User Guide – PlankA16 v10.4 Point & Servo Controller

Overview

THIS IS STILL A WORK IN PROGRESS!

This Arduino sketch allows you to control **up to 16 servos** and 16 **corresponding LEDs**, with input from buttons (via a PCF8575 I²C I/O expander) and optional CMRI/RS485 communication.
It features:

- Manual servo control via buttons or optionally JMRI on various platforms
- Visual feedback via LEDs and a 20x4 I²C LCD
- Menu navigation using a rotary encoder and pushbutton
- EEPROM storage for calibration, throw speed, and settings
- Optional “centre servo” function for calibration

User-Configurable settings (set once and forget)

Constants at the top of the sketch can be safely adjusted by the user to match hardware setup. Explanation is included in the code comments

Point pairs/groups. These are 2 or more points that operate by the same switch.

3 way point. One point can be a 3 way point with 2 servos where the software will automatically prevent both being thrown at the same time. Use the lowest servo number for the threewaypoint constant in the code.

LED sequence. This can be set to suie panel layout limitations. The LEDs are assumed to have “light pipes” to the panel positions.

User-Configurable settings (On-the-fly)

Press encoder to enter main menu, rotate the encoder to desired option press encoder button to select the option

- Individual Thrown and closed point throw using encoder, buttons LCD and LEDs

- Throw speed
- Enable / disable Point pairs / groups
- Local Automation (future implementation)
- Centre Servo for servo replacement and installation.
- Memory storage of settings
- Undo setting changes option

Hardware Requirements

- **Arduino** (Pro Mini, Uno, Nano, etc.)
- **PCF8575** I/O expander for buttons
- **PCA9685** servo driver
- **Servos** (up to 16)
- **Neopixel LEDs** (up to 16)
- **Push buttons** (up to 16)
- **20x4 I²C LCD text display**
- **Rotary encoder** with pushbutton
- **Optional RS485 / CMRI module**
- Power supply: 5V regulated for servos and logic (seperate PSUs recommended)

System Operation

Startup

1. Holding in the encoder will set all servos to the centre position.
2. Arduino initializes I²C devices, PCA9685 servo driver, LCD, and PCF8575.
3. EEPROM is read to load stored servo positions and settings.
4. Servos are moved to their start of day positions.
5. LCD displays “Point control with current loaded version” and startup info.

Manual Control

- Press a button connected to PCF8575 to toggle a point (servo) between “closed” and “thrown”.
- The LED will change color:
    - **Red**: thrown
    - **Green**: closed
    - **Orange**: Paired point thrown
    - **Cyan:** Paired points closed
    - **Blue** : point moving
    - **Purple Red** : point currently being calibrated in thrown position
    - **Purple Green** : point currently being calibrated in closed position
    - **Yellow :** servo centred

Calibration

- Select “Calibrate Positions” from the menu.
- Use the encoder to fine-tune the selected servo.
- Press the encoder to save or undo the position.
- Changes are written to EEPROM for persistent storage.

LEDs

- Each point has a corresponding LED defined in `LEDS_MIMIC`.
- LED color indicates the current state of the point (see above).

EEPROM Storage

- Stores:
    - Closed and thrown positions for each servo
    - Movement speed
    - Point pairing and local automation settings
- Automatically loaded on startup.

Notes

- **Always use a separate external 5V supply for servos and LEDSs.**
- **Common ground** required between Arduino, PCA9685, PCF8575, and servos.
- **CMRI/RS485** must match JMRI / C/MRI settings if used.
- Adjust only **user-configurable constants**; other changes require code knowledge.
- Serial monitor will display version number and all connected i2c addresses on startup