

# S.BUS Development Guide

This tutorial explains how to enable and customize **S.BUS control** on the ESP32-based driver board.

After completing these steps, your RC transmitter can directly control the robot through the **S.BUS receiver** connected to the board.

---

## 1. Enable S.BUS Mode

Open the project and edit the `Config.h` file.

Add the following line to enable S.BUS on **UART0** (default baud rate: **1 Mbps**):

```
#define UART0_AS_SBUS
```

Once enabled, UART0 will automatically act as the S.BUS input interface.

---

## 2. Modify the Example Code

In `main.cpp`, locate the `loop()` function.

You'll find an example S.BUS routine that reads RC channel values and converts them into motor control commands.

You can adjust this section to fit your transmitter's channel mapping and value range.

Example snippet:

```
#ifdef UART0_AS_SBUS
  if (sbus.Read()) {
    sbusData = sbus.data();

    // Print received channel data
    for (int8_t i = 0; i < sbusData.NUM_CH; i++) {
      Serial.print(sbusData.ch[i]);
      Serial.print("\t");
    }
  }
}
```

```

}
Serial.print(sbusData.lost_frame);
Serial.print("\t");
Serial.println(sbusData.failsafe);

// Define speed scaling based on CH5 (mode switch)
float speed_limit = 1.0;
if (sbusData.ch[4] < 300) {
    speed_limit = 0.33;
} else if (sbusData.ch[4] == 1002) {
    speed_limit = 0.66;
} else if (sbusData.ch[4] > 1700) {
    speed_limit = 1.0;
}

// Convert joystick input to differential drive commands
float speed_input = constrain(float(sbusData.ch[2] - SBUS_MID)/SBUS_RANGE, -1.0, 1.0);
float turn_input = -constrain(float(sbusData.ch[3] - SBUS_MID)/SBUS_RANGE, -1.0, 1.0);

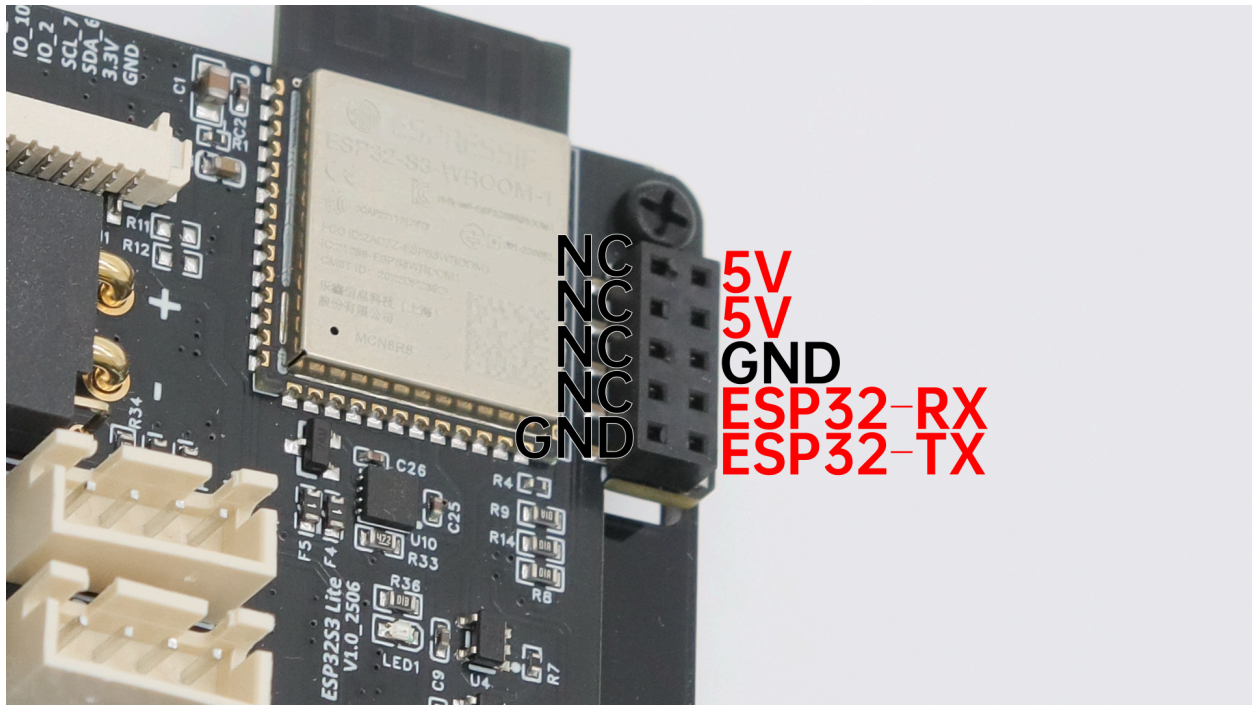
float left_speed = (speed_input) * speed_limit * 6000.0 - (turn_input * 0.33 * 6000.0);
float right_speed = (speed_input) * speed_limit * 6000.0 + (turn_input * 0.33 * 6000.0);

jointsCtrl.hubMotorCtrl(left_speed, right_speed, right_speed, left_speed);
}
#endif

```

After making changes, **build and flash the firmware** to the driver board.

### 3. Hardware Connection



Connect your **RC receiver** to the driver board as follows:

Receiver Pin	Driver Board Pin	Description
<b>+5V</b>	<b>5V</b>	Power supply (up to <b>5A</b> )
<b>GND</b>	<b>GND</b>	Ground
<b>S.BUS Signal</b>	<b>ESP32 RX (UART0)</b>	S.BUS data input

⚠ Important: Ensure the receiver operates at 5V logic.

Connecting higher-voltage signals directly to ESP32 RX may cause damage.

## 4. Customizing Control Logic

You can freely adjust:

- **Channel mapping** (e.g., throttle, steering, mode switch)
- **Speed scaling** and **mixing ratio**
- **Failsafe behavior** or **loss-frame handling**

This allows you to adapt the control scheme to any **RC transmitter** or **custom robotic chassis**.

---

## Summary

Once configured, the ESP32 board can:

- Receive up to **16 S.BUS channels** at 1 Mbps
- Map RC input to motion control in real-time
- Provide stable 5V/5A power for the receiver
- Support full customization through open-source firmware