Documentation: Roller Blinds Automatic Control system	
Date:	22.02.2026
Author:	Aksamit Michał

TABLE OF CONTENTS

1. Purpose of work.

The purpose of this work is to show user experience with automatic control systems, especially using Real Time Operating System (RTOS). This is important to note that the subject of this work should be able to be used in real-world applications. The Roller blinds automatic control is usable in that scenario. Additionally, because of motors (moving, dynamic elements) RTOS system would be great for application. It is reliable, deterministic, and safe to use.

2. Scope of work.

There will be a few points found in this work, done in sequence:

- Requirements.
- Electronic diagram with chosen elements.
- Mechanics and construction.
- Block diagram of program.

3. Requirements.

The project has multiple requirements needed to run without issues, including:

- DC Motors with worm gear (needed for self-lock, if not working) having enough torque (estimate 10 kgcm) and encoders, powered with 5-12 V
- DC Motor controller with direction control, cheap, powered with 5-12 V (If 12, then output power option of 5V for powering main controller) and small size
- 2 Switches for manual steering and e-stop button
- Electrical box for modules
- Cables (6x0.5/0.35) and pitch 2.54 mm, male and female connectors
- 3 LED Diodes
- Microcontroller (5 V) with multicore and technical support for RTOS (FreeRTOS), wi-fi and small size
- 3D printed parts for connecting motors with roller slides and to hang motors on the window
- 12 VDC power supply

Because of its DIY project purpose it is important to note that selected parts should be as cheap as possible just to not generate additional costs.

3.1 The Comparison table.

Name	Description	Price
Microcontroller		
ESP32 WiFi + BT 4.2 - platform with module ESP-WROOM-32 compatible with ESP32-DevKit	> Supply voltage: 5 V - with microUSB > Microcontroller Dual Core Tensilica LX6 240 MHz > The SRAM 520 KB > Flash memory: 4 MB > Built-in wi-fi module 802.11 BGN HT40 Wi-fi security: WEP, WPA/WPA2 PSK/Enterprise, AES / SHA2 / Elliptical Curve Cryptography / RSA-4096 > Built-in Bluetooth BLE > Integrated Hall sensor and touch interface > 30 GPIO leads, including: 3x UART 3x SPI 2x I2C (2x I2S) 12 - channel >ADC converter 2 - channel DAC converter	€11.50

	> The PWM outputs > Interface of SD card > Dimensions: 55 x 28 x 8 mm	
DFRobot FireBeetle ESP8266 IoT WiFi	> Supply voltage: 3.3V / 5V > Supports USB charging > Microcontroller: 32-bit Tensilica L106 Frequency: 800 MHz > SRAM: 50KB > Flash: 16Mbit > Low current consumption: 46 uA > Wi-fi: 802.11 b / g / n / > Leads: Digital x 10 Analog x 1 SPI x 1 I2C x 1 IR x 1 I2S x 1 > Supports MicroPython, Arduino IDE and RTOS > Dimensions: 24 x 53 mm	€8.00
NUCLEO-WL55JC1	> Ultra-low-power MCU, 256Kbyte flash memory and 64Kbyte SRAM > 3 user LEDs, 3 user buttons and 1 reset push-button > 32.768KHz LSE crystal oscillator, 32MHz HSE on-board oscillator > USB with Micro-B, MIPI® debug connector > ST morpho extension pin headers for full access to all STM32WL I/Os > ARDUINO® Uno V3 expansion connector, delivered with SMA antenna > Fully open hardware platform > Flexible power-supply options: ST-LINK, USB VBUS, or external sources	€40.00
Motor Controller		
L298N - two-channel motor controller - 12V/2A	> Supply voltage: 12 VDC > Supply voltage of the logic part: 5 VDC > Built-in voltage regulator 5 V to power the logic part with the possibility of disconnection via jumper > Maximum output current: 2 A per channel > Module to control two DC motors or one stepper motor > Direct control and fast,free braking-stop options > The outputs for the motors and the power inputs was displayed on the screw connectors ARK > The input pins of the logic part are available on the goldpin connectors - 2.54 mm pitch	€4.50

	> The board consists of all necessary for the proper operation of the controller, passive components > The system has a mounted radiator > Tile sizes: 44 x 44 x 30 mm	
DFRobot L298N - two-channel motor controller - 12V / 2A	> Voltage supply for the logical part: from 6 V to 12 V > Supply voltage for motors: 4.8 V to 46 V > Maximum current of the logical part: 36 mA > Maximum current per channel: 2 A > Maximum power: 25 W > Level of control output signals: > High state: 2.3 V = Vin = Vss > Low state: 0.3 V = Vin = 1.5 V > Operating temperature: -25°C to 130°C > Controller type: double bridge H > Dimensions: 47 x 53 mm > Weight: approx. 29 g	€13.90
Cytron MDD3A - dual-channel 16V / 3A motor controller	> Supply voltage: 4 V to 16 V > Voltage of the logical part: 1.8 V, 3.3 V, 5 V and 12 V > Current per channel: 3 A (momentary 5 A) > Number of channels: DC motor: 2 Stepper motor: 1 > Full NMOS H bridge for better performance without heat sinks > PWM speed control frequency up to 20 kHz > 4 buttons for quick system testing > Dimensions: 48 x 42 mm > Weight: 14,5 g	€8.00
Motors		
JGY-370 DC 12V Motor Reduction Gear Turbine Worm Self-locking Encoder Signal Feedback Motor - 10RPM	> Brand: Machifit Type: Reduction Motor > Material: Metal > Voltage: DC 12V > Weight: 167g > Power: 3W > Rated Voltage: 3 - 24V > Speed Regulation: Yes > Rotation In Both Directions: Yes > Encoder: Yes, hall with 2 sensors > Rated Torque: 12 kgcm > Current: 0.3 A (stall - 0.85A) > Rated Speed: 10 RPM > Size: Outer Dia.: 6mm > Motor: 46 x 32 mm (without shaft) Line Length: 230mm	€10.00

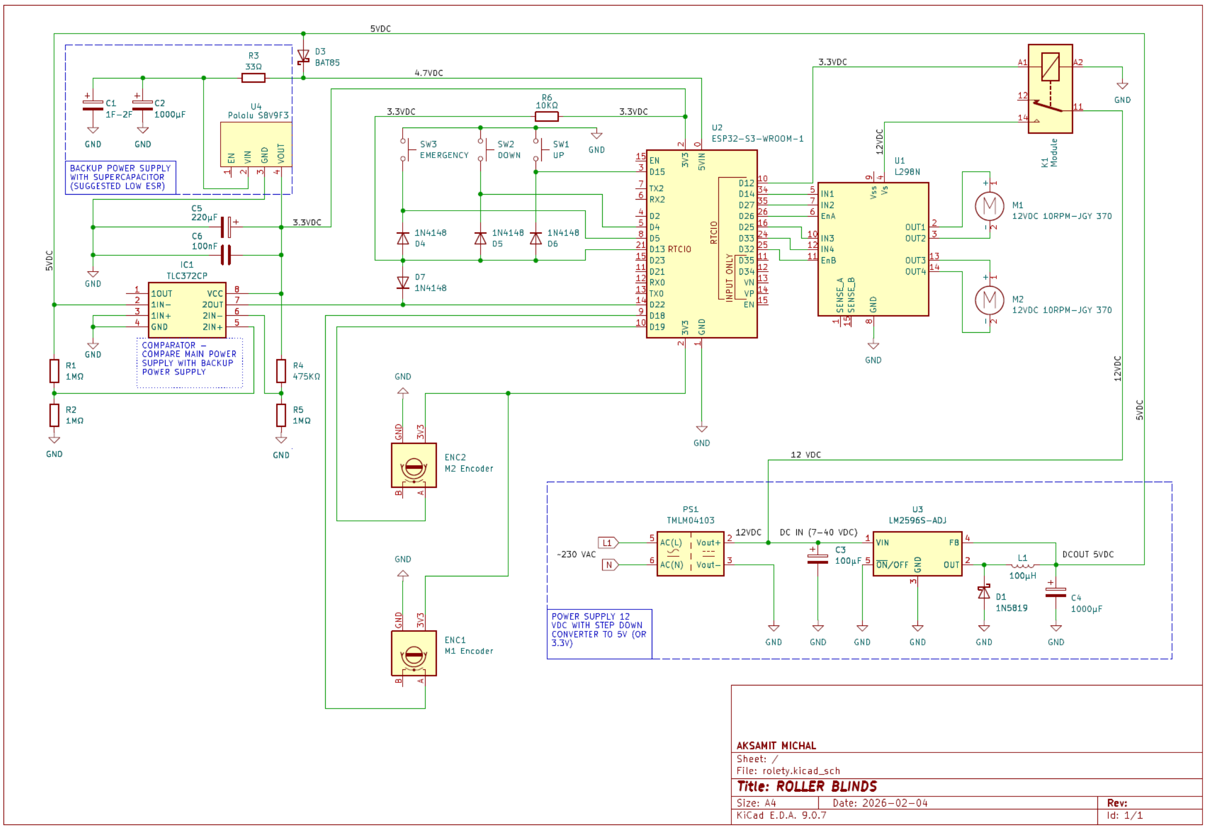
Micro DC Worm Gear Motor w/ Encoder - 12V 39RPM	> Model: 1220WG-N20VA-603-EN 12V > Gear ratio : 1/603 > No Load Current: 65 mA > No Load Speed : 39 RPM > Rated Torque : 1.8 Kg.cm > Rated Speed : 27 RPM > Rated Current : 0.15 A > Stall Current : 0.4 A > Stall Torque : > 3 Kg.cm > Encoder: hall, 7 ppr	€11.00

Table. 3.1.1

Selected items in category are marked with green glow of cell. Green marks show pros of the parts, critical cons are marked red.

4. Electronic diagram.

The purpose of this section is to describe all wiring with chosen modules from section above.



Electronic diagram 4.1

As it can be seen on above diagram (4.1) for proper installation of one device it is needed:

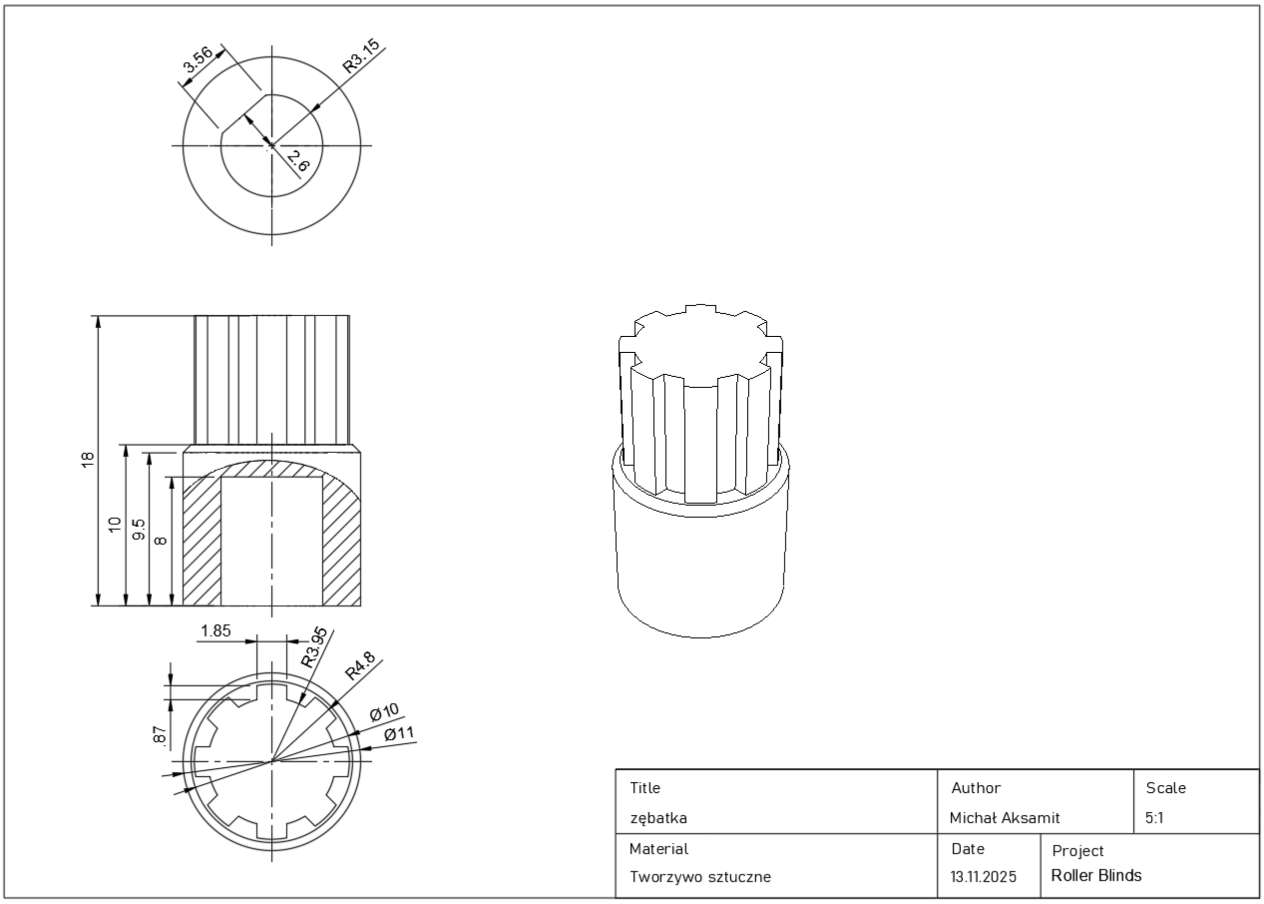
- 1 microcontroller EPS32 (U2)
- 1 motor controller L298N (U1), 1 comparator TLC372CP (IC1)
- 2 motors (M1, M2) with 2 encoders (ENC1, ENC2) installed by manufacturer
- 3 mono switches (SW1, SW2, SW3), 5 resistors, 6 capacitors, 4 diodes, 2 shottky diodes, 1 relay module with optic separation
- 1 power supply, 230 VAC to 12 VDC (PS1), 1 DC-DC converter LM2596S-ADJ (U3), 1 backup DC-DC converter Pololu S8V9F3 (U4)
- 2 wires (W1, W2) 6x0.35 mm^2 with 2,54 mm connectors (about 5 m each)
- 1 electrical box (size: 196 cm x 150cm x 30cm)

It should be notified that the device is equipped with a backup power supply. The most important element is a super capacitor (C1) which is low ESR. That capacitor is enough for 50 seconds of esp32 work during shut down and is used to save encoder data on flash. In the middle section there are 4 diodes (D4, D5, D6, D7) connected to one RTCO input of ESP32. It is used as a logical OR to wake up ESP32 from light sleep mode. The output of TLC372CP is in high state when IN+ goes to low and IN- is higher voltage than IN+ (about 2.5 V).

Electronic diagrams are available as PDF in the “doc” folder.

5. Mechanics and Construction.

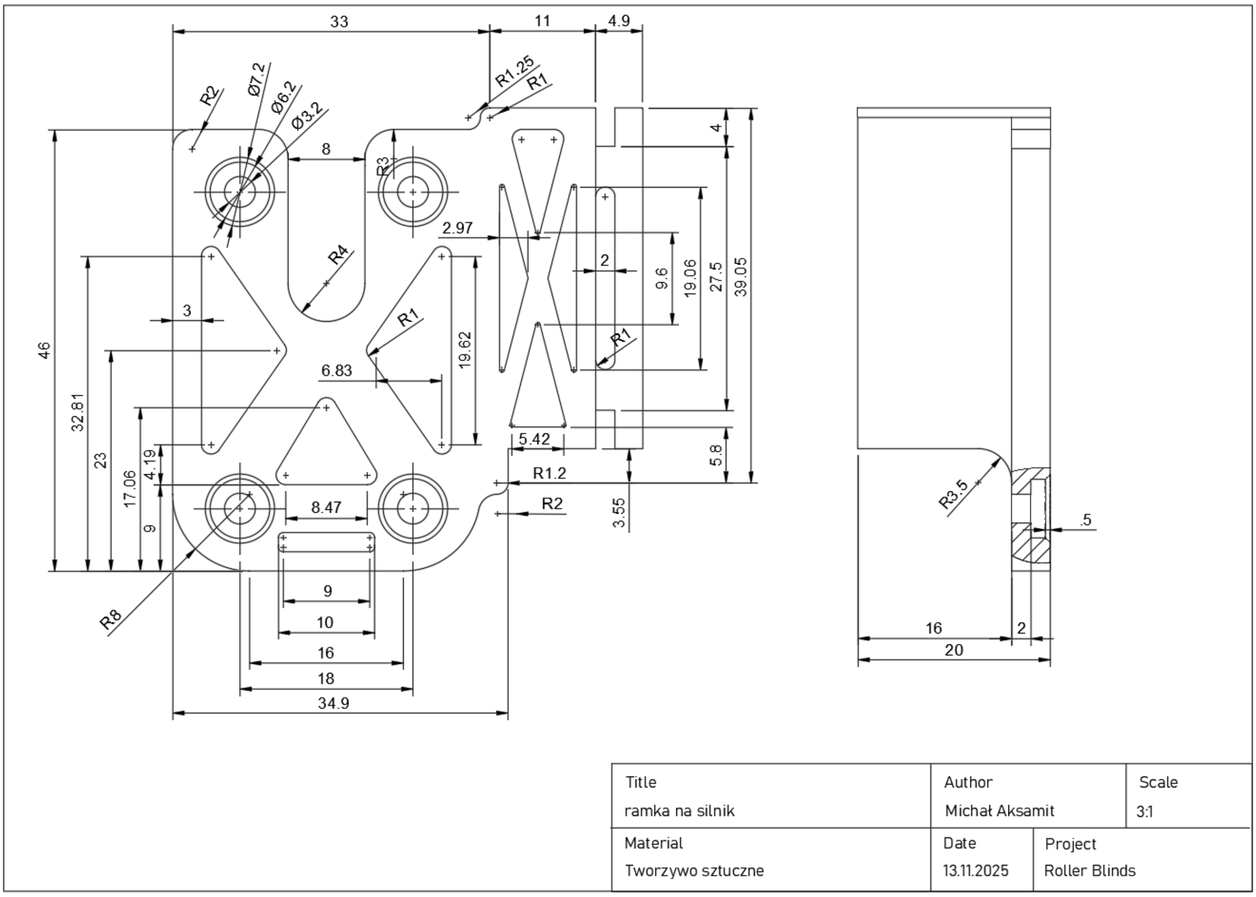
Mechanical installation of motors connected with roller blinds on windows requires specific parts to connect all of them to become one functional device. The needed parts are 3D printed with PLA filament. Technical drawings of those parts are visible below.



Title zębatka	Author Michał Aksamit	Scale 5:1
Material Tworzywo sztuczne	Date 13.11.2025	Project Roller Blinds

Technical Drawing 5.1, Gear mechanism

Gear mechanism represented in drawing above (4.1) is responsible for physical connection of motor shaft with roller blind gear.



Title ramka na silnik	Author Michał Aksamit	Scale 3:1
Material Tworzywo sztuczne	Date 13.11.2025	Project Roller Blinds

Technical Drawing 5.2, mounting frame of motor

The mounting frame of motor (4.2) connects motor with window by four m3 screws and one additional window frame (installed in roller blinds by manufacturer) which is slipped on the connector of this part (looking at the right side of main view). Because of technical holes, the part itself is light, and material consumption of filament is maximally reduced.

6. Block Diagram and Program Description.

This section is dedicated for describing program structure, work of particular tasks and architecture of system. It shows design and pattern used in project and answer question on topic such as:

- purpose of the program
- detailed description
- functionalities

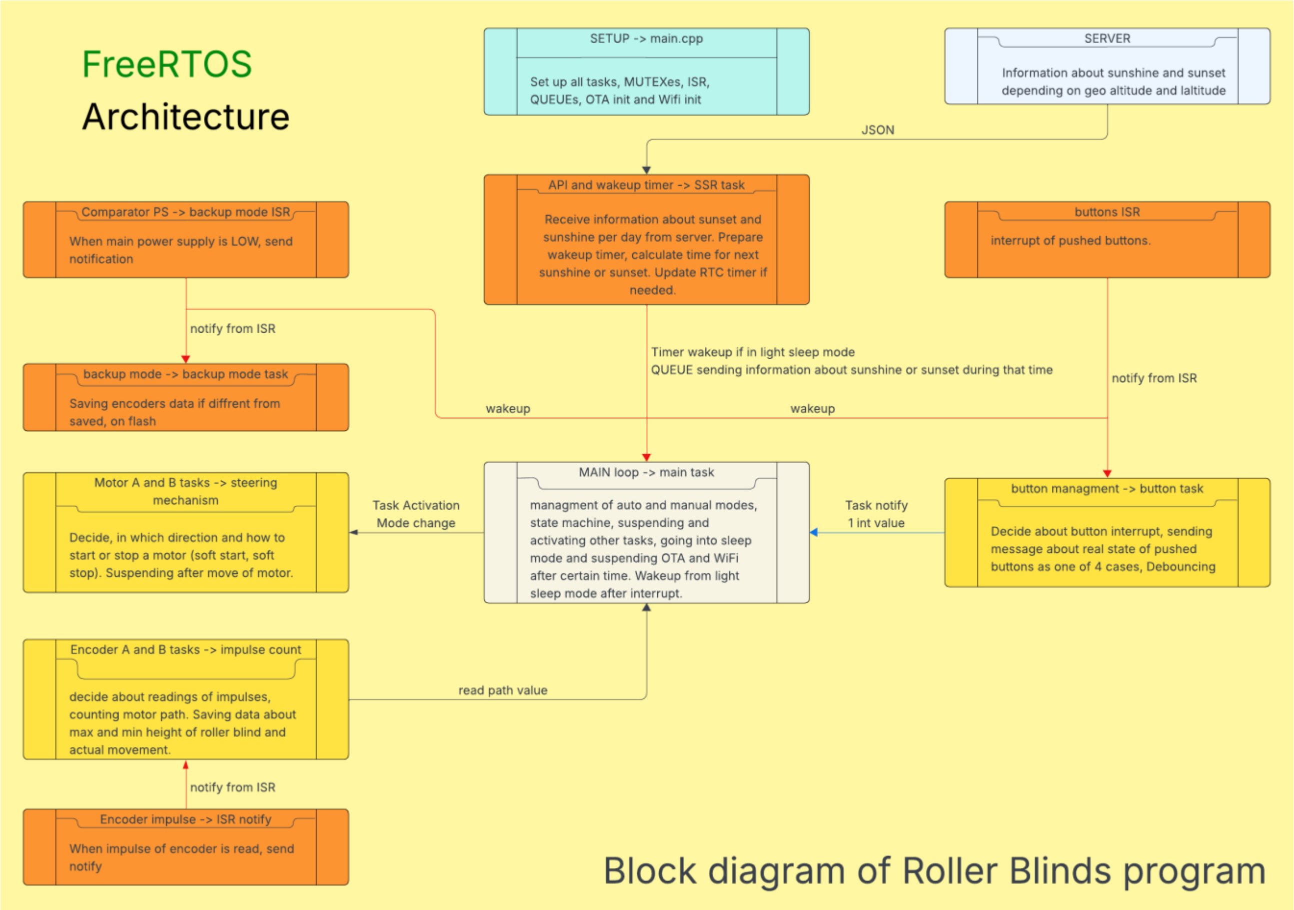

FreeRTOS
Architecture
SETUP -> main.cpp
Set up all tasks, MUTEXes, ISR, QUEUEs, OTA init and Wifi init
SERVER
Information about sunshine and sunset depending on geo altitude and laltitude
JSON
Comparator PS -> backup mode ISR
When main power supply is LOW, send notification
API and wakeup timer -> SSR task
Receive information about sunset and sunshine per day from server. Prepare wakeup timer, calculate time for next sunshine or sunset. Update RTC timer if needed.
buttons ISR
interrupt of pushed buttons.
notify from ISR
backup mode -> backup mode task
Saving encoders data if diffrent from saved, on flash
Timer wakeup if in light sleep mode
QUEUE sending information about sunshine or sunset during that time
notify from ISR
wakeup
wakeup
Motor A and B tasks -> steering mechanism
Decide, in which direction and how to start or stop a motor (soft start, soft stop). Suspending after move of motor.
Task Activation
Mode change
MAIN loop -> main task
managment of auto and manual modes, state machine, suspending and activating other tasks, going into sleep mode and suspending OTA and WiFi after certain time. Wakeup from light sleep mode after interrupt.
Task notify
1 int value
button managment -> button task
Decide about button interrupt, sending message about real state of pushed buttons as one of 4 cases, Debouncing
Encoder A and B tasks -> impulse count
decide about readings of impulses, counting motor path. Saving data about max and min height of roller blind and actual movement.
read path value
notify from ISR
Encoder impulse -> ISR notify
When impulse of encoder is read, send notify
Block diagram of Roller Blinds program

Diagram 6.1, Full structure of program.

The program itself has been written in FreeRTOS system, using c++ code.

Main purpose of project was to create deterministic and safe to use (due to moving elements as motors), fast reacting system with real two motors usage (so one controller can move two roller blinds). It has manual move mode and automatic mode. Both modes are essentially important. In short terms, users should be able to move roller blinds with two switches, first switch move motor up and second down. There is another switch which is an emergency type switch (to see electrical connections, see electronic diagram). If something wrong happens during a move or program is not responding, the user should be able to stop the machine at any time, and it should be certain to stop it. Device itself download data from servers about sunshine and sunset in the user region daily and open or close roller blinds during that time automatically. Project should save as much electrical energy as possible, so if nothing is going on, after certain time esp32 goes in light sleep mode to save power. Additionally, after cutting power off suddenly the controller enters backup mode to save encoders data in flash to remember position of motors after next power on. That solution allows return to work immediately without needing to set base position of motors after every turn on manually (to see build of backup power comparator used in project refer to electronic diagram).

Manual mode.

Manual mode is used when user wants to move roller blind manually by switches or to program maximal and minimal height in which motor stops during automatic phase. If the controller is in light sleep mode, pressing any button will wake up esp32 automatically. For manual mode, mostly button task is used, awakened by ISR, which sends case to main by task notification. The state machine cases of main task in numbers are described as below:

- -1- up switch pressed – move motor up
- -2- down switch pressed – move motor down
- -3- emergency button pressed – stop motor, hard stop. WARNING: After pressing that button, the red LED starts flashing. MOTORS cannot be moved anymore; user needs to restart device by turn off and on.
- -4- both up and down button pressed – save maximal or minimal position of roller blind. When the user presses 2 buttons, the system will signal it by flashing red LED diode. During certain time (5 seconds) users choose if the position of motor is max or min by pressing up or down button. If they don't select anything, the red diode will go off, and mode will be changed to neutral again. If the user selects a maximum pos which will be below minimum pos, min pos will be reset and program will signal it with red light. The same situation will be with min pos above max position. Users should change max pos or min if that occurs.
- -5- the button unpressed – stop motors with soft stop

If manual mode was activated, the program cannot change to automatic for 5 seconds because of safety reasons. After 5 seconds of waiting for the next press, the controller will go into light sleep mode.

Automatic mode.

Automatic mode uses mostly SSR task. That task launches the Wi-Fi module, shuts down that module in device and sends requests to the server for sunrise and sunset data. If data cannot be downloaded, then the system will try to reconnect after the next 5 seconds. If the time for sunrise or sunset has gone, then it sets timer for next day to try downloading the next data. Otherwise, the automatic mode will set a wakeup timer for the next sunset or sunrise. During Sunset or sunrise, the SSR after wakeup will send a QUEUE message for main task with number (1 - sunrise, 2 - sunset) and main will move motors, using state machine and monitoring encoders state. If the position of motors is in minimal or maximal position set by user, motor will stop and main will notify SSR task to calculate and set timer for next event. After that Esp32 will go to light sleep mode.