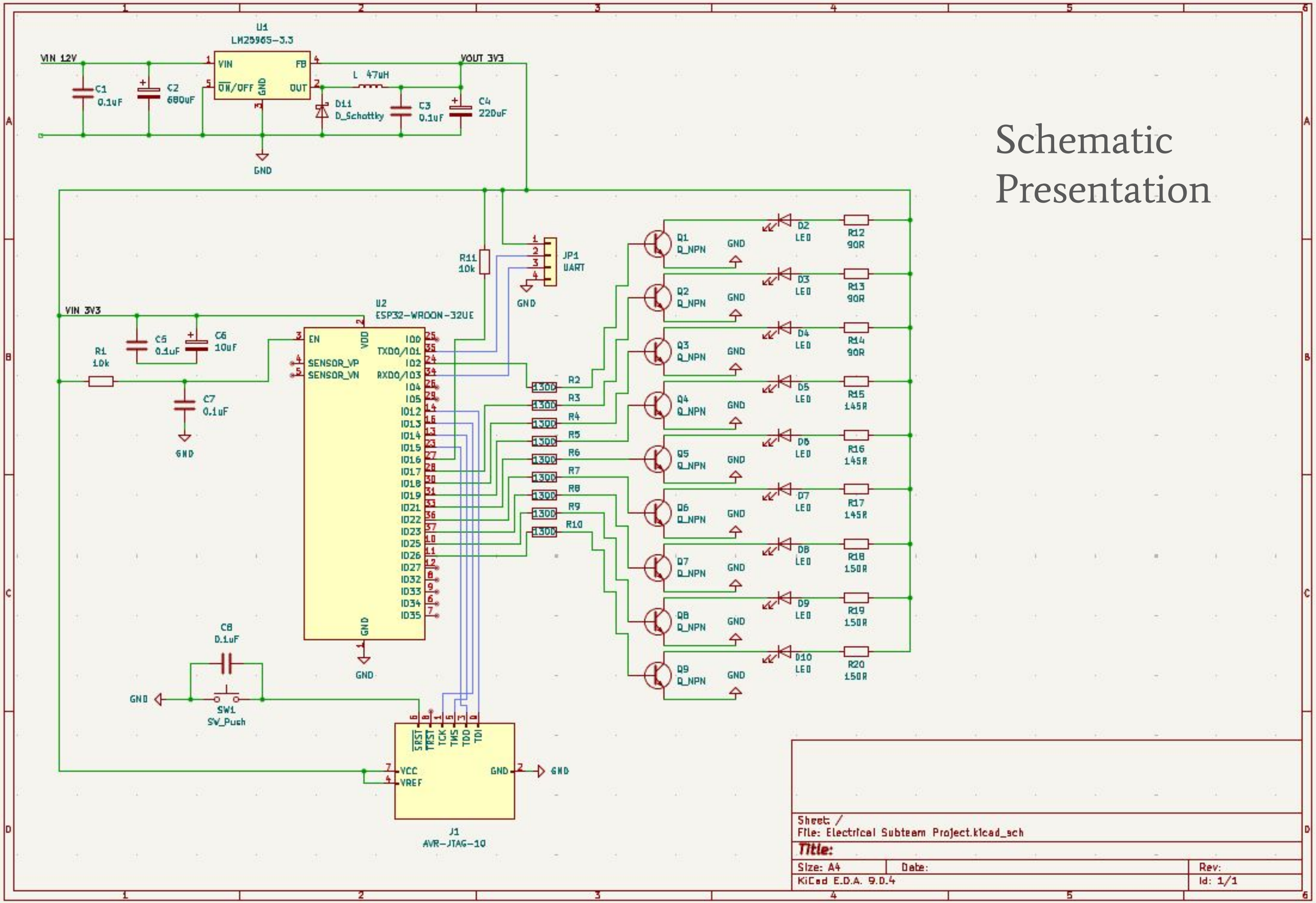
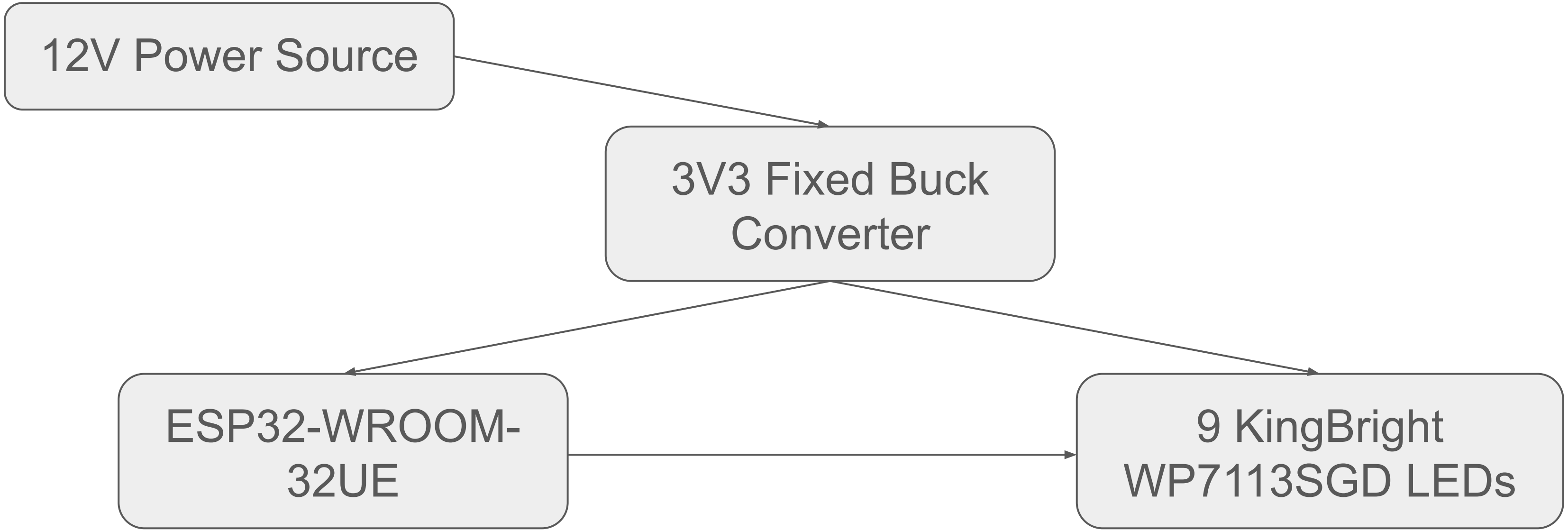
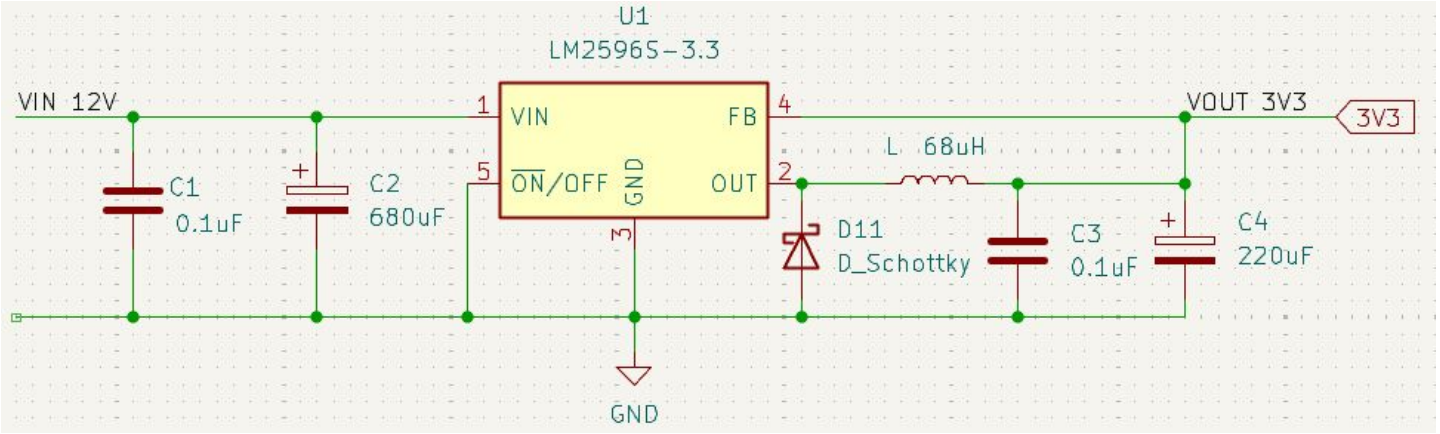
Schematic Presentation



Block Diagram



Buck Converter



Design choices:

- Fixed 12V to 3.3V, simpler feedback loop
- Efficient voltage conversion (73% according to data sheet)
- Capacitor and inductor values chosen based off of calculated current draw
- Added unpolarized capacitor for additional decoupling

Buck Converter (cont.)

SYSTEM PARAMETERS (Note 5) Test Circuit *Figure 1*

V_{OUT}	Output Voltage	$4.75V \le V_{IN} \le 40V$, $0.2A \le I_{LOAD} \le 3A$	3.3	3.168/**3.135** 3.432/**3.465**	V V(min) V(max)
η	Efficiency	$V_{IN} = 12V$, $I_{LOAD} = 3A$	73		%

Table 15: Current Consumption Depending on RF Modes

Work mode	Description		Average (mA)	Peak (mA)
Active (RF working)	TX	802.11b, 20 MHz, 1 Mbps, @19.5 dBm	239	379
		802.11g, 20 MHz, 54 Mbps, @15 dBm	190	276
		802.11n, 20 MHz, MCS7, @13 dBm	183	258
		802.11n, 40 MHz, MCS7, @13 dBm	165	211
	RX	802.11b/g/n, 20 MHz	112	112
		802.11n, 40 MHz	118	118

Buck Converter (cont.)

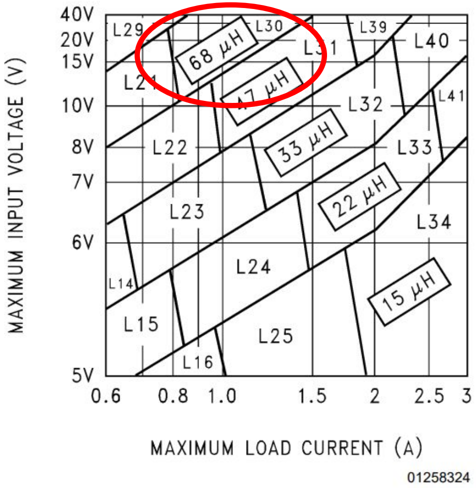


FIGURE 4. LM2596-3.3

Absolute Max Current Draw Calculations:

$$379mA + 3(\frac{62.5mW}{3.3V}) + 3(\frac{75mW}{3.3V}) + 3(\frac{75mW}{3.3V}) = 572.18mA \approx 600mA$$

Capacitor values are safest for voltage and current ratings (also from datasheet)

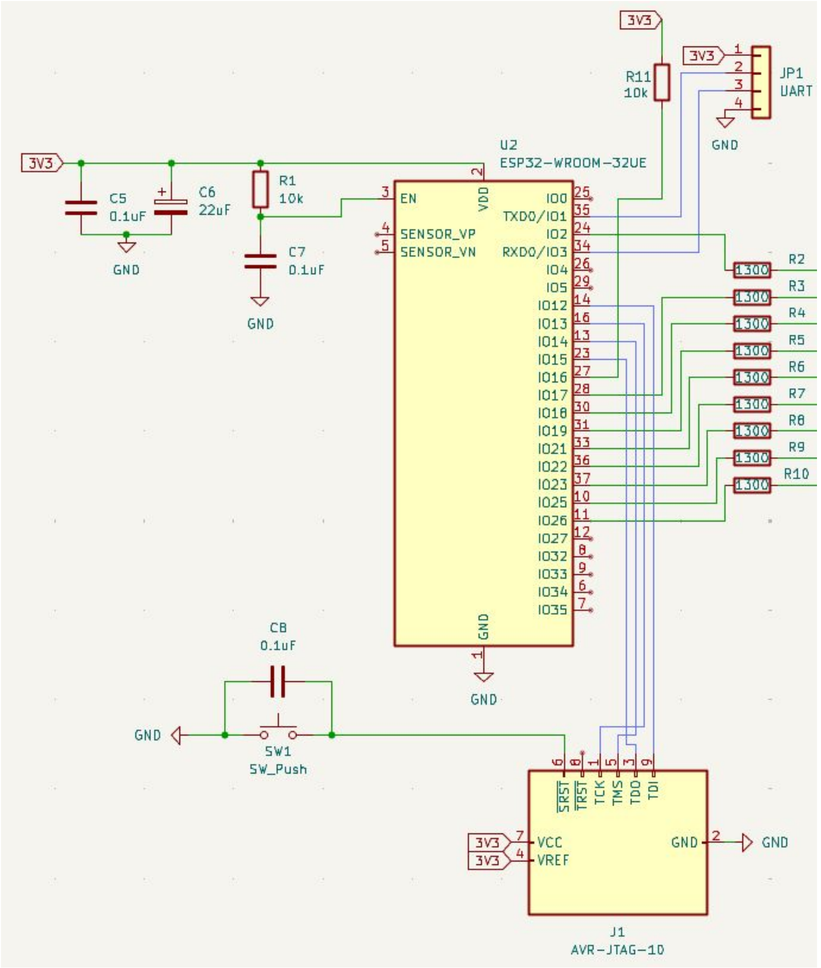
ABSOLUTE MAXIMUM RATINGS at T_A=25°C

Parameter	Symbol	Value	Unit
Power Dissipation	P_D	62.5	mW

ESP32-WROOM-32UE Microcontroller

Design Choices:

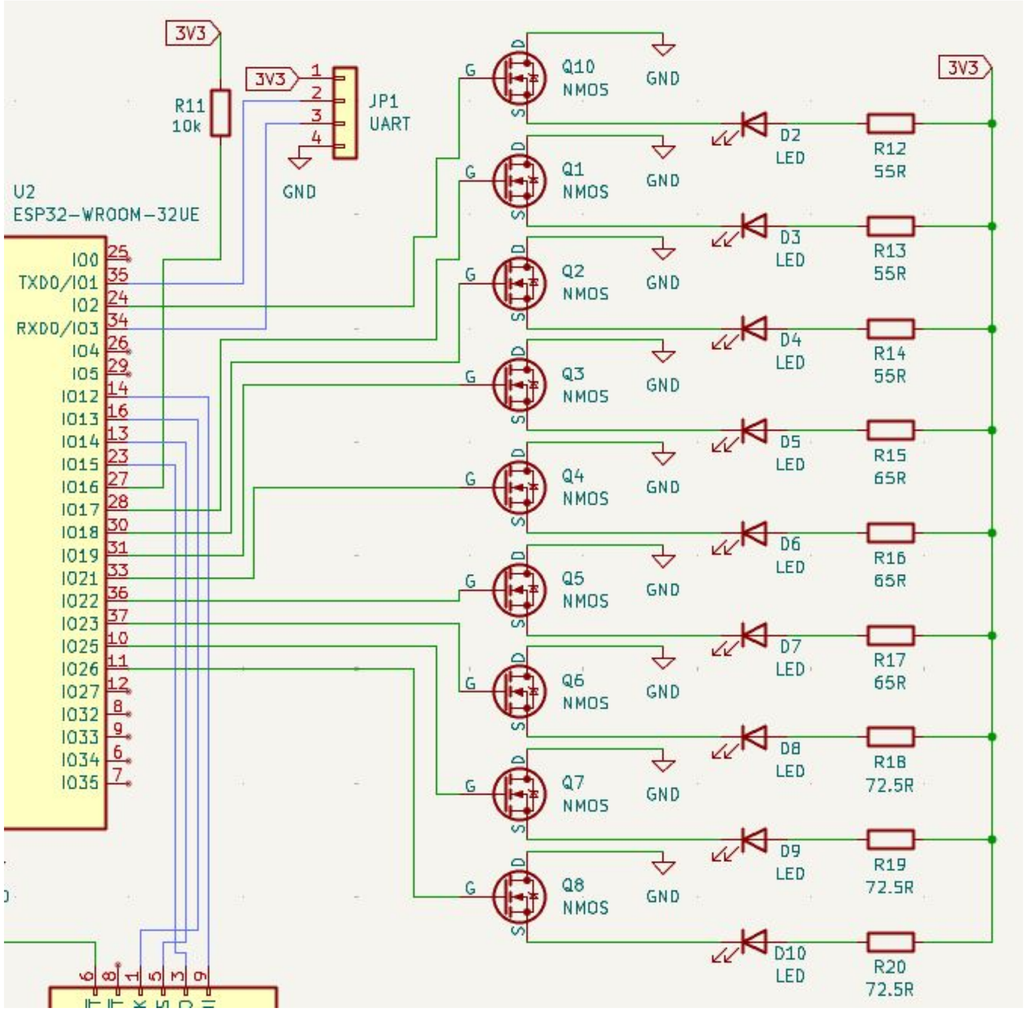
- Decouplers to smooth input voltage
- Simple pull up resistor for EN pin
- UART port without USB-to-UART to keep simple
- JTAG for debugging hardware
- All capacitor and resistor values are chosen because they meet minimum requirements, reduce high and low frequency noise, while providing leeway for voltage spikes



LEDs

Design Choices:

- Used the Kingbright WP7113 series LEDs (3 green, 3 yellow, 3 red)
- Low power usage (193.18 mW)
- Powered by the 3V3 buck converter, not the microcontroller
- Use NMOS MOSFET for current switch (quicker and more efficient than BJT)
- Resistance values are calculated based on LED data sheets



LEDs (cont.)

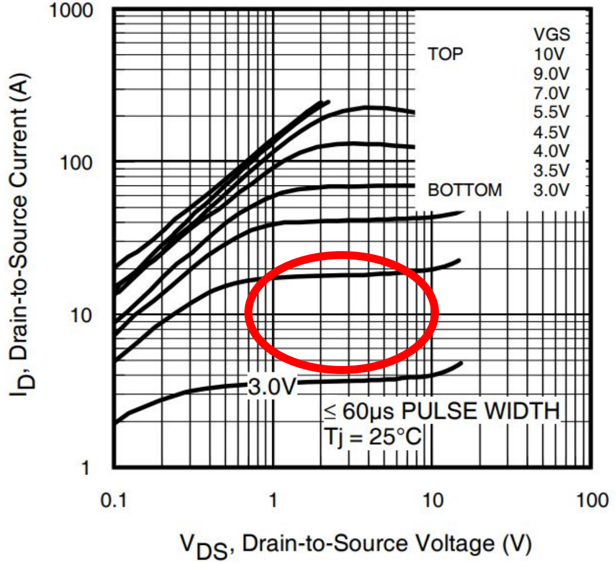
$$R_{green} = \frac{3.3\,\mathrm{V} - 2.2\,\mathrm{V}}{0.020\,\mathrm{A}} = 55\,\Omega$$

$$R_{yellow} = \frac{3.3\,\mathrm{V} - 2.0\,\mathrm{V}}{0.020\,\mathrm{A}} = 65\,\Omega$$

$$R_{red} = \frac{3.3\,\mathrm{V} - 1.85\,\mathrm{V}}{0.020\,\mathrm{A}} = 72.5\,\Omega$$

$$P_{per\_LED} = 3.3\,\mathrm{V} \times 0.020\,\mathrm{A} = 66\,\mathrm{mW}$$

$$P_{total,9LED} = 9 \times 66\,\mathrm{mW} = 594\,\mathrm{mW}$$



Forward Voltage I_F = 20mA	V_F [2]	Super Bright Green	2.2	2.5	V
Forward Voltage I_F = 20mA	V_F [2]	Super Bright Yellow	2	2.5	V
Forward Voltage I_F = 20mA	V_F [2]	Super Bright Red	1.85	2.5	V

Pseudocode for ESP32-WROOM-32UE

```
// Constants
NUM_LEDS = 9
LED_PINS = [pin1, pin2, ..., pin9]        // GPIO pins connected to NMOS gates
RPM_THRESHOLDS = [1000, 2000, ..., 9000]  // Example thresholds for each LED

// Setup
for i in 0..NUM_LEDS-1:
    configurePinAsOutput(LED_PINS[i])
    setPinLow(LED_PINS[i])   // all LEDs off at start

// Main loop
while true:
    rpm = getCurrentRPM()    // assume function exists that always returns latest RPM

    // Turn on LEDs based on rpm thresholds
    for i in 0..NUM_LEDS-1:
        if rpm >= RPM_THRESHOLDS[i]:
            setPinHigh(LED_PINS[i])   // LED ON
        else:
            setPinLow(LED_PINS[i])    // LED OFF

    delay(small_interval)   // e.g., 10-50 ms to limit CPU usage
```