ByteRider

Version

Table of Contents

ByteRider documentation

Add your content using `reStructuredText` syntax. See the reStructuredText documentation for details.

INTRODUCTION

HOW DOES IT WORK?

The BitByteRider RC car is powered by ESP32-C3 Breadboard & Power adapter developmemt board.

Reserved Pins & GPIOs

The following table summarizes GPIOs and pins reserved for operations purposes.

The GPIO numbers correspond to those on the ESP32-C3 WROOM microcontroller. The Pin number corresponds to the pin on the Breadboard and Power adapter development board.

x- and y- axis

The **GPIO0** and **GPIO1** assigned to measuring the voltage of x- and y- axis of the Joystick. Lastly, there is a group of GPIO pairs responsible for PWM for DC motors.

Direction and Speed

The pairs of DC motors on the left side are wired to the dedicated PWM channels. This means that *ESP32-C3 Breadboard DevBoard* can control rotation speed and direction of DC motors in pairs only (i.e. left and right side). Consequently, only four PWM channels are used for controlling the direction of the RC car. Based on this constraint, the RC car can only move front, back, and turn/ rotate left and right. Any other movements are not possible (i.e. diagonal or sideways).

A pair of PWM channels are required for defining rotation speed and direction of the DC motors on each side. In particular, **GPIO6** and **GPIO5** provide PWM to the left- and right- side DC motors to rotate in a **clockwise** direction. Similarly, **GPIO4** and **GPIO7** provide PWM to the left- and right- side DC motors to rotate in a **counter-clockwise** direction. Changing PWM on each channel determines the speed and direction of the RC car.

The following images illustrate various PWM duty cycles recorded by oscilloscope.

GPIO	Pin	Function	Notes
0	16	Joystick x-axis	ADC1_CH0
1	15	Joystick y-axis	ADC1_CH1
8	5	Joystick push button	
6	4	PWM for clockwise rotation of left-side motors	LEDC_CHANNEL_1
5	3	PWM for clockwise rotation of right-side motors	LEDC_CHANNEL_0
4	2	PWM for counter-clockwise rotation of right-side motors	LEDC_CHANNEL_2
7	6	PWM for counter-clockwise rotation of left-side motors	LEDC_CHANNEL_3

Fusion of Software with Hardware

The *struct* for storing motors PWM values.

```
struct motors_rpm {
    int motor1_rpm_pwm;
    int motor2_rpm_pwm;
    int motor3_rpm_pwm;
    int motor4_rpm_pwm;
};
```

The function for updating motors' PWM values.

```
// Function to send data to the receiver
void sendData (void) {
    sensors_data_t buffer;              // Declare data struct

    buffer.crc = 0;
    buffer.x_axis = 0;
    buffer.y_axis = 0;
    buffer.nav_bttn = 0;
```

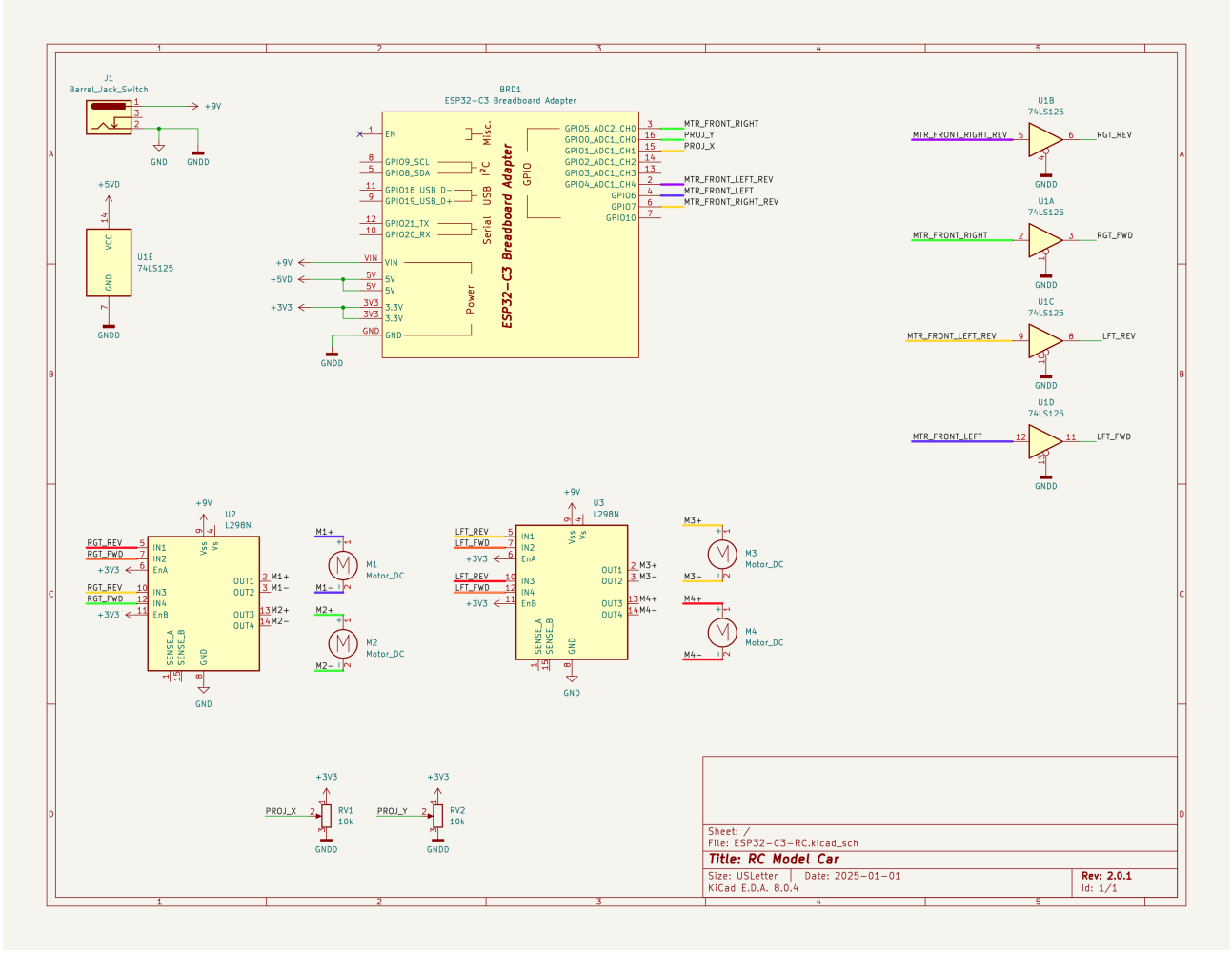
```
    buffer.motor1_rpm_pwm = 0;
    buffer.motor2_rpm_pwm = 0;
    buffer.motor3_rpm_pwm = 0;
    buffer.motor4_rpm_pwm = 0;

    // Display brief summary of data being sent.
    ESP_LOGI(TAG, "Joystick (x,y) position ( 0x%04X, 0x%04X )",
(uint8_t)buffer.x_axis, (uint8_t)buffer.y_axis);
    ESP_LOGI(TAG, "pwm 1, pwm 2 [ 0x%04X, 0x%04X ]",
(uint8_t)buffer.pwm, (uint8_t)buffer.pwm);
    ESP_LOGI(TAG, "pwm 3, pwm 4 [ 0x%04X, 0x%04X ]",
(uint8_t)buffer.pwm, (uint8_t)buffer.pwm);

    // Call ESP-NOW function to send data (MAC address of receiver,
pointer to the memory holding data & data length)
    uint8_t result = esp_now_send(receiver_mac, &buffer,
sizeof(buffer));

    // If status is NOT OK, display error message and error code (in
hexadecimal).
    if (result != 0) {
        ESP_LOGE("ESP-NOW", "Error sending data! Error code:
0x%04X", result);
        deletePeer();
    }
    else
        ESP_LOGW("ESP-NOW", "Data was sent.");
}
```

The onDataReceived() and onDataSent() are two call-bacl functions that get evoked on each corresponding event.

The rc_send_data_task() function runs every 0.1 second to transmit the data to the receiver.

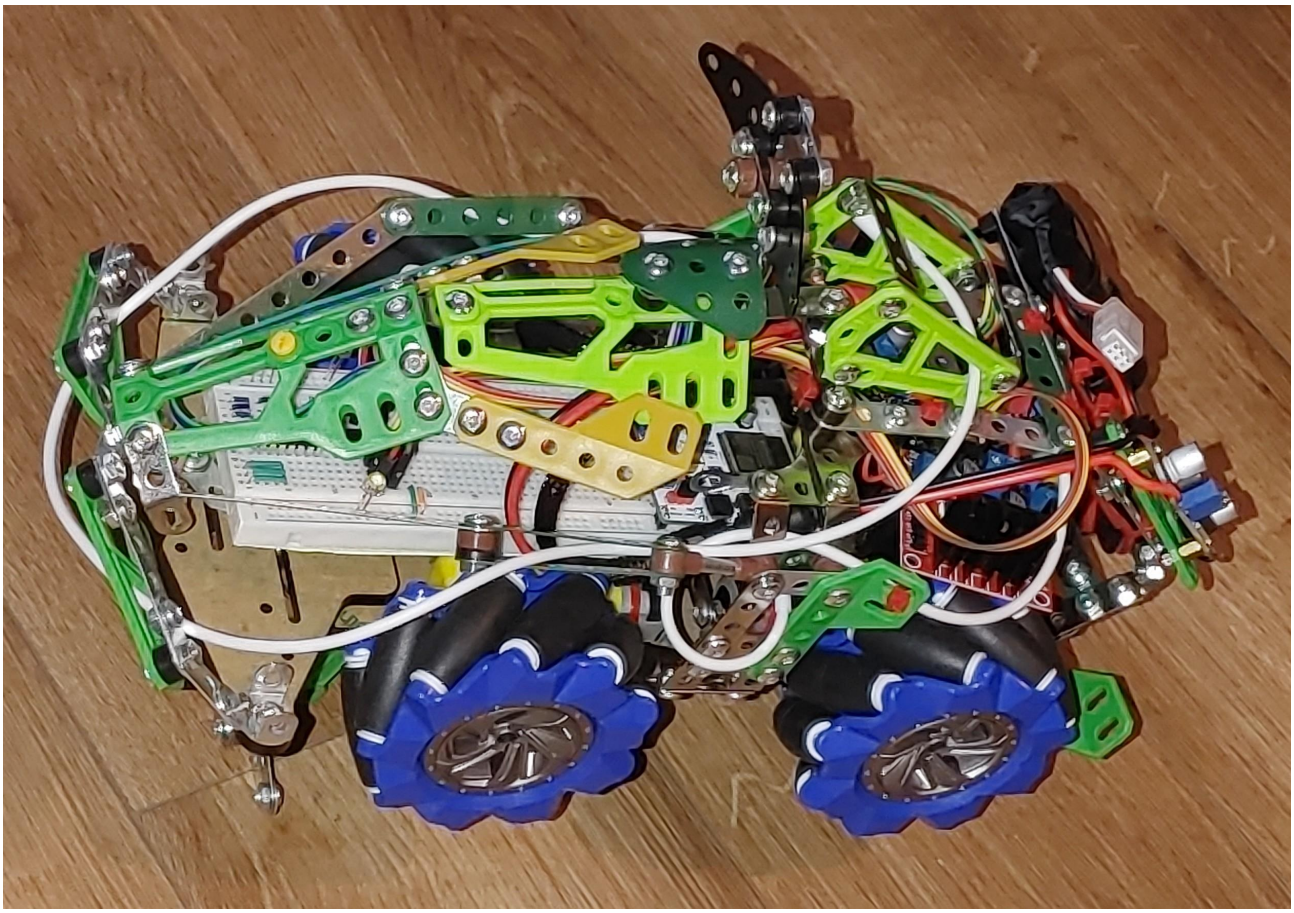
```
// Continous, periodic task that sends data.
static void rc_send_data_task (void *arg) {

    while (true) {
        if (esp_now_is_peer_exist(receiver_mac))
            sendData();
        vTaskDelay (100 / portTICK_PERIOD_MS);
    }
}
```

Schematic

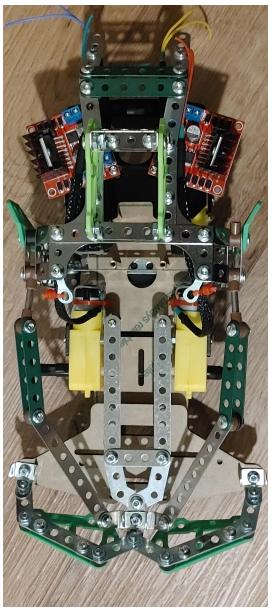

J1
Barrel_Jack_Switch
+9V
GND
GNDD
BRD1
ESP32-C3 Breadboard Adapter
ESP32-C3 Breadboard Adapter
Misc.
I²C
Serial
USB
GPIO
Power
EN
GPIO9_SCL
GPIO8_SDA
GPIO18_USB_D-
GPIO19_USB_D+
GPIO21_TX
GPIO20_RX
VIN
5V
3.3V
GPIO5_ADC2_CH0
GPIO0_ADC1_CH0
GPIO1_ADC1_CH1
GPIO2_ADC1_CH2
GPIO3_ADC1_CH3
GPIO4_ADC1_CH4
GPIO6
GPIO7
GPIO10
MTR_FRONT_RIGHT
PROJ_Y
PROJ_X
MTR_FRONT_LEFT_REV
MTR_FRONT_LEFT
MTR_FRONT_RIGHT_REV
U1E
74LS125
U1B
74LS125
U1A
74LS125
U1C
74LS125
U1D
74LS125
RGT_REV
RGT_FWD
LFT_REV
LFT_FWD
U2
L298N
U3
L298N
IN1
IN2
EnA
IN3
IN4
EnB
OUT1
OUT2
OUT3
OUT4
Vss
Vs
SENSE_A
SENSE_B
M1
Motor_DC
M2
Motor_DC
M3
Motor_DC
M4
Motor_DC
RV1
10k
RV2
10k
PROJ_X
PROJ_Y
Sheet: /
File: ESP32-C3-RC.kicad_sch
Title: RC Model Car
Size: USLetter
Date: 2025-01-01
Rev: 2.0.1
KiCad E.D.A. 8.0.4
Id: 1/1

WORK-IN-PROGRESS WALK THROUGH

Finished Work

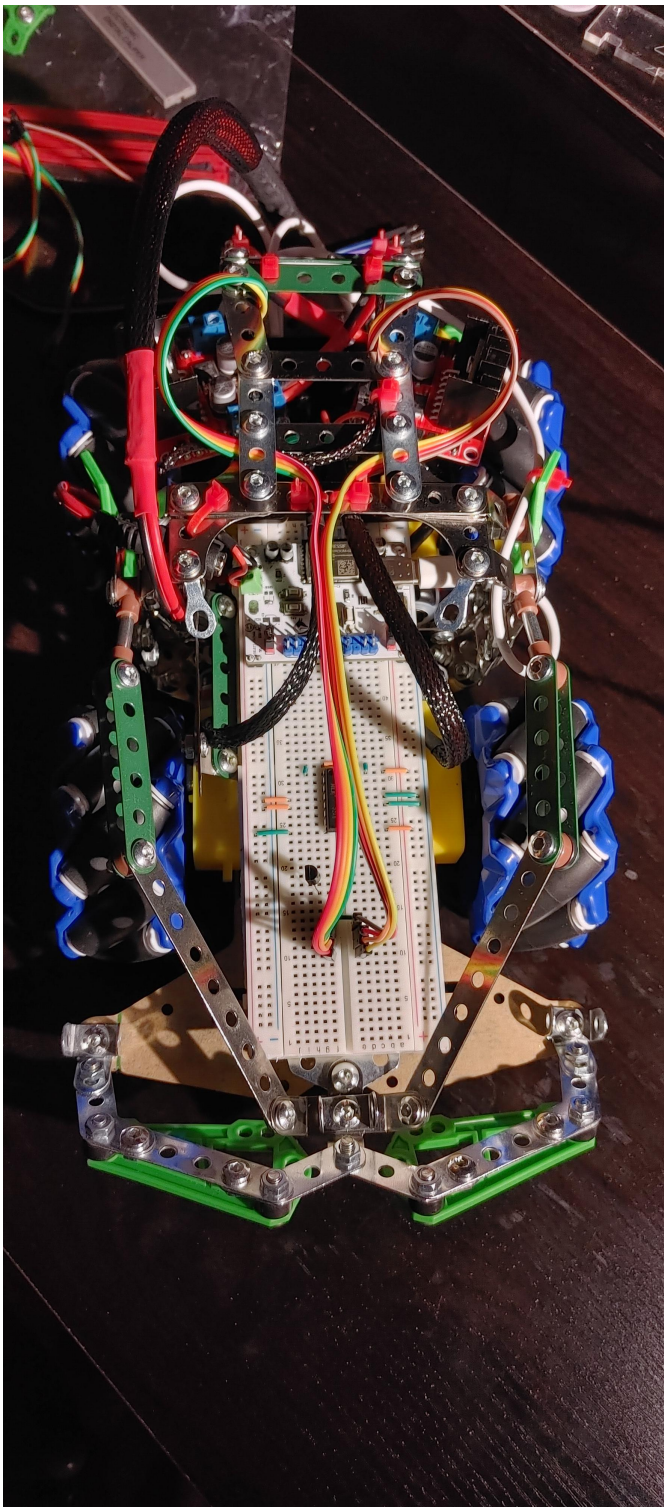


Chassis



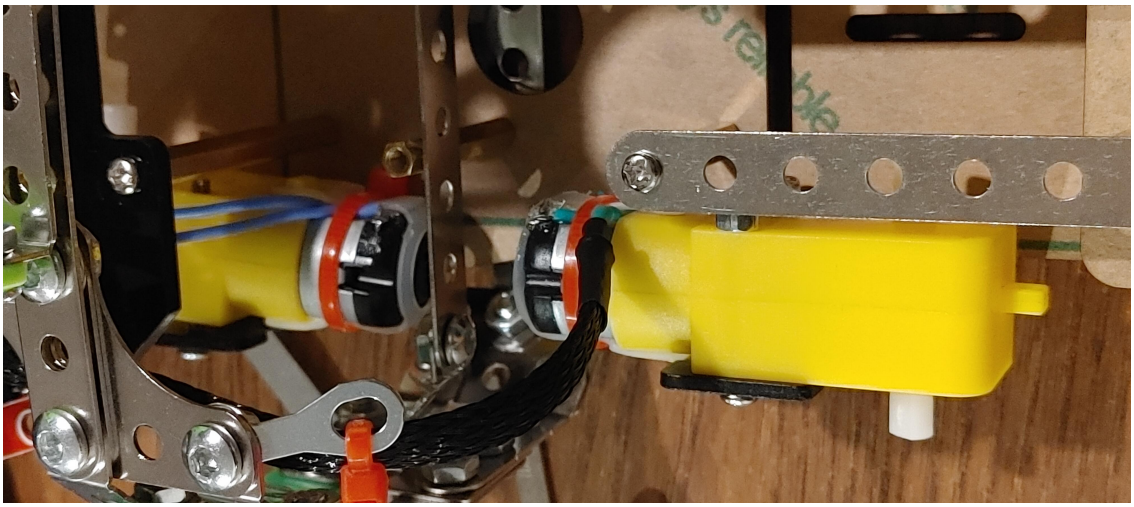
Completed chassis with only DC motor controllers installed.

Wiring



Completed wiring.

Motor Wires Harness



DC Motors wires secured inside harnes.