Romeo BLE Quad Robot Controller SKU: DFR0398

Introduction

Romeo BLE Quad is an arduino compatible robot controller based on STM32 ARM chip. It inherits all features from the Bluno M3, including wireless programming support, wireless communication between iOS/Android apps and remote control. Beyond this it also includes a 4-way DC motor driver and encoder interfaces. You can implement a robot with PID closed-loop feedback control directly with our TT Geared Motor.

As well as this the Romeo BLE Quad offers powerful performance thanks to the STM32 ARM 32-bit microcontroller with more storage space and more interface resources. DFRobot has developed bespoke firmware to make it compatible with Arduino IDE and accessible to beginners to robotics.

⚠️ Note: The operating voltage of Romeo BLE Quad is 3.3V, please read the Board Overview carefully before usage!

Features

- Supports Arduino IDE/Arduino C
- Supports Bluetooth Wireless Communications/Programming
- Supports Android and iOS applications, open source code, suitable for secondary development by the user
- Supports AT commands to configure BLE
- Upgradable firmware
- 4WD Support

Specification

- Microcontroller: STM32 F103RET6
- Clock Speed: 72 MHZ
- Bluetooth Chip: TI CC2540 (Bluetooth 4.0)
- Communication Range: 30m
- Operating Voltage: 3.3V
- DC Supply: USB Powered or External 7V~10V DC
- DC Motor: 4 way
- Supports USB and external power supply automatic switching
- Digital I/O Pins: 10
- Analog I/O Pins: 5
- I2C/IWC: 30 (Default SDA), 29 (Default SCL)
- SRAM: 64k
- Flash: 512K (Customizable Flash assignment, refer to Flash chapter for more info)
  Default User Code repository: 492K
  Default User Data repository: 20KA
- Serial Port: 2 (Serial1, Serial3)
  Serial1 0 (Rx1) and 1 (Tx1)
  Serial3 30 (Rx3), and 29 (Tx3)
- Size: 67 x 42 (mm)
- Weight: 54g

> **NOTE**: Romeo BLE Quad serial port starts from **Serial1**, it is in charge of USB & Bluetooth communication. You need to change **Serial** to '''Serial1''' in the sketch if you want to use serial monitor.

Board Overview
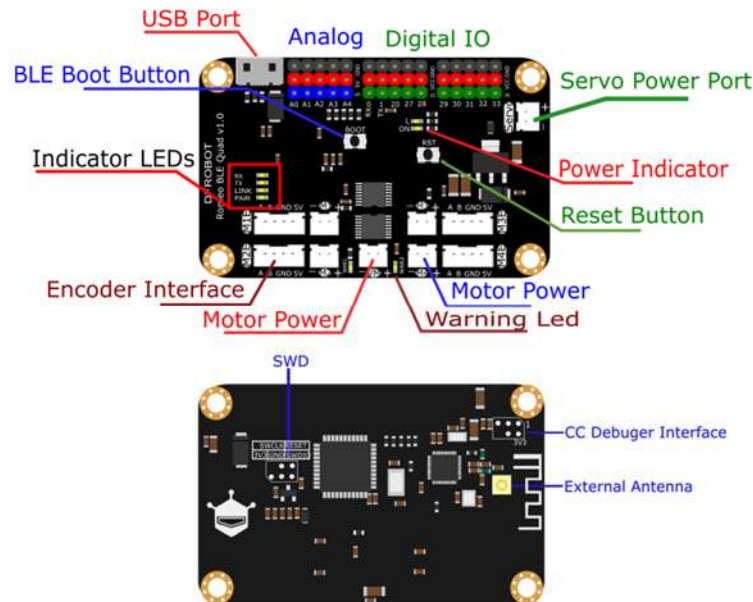
Board Overview

Motor	GPIO 1	GPIO 2	Encoder A	Encoder B
M1	8	23	12	11
M2	7	9	2	3
M3	24	14	5	26
M4	4	25	35	36

Note: Romeo BLE Quad integrates 2x HR8833 motor driver, the driving method is a little different to the common L298 motor controller, you can check the detail here: Dual 1.5A Motor Driver - HR8833 SKU: DRI0040
Recommend fast decay mode. (Encapsulated in the library)

Forward: GPIO 1 = Low (Direction); GPIO 2 = PWM (Speed)
Reverse: GPIO 2 = Low (Direction): GPIO 1 = PWM (Speed)

Special Attention: The operating voltage of Romeo BLE Quad is 3.3V, only some of pins support 5V input, please read the following instruction carefully before usage! Or it will destroy the micro chip.

3.3V only: D20, D27, D28
5VCompatible : D0, D1, D29, D30, D31, D32, D33
Analog Input Pins, A0~A4, have 5V bleeder circuit, "0~5V" will be mapped to "0~1023" analog value.

Tutorial

We need to install Romeo BLE Quad development environment before usage. Since this is a secondary development edition based on Bluno M3, and all pins function are same to Bluno M3, we can refer to Bluno M3 wiki for the environment installation.
Bluno M3 V2.2 Software Development Environment\Install Software Development Environment

PID Speed Control

In this tutorial, we'll use TT Geared Motor with Encoder. There are some important parameters:

- Sampling period: **setSampleTime** (in this case: 100 ms, 0.1s)
- The number of quadrature encoders in the sampling period **motorSpeed** (In this case: 200)
- Motor encoder pole number (in this case: 16)
- Motor reduction ratio (In this case: 120:1)

> E.g. When the motor encoder pole number is 16, the encoder increments will be 16 * 2 = 32 in one circle. If you get 200 pulse change in a sampling period, it means the real speed of motor is 200÷0.1(s)÷32 = 62.5 r/s = 62.5*60 r/min =3750 r/min; and its output speed is 3750/120=31.25r/min

- Most of function has been encapsulated in the library, you can modify the output content in "**Motor.cpp**", line 241.

```cpp
230     void Motor::calibrate(void)
231     {
232
233         int wheelSpeed;
234         wheelSpeed = getCounter()-30000;
235         input = (double)abs(wheelSpeed);
236         result=pid->Compute();
237         if(result){
238             setCounter(30000);
239             distance += wheelSpeed;
240             PWMSet(output);
241             Serial1.println(wheelSpeed);
242         }
243     }
```

Requirements

- **Hardware**
  Romeo BLE Quad x1
  TT Geared motor with Encoder
  M-M/F-M/F-F Jumper wires

- **Software**
  Arduino IDE 1.5.5 Bluno M3 version Download on Bluno M3 wiki page
  https://www.dfrobot.com/wiki/index.php/Bluno_M3_SKU:DFR0329#Setup_Software_Development_Environment

Connection Diagram
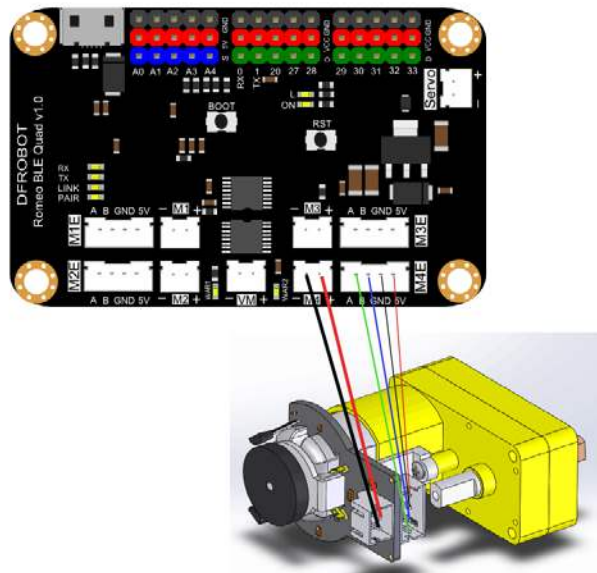
Sample Code

Click to download arduino library: Motor.h arduino library and PID_V1.h Arduino library.
How to install Libraries in Arduino IDE

```
/*!
* @file RemeoBLEQuadDrive.ino
* @brief RemeoBLEQuadDrive.ino PID control system of DC motor
*
*  RemeoBLEQuadDrive.ino Use PID control 4 way DC motor direction an
d speed
*
* @author linfeng(490289303@qq.com)
* @version  V1.0
* @date  2016-4-14
*/


#include "PID_v1.h"
#include "Motor.h"


Motor motor[4];
int motorSpeed[4] = {-200,200,400,-400};/*Set 4 speed motor*/
/* Speed=motorSpeed/(32*(setSampleTime/1000))(r/s) */
const int motorDirPin[4][2] = { //Forward, Backward
/*Motor-driven IO ports*/
  {8,23},
  {7,9},
  {24,14},
  {4,25}
};



//const double motorPidParam[3]={0.6,1,0.03};/*DC MOTOR,Yellow??180d
egree*/
```

```cpp
//const double motorPidParam[3]={1.5,1,0.05};/*DC MOTOR,Yellow??90 d
egree*/

const double motorPidParam[3]={1.2,0.8,0.05};/*Encoder V1.0,160rd/mi
n ;19500/min; 32:1,Kr=3.5*/

void setup( void )

{

  Serial1.begin(115200);

     for(int i=0;i<4;i++){

             motor[i].setPid(motorPidParam[0],motorPidParam[1],motor
PidParam[2]);/*Tuning PID parameters*/

             motor[i].setPin(motorDirPin[i][0],motorDirPin[i][1]);/*
Configure IO ports*/

             motor[i].setSampleTime(100);/*Sets the sampling period*
/

               motor[i].setChannel(i);/*Sets the motor channel */

             motor[i].ready();/*Motor enable*/

               motor[i].setSpeed(motorSpeed[i]);/*Set motor speed*/

     }

}


void loop( void )

{

     for(int i = 0; i < 4; i++){

             motor[i].calibrate();/*motor PID calibrate*/

     }


}
/******************************************************************
***********

  Copyright (C) <2016>  <linfeng>

  This program is free software: you can redistribute it and/or modi
fy

  it under the terms of the GNU General Public License as published
by

  the Free Software Foundation, either version 3 of the License, or

  (at your option) any later version.

  This program is distributed in the hope that it will be useful,

  but WITHOUT ANY WARRANTY; without even the implied warranty of
```

```
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the

    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License

    along with this program.  If not, see <http://www.gnu.org/licenses
/>.

    Contact: 490289303@qq.com

   ********************************************************************
***********/
```

Expected Results

Motor speed runs with expected speed.