**TT-02R: Transmitter + Receiver Development Documentation**

**1. System Overview**

The TT-02R RC car has been converted to use an **Arduino-based transmitter/receiver pair** with an **nRF24L01 wireless link** and a **PCA9685 PWM driver**.  
This allows one Pro Mini to control the car via I²C-driven, hardware-timed PWM signals for the ESC, steering servo, and head swivel servo.

* **Transmitter**: Arduino Pro Mini + joystick + switches, sends Data\_Package over nRF24.
* **Receiver**: Arduino Pro Mini + PCA9685, receives Data\_Package and drives ESC/servos.

**2. Hardware Configuration**

**Receiver Board Connections (PCA9685):**

* **CH0** → ESC (EzRun brushless ESC)
* **CH1** → Steering servo
* **CH2** → Head swivel servo

**nRF24L01 Radio:**

* CE → D9
* CSN → D10
* SPI pins (MISO/MOSI/SCK) → Arduino hardware SPI

**Trim Pots (optional):**

* A3 = steering trim
* A4 = throttle trim

**Power:** ESC provides BEC voltage (~5.9V) to PCA9685 V+ and receiver VCC.

**3. Software Architecture**

**Transmitter:**

* Reads joystick values for throttle (0–255) and steering (0–255).
* Sends control packet:

struct Data\_Package {

byte throttle; // 0–255

byte steering; // 0–255

byte button1;

byte button2;

byte speedMode; // 0=slow, 1=medium, 2=fast

byte headSwivel; // 0–255

};

**Receiver:**

* Reads packet from nRF24.
* Generates ESC, steering, and head servo PWM via PCA9685.
* Includes failsafe (soft stop if no signal >1s).
* Implements **double-tap reverse logic** for EZRun ESC.

**4. ESC Calibration (PCA9685 → EzRun)**

From bench and on-car testing:

| **Action** | **Pulse Width (µs)** | **PCA9685 ticks (approx)** |
| --- | --- | --- |
| Neutral | 1500 | 313 |
| Brake | 1400 | 292 |
| Reverse entry (soft) | 1425 | 298 |
| Full reverse | 1300 | 272 |
| Forward slow | 1535 | 320 |
| Forward medium | 1570 | 328 |
| Forward fast | 1600 | 336 |

**5. Reverse Logic (Double-Tap)**

The EzRun ESC requires a **double-tap sequence**:

1. **First pull below neutral** → Brake only (1400 µs).
2. **Return to neutral** (~1500 µs).
3. **Second pull below neutral** → Reverse modulation (1425 → 1300 µs).

The receiver uses a small **state machine**:

* RS\_IDLE → normal (neutral/forward).
* RS\_BRAKE → first pull, sends brake.
* RS\_WAITING\_REV → neutral detected after brake.
* RS\_REV → second pull, sends reverse throttle.

This matches stock radio behavior and prevents “reverse → stop → reverse” glitches.

**6. Speed Modes**

Throttle scaling is controlled by speedMode byte:

* **Slow Mode**: Forward capped at 1550 µs
* **Medium Mode**: Forward capped at 1570 µs
* **Fast Mode**: Forward capped at 1600 µs

Each mode spreads its usable throttle range across the full joystick movement (127–255).  
This avoids having all speed modes feel identical.

**7. Steering & Head Servo**

* **Steering Center**: ~1090 µs (measured wheels-straight position).
* Mapping: 0–255 joystick → ~±400 µs range (690–1490 µs or reversed if needed).
* **Head Swivel**: Limited 1100–1900 µs for realistic motion.