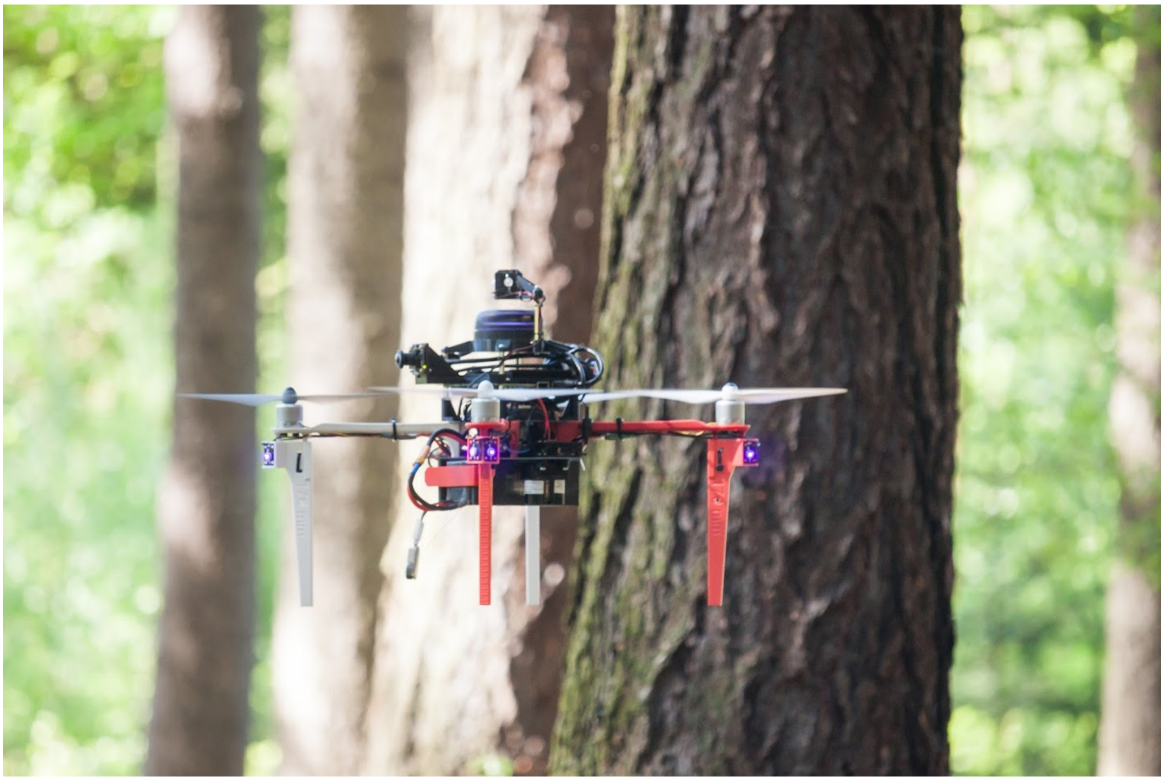


Figure 2: STANDALONE UVDAR CONTROLLER: Photo of an F450 drone with UVDAR

2 Dimensions

The Board is 67mm long and 45mm wide with three M3 mounting holes.

The mechanical drawing can be seen in Figure 3 below.

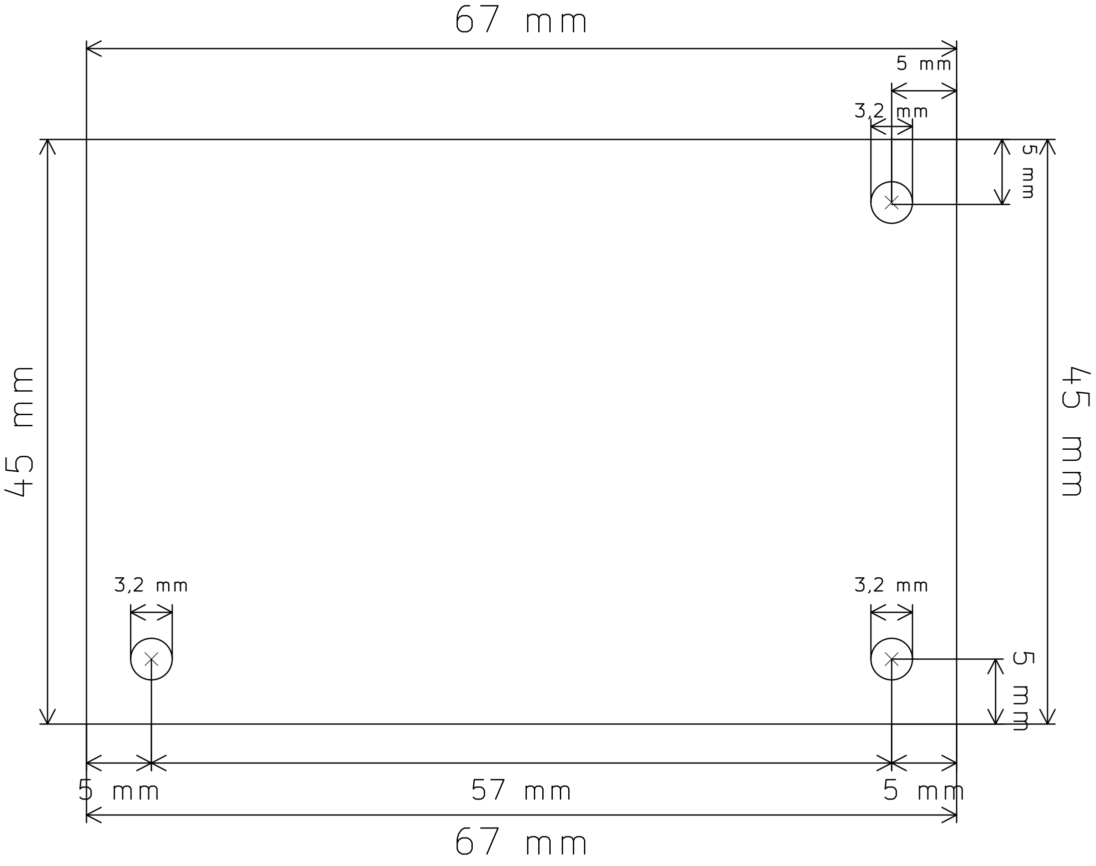


Figure 3: STANDALONE UVDAR CONTROLLER: Dimensions

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 ??.

3.1 Connector Placement Drawing

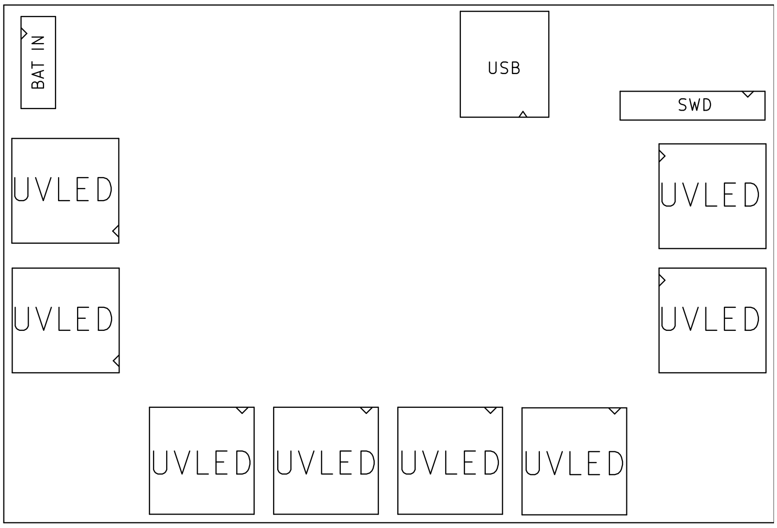


Figure 4: STANDALONE UVDAR CONTROLLER: Connectors Placement, Top Layer

3.2 Connector Pinouts

UVLED	
1	LED ANODE
2	LED KATHODE

BAT IN	
1	BAT+
2	BAT-

USB	
1	VUSB
2	D-
3	D+
4	NC
5	GND

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

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 USB; 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: STANDALONE UVDAR CONTROLLER: 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 ??.

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.

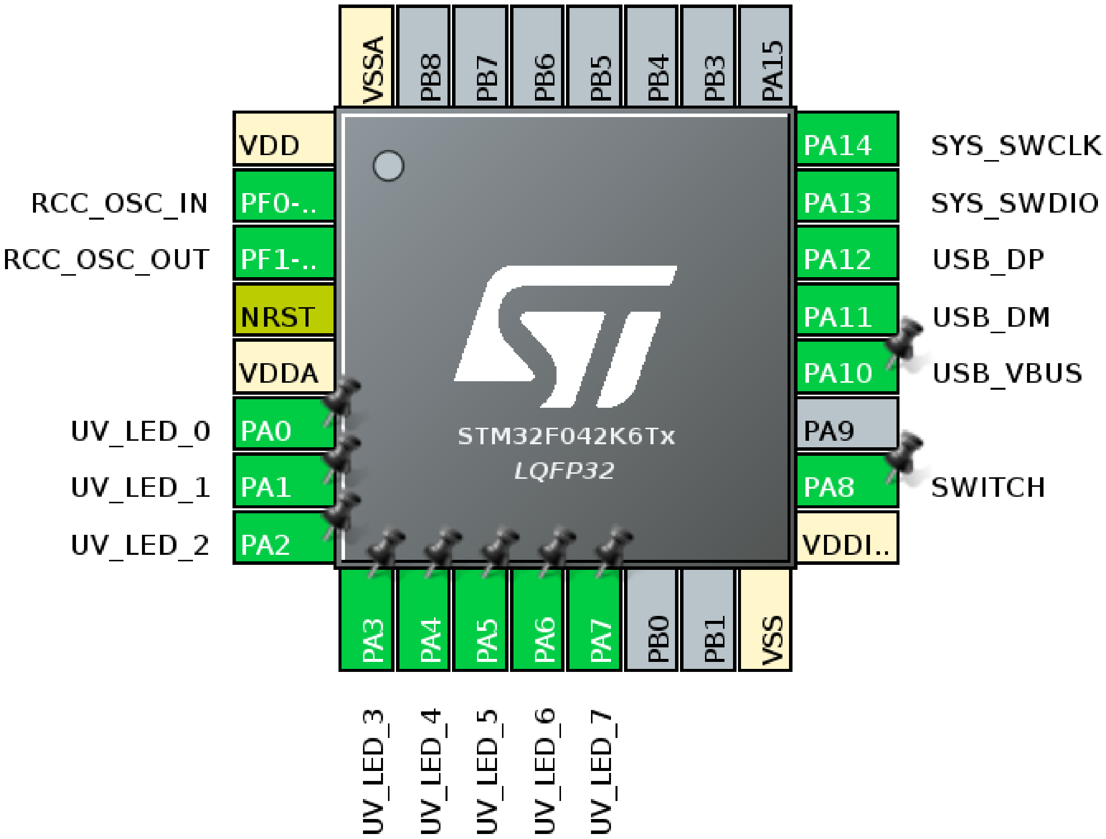


Figure 5: STANDALONE UVDAR CONTROLLER: MCU pinout

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. This may of course be 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 USB implementation

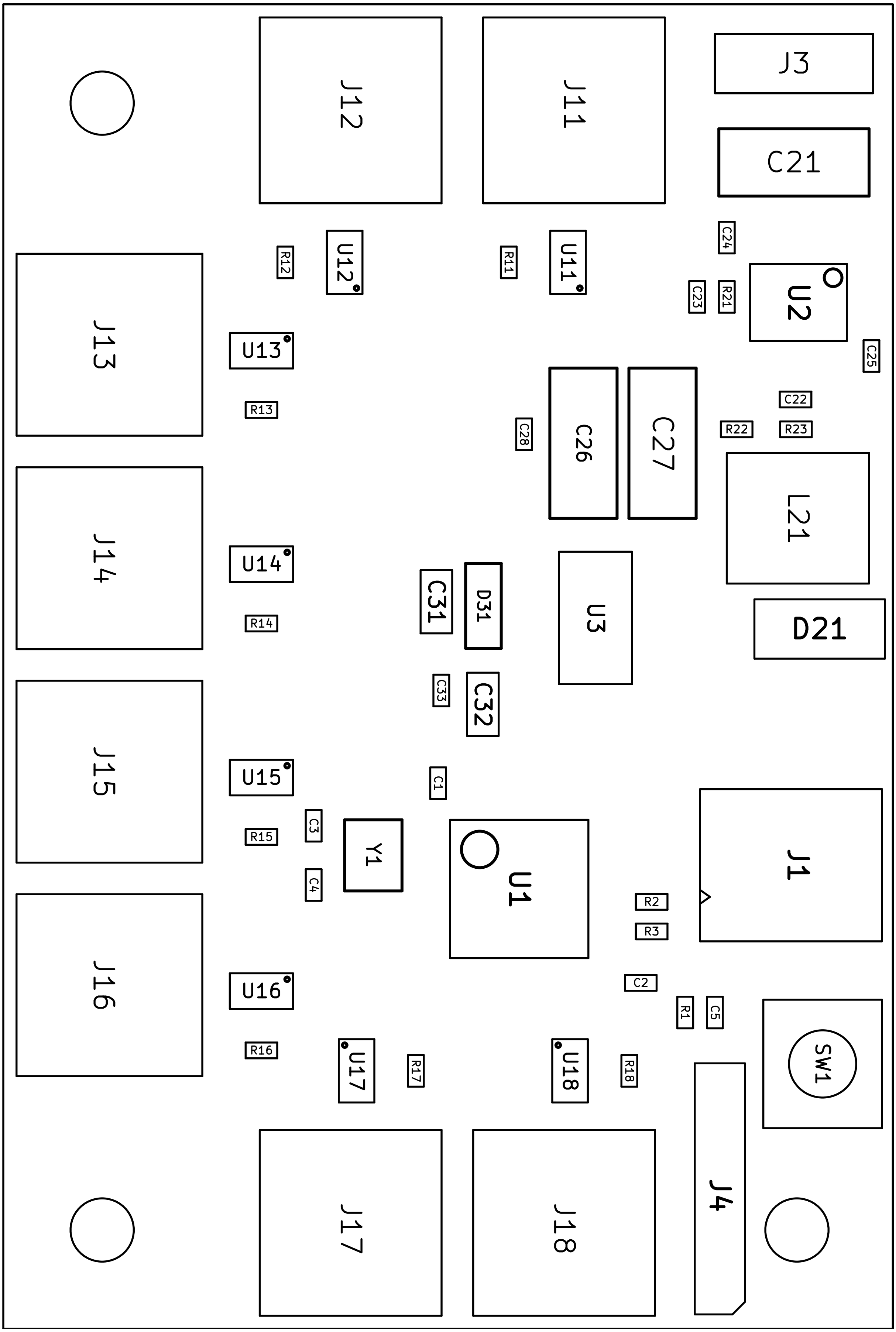
TODO: Rewrite to USB The USB communication using the STM Middleware. The message is processed in `CDC_Receive_FS` function in `usbd_cdc_if.c` file. If the message has the correct format (if the checksum is correct and has one of the valid message IDs), the apropriate function is called.

VBUS Sensing has not yet been implemented.

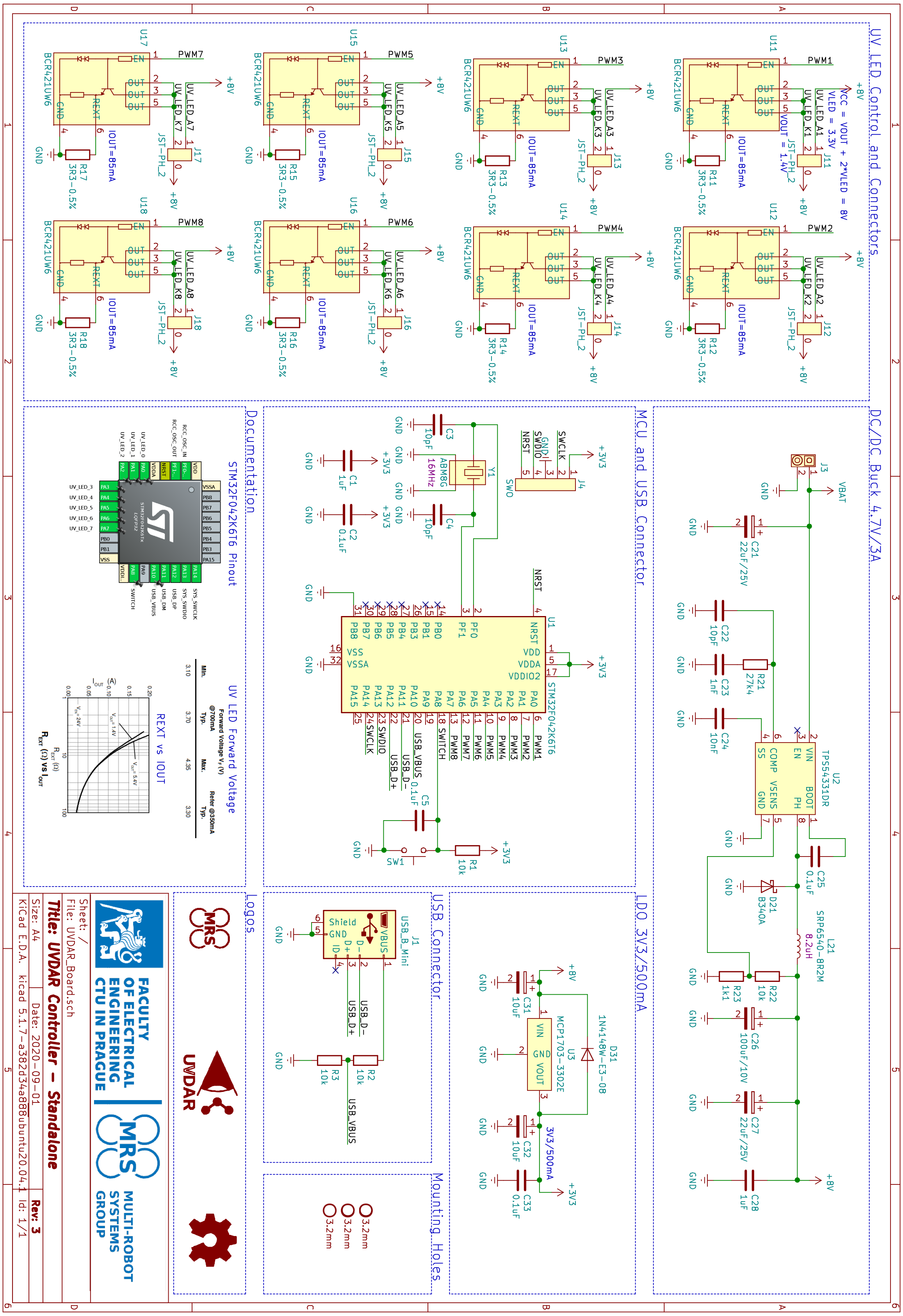
5 Bill Of Materials and Assembly Drawings

References	**Value**	**Footprint**	**Quantity**
C2, C5, C25, C33	0.1uF	C0603	4
C3, C4, C22	10pF	C0603	3
C1, C28	1uF	C0603	2
C23	1nF	C0603	1
C24	10nF	C0603	1
C31, C32	10uF	CTAN1206	2
C26	100uF/10V	CTAN2312	1
C21, C27	22uF/25V	CTAN2412	2
R11, R12, R13, R14, R15, R16, R17, R18	3R3-0.5%	R0603	8
R1, R2, R3, R22	10k	R0603	4
R21	27k4	R0603	1
R23	1k1	R0603	1
L21	SRP6540-8R2M	L_SRP6540	1
D21	B340A	D_SMA	1
D31	1N4148W-E3-08	SOD-123	1
U1	STM32F042K6T6	LQFP-32_7x7mm_P0.8mm	1
U2	TPS54331DR	SOIC-8	1
U3	MCP1703-3302E	SOT-223-3(TAB-pin-2)	1
U11, U12, U13, U14, U15, U16, U17, U18	BCR421UW6	SOT-26-6	8
Y1	ABM8G	ABM8G	1
SW1	Push-Switch_SMT_6x6mm	Tactile_Switch_6x6_SMT	1
J4	SWO	Header_1x05_P2.54mm	1
J11, J12, J13, J14, J15, J16, J17, J18	JST-PH_2	JST_PH_2	8
J1	USB_B_Mini	USB_Mini-B_Lumberg_2486_01_Horizontal	1

Table 2: STANDALONE UVDAR CONTROLLER: Bill Of Materials


J12
J11
J3
C21
C24
R12
U12
R11
U11
C23
R21
U2
C25
J13
U13
R13
C22
R22
R23
C28
C26
C27
L21
J14
U14
R14
C31
D31
U3
D21
C33
C32
J15
U15
R15
C1
C3
Y1
U1
J1
C4
R2
R3
J16
U16
C2
R1
C5
R16
U17
R17
U18
R18
SW1
J17
J18
J4

6 Schematic


UV LED Control and Connectors
VCC = VOUT + 2*VLED = 8V
VLED = 3.3V
VOUT = 1.4V
U11 U12 U13 U14 U15 U16 U17 U18
BCR421UW6
EN OUT OUT OUT GND REXT
PWM1 PWM2 PWM3 PWM4 PWM5 PWM6 PWM7 PWM8
UV_LED_A1 UV_LED_K1 UV_LED_A2 UV_LED_K2 UV_LED_A3 UV_LED_K3 UV_LED_A4 UV_LED_K4
UV_LED_A5 UV_LED_K5 UV_LED_A6 UV_LED_K6 UV_LED_A7 UV_LED_K7 UV_LED_A8 UV_LED_K8
J11 J12 J13 J14 J15 J16 J17 J18
JST-PH_2
IOUT=85mA
R11 R12 R13 R14 R15 R16 R17 R18
3R3-0.5%
+8V
GND
DC/DC Buck 4.7V/3A
J3
VBAT
C21 22uF/25V
C22 10pF
C23 1nF
R21 27k4
C24 10nF
U2 TPS54331DR
VIN EN COMP SS BOOT PH VSENS GND
C25 0.1uF
D21 B340A
L21 SRP6540-8R2M 8.2uH
R22 10k
R23 1k1
C26 100uF/10V
C27 22uF/25V
C28 1uF
MCU and USB Connector
J4 SWD
SWCLK GND SWDIO NRST
+3V3
Y1 ABM8G 16MHz
C3 10pF
C4 10pF
C1 1uF
C2 0.1uF
U1 STM32F042K6T6
NRST VDD VDDA VDDIO2 VSS VSSA PF0 PF1 PB0 PB1 PB3 PB4 PB5 PB6 PB7 PB8
PA0 PA1 PA2 PA3 PA4 PA5 PA6 PA7 PA8 PA9 PA10 PA11 PA12 PA13 PA14 PA15
SWITCH USB_VBUS USB_D- USB_D+ SWDIO SWCLK
C5 0.1uF
SW1
R1 10k
Documentation
STM32F042K6T6 Pinout
UV LED Forward Voltage
REXT vs IOUT
LDO 3V3/500mA
D31 1N4148W-E3-08
U3 MCP1703-3302E
VIN GND VOUT
C31 10uF
C32 10uF
C33 0.1uF
3V3/500mA
USB Connector
J1 USB_B_Mini
Shield GND VBUS D- D+ ID
R2 10k
R3 10k
Logos
UVDAR
Mounting Holes
3.2mm
FACULTY OF ELECTRICAL ENGINEERING CTU IN PRAGUE
MRS MULTI-ROBOT SYSTEMS GROUP
Sheet: /
File: UVDAR_Board.sch
Title: UVDAR Controller – Standalone
Size: A4 Date: 2020-09-01 Rev: 3
KiCad E.D.A. kicad 5.1.7-a382d34a8 ubuntu20.04.1 Id: 1/1