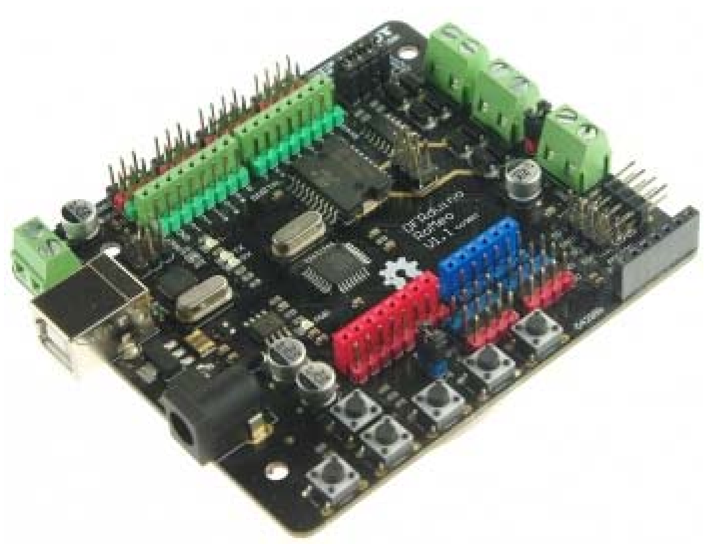
DFRduino Romeo-All in one Controller V1.1(SKU:DFR0004)



DFRduino RoMeo V1.1

Contents

Introduction

RoMeo is an All-in-One microcontroller especially designed for robotics application. Benefit from Arduino open source platform, it is supported by thousands of open source codes, and can be easily expanded with most Arduino Shields. The integrated 2 way DC motor driver and wireless socket gives a much easier way to start your robotic project.

```
Note:

A. Please read this manual carefully before applying power on the device.

B. Do not use this device for military or medical purpose as they are not des
igned to.
```

Specification

- Atmega 168/328
- 14 Channels Digital I/O
- 6 PWM Channels (Pin11,Pin10,Pin9,Pin6,Pin5,Pin3)
- 8 Channels 10-bit Analog I/O
- USB interface
- Auto sensing/switching power input
- ICSP header for direct program download
- Serial Interface TTL Level
- Support AREF
- Support Male and Female Pin Header
- Integrated sockets for APC220 RF Module and DF-Bluetooth Module
- Five I2C Interface Pin Sets
- Two way Motor Drive with 2A maximum current
- 5 key inputs
- DC Supply：USB Powered or External 7V~12V DC。
- DC Output：5V /3.3V DC and External Power Output
- Dimension：90x80mm

DFRduino RoMeo Pinout

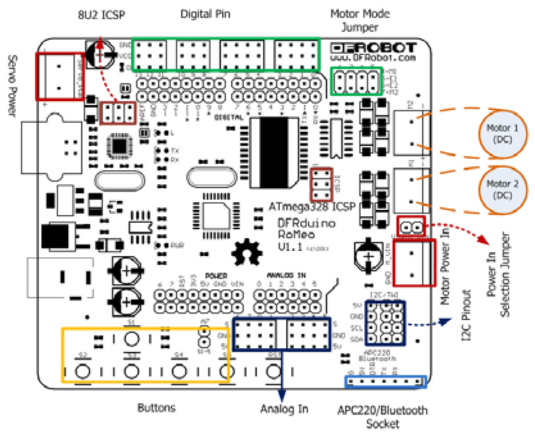


Fig1: Romeo Pin Out

The picture above shows all of the I/O lines and Connectors on the Romeo, which includes:

- One Regulated Motor Power Input Terminal (6v to12v)
- One Unregulated Servo Power Input Terminal (you supply regulated 4v to 7.2v)
- One Servo input power selection jumper
- One Serial Interface Module Header for APC220/Bluetooth Module
- Two DC Motor Terminals – Handles motor current draw up to 2A, each terminal
- One I2C/TWI Port – SDA, SCL, 5V, GND
- One Analog Port with 8 analog inputs – Analog input 7 will be occupied when connecting "A7" jumper
- One General Purpose I/O Port with 13 I/O lines – 4,5,6,7 can be used to control motors
- One Reset Button
- Jumper bank to Enable/Disable Motor Control

Before you start

Applying Power

This is one of the most important steps in getting the Romeo up and communicating with your host controller. You MUST make sure that you apply power to the Power Terminal using the correct polarity. Reverse Polarity will damage the Romeo. *We are not responsible for such damage, nor do we warranty against such damage.* Make sure you take time to apply power correctly. Otherwise, it could get costly for you!

Power from USB: Simply plug USB cable, and the Romeo is able to work. Please notice that the USB can only supply 500 mA current. It should be able to meet the most requirements for LED lit application. However it is not enough to power DC motors or servo.

Power from Motor Power Input: Simply connect the ground wire from your supply to the screw terminal labeled “GND”, and then connect the positive wire from your supply to the screw terminal labeled “VIN".

NOTE: Maximum supply voltage cannot exceed 14V DC.

Software

RoMeo can be programmed by Arduino IDE 0022 and above. It can be downloaded at Arduino.cc, Please select “Arduino UNO” as the hardware.

Romeo Configuration

Servo Power Select Jumper

As most servos draw more current than the USB power source can supply. A separate servo power terminal is provided to power the servo individually. This option can be Enabled/Disabled by the Servo Power Select Jumper.

When the Servo Power Select Jumper is applied, the servo is powered by internal 5V.

When the Servo Power Select Jumper is not applied, the servo is powered by external power source.

The Romeo V1.0 uses an automatic switcher for the power source selection. When the external power source has been applied, the servo will be automatically powered by the external power instead of USB power.

Motor Control Pin Jumper

Applying the Motor Control Pin Jumpers will allocate Pin 5,6,7,8 for motor control.

Removing the jumpers will release the above Pins, and the motor controller will be disabled.

Tutorial

Button Press

RoMeo has 5 build in buttons S1-S5 (Figure 2). S1-S5 use analog input 7,

"Button Pin Map"

Pin	Function
Analog Pin 7	Button S1-S5

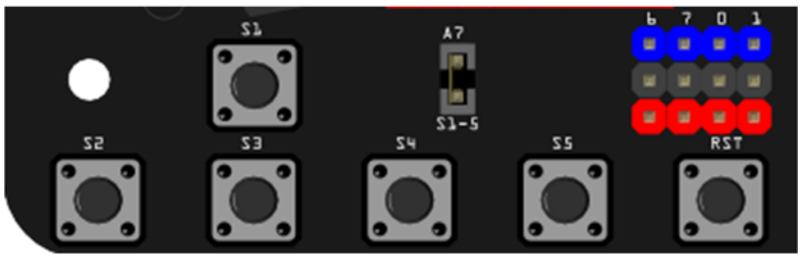


Fig2: Romeo Buttons

Example use of Button 1-5

```
char msgs[5][15] = {
  "Right Key OK ",
  "Up Key OK    ",
  "Down Key OK  ",
  "Left Key OK  ",
  "Select Key OK" };
```

```
char start_msg[15] = {
  "Start loop "};
int  adc_key_val[5] ={
  30, 150, 360, 535, 760 };
int NUM_KEYS = 5;
int adc_key_in;
int key=-1;
int oldkey=-1;
void setup() {
  pinMode(13, OUTPUT);  //we'll use the debug LED to output a heartbeat
  Serial.begin(9600);

  /* Print that we made it here */
  Serial.println(start_msg);
}

void loop()
{
  adc_key_in = analogRead(7);    // read the value from the sensor
  digitalWrite(13, HIGH);
  /* get the key */
  key = get_key(adc_key_in);    // convert into key press
  if (key != oldkey) {   // if keypress is detected
    delay(50);      // wait for debounce time
    adc_key_in = analogRead(7);    // read the value from the sensor
    key = get_key(adc_key_in);    // convert into key press
    if (key != oldkey) {
      oldkey = key;
      if (key >=0){
        Serial.println(adc_key_in, DEC);
        Serial.println(msgs[key]);
      }
    }
  }
  digitalWrite(13, LOW);
}
// Convert ADC value to key number
int get_key(unsigned int input)
{
  int k;
  for (k = 0; k < NUM_KEYS; k++)
  {
    if (input < adc_key_val[k])
    {
      return k;
    }
  }
  if (k >= NUM_KEYS)
    k = -1;     // No valid key pressed
  return k;
}
```

Dual DC Motor Speed Control

Hardware Setting

Connect four motor wires to Motor Terminal. And apply power through motor power terminal (Figure 4).

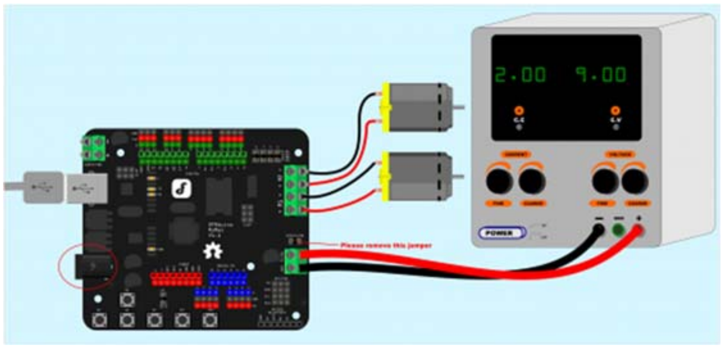


Fig3: Romeo Motor Connection Diagram

Pin Allocation

"PWM Mode"

Pin	Function
Digital 4	Motor 1 Direction control
Digital 5	Motor 1 PWM control
Digital 6	Motor 2 PWM control
Digital 7	Motor 2 Direction control

"PLL Mode"

Pin	Function
Digital 4	Motor 1 Enable control
Digital 5	Motor 1 Direction control
Digital 6	Motor 2 Direction control
Digital 7	Motor 2 Enable control

PWM Control Mode

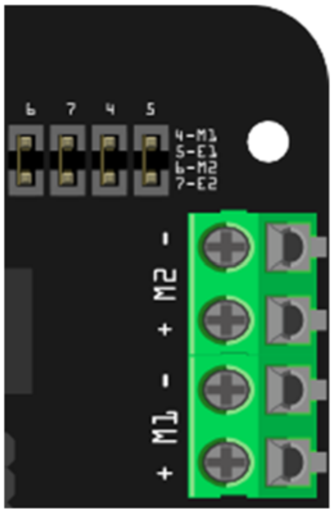


Fig4: PWM Motor Control Pin Allocation

The PWM DC motor control is implemented by manipulating two digital IO pins and two PWM pins. As illustrated in the diagram above (Figure 5), Pin 4,7 (7,8 for old Romeo version) are motor direction control pins, Pin 5,6 (6,9 for old Romeo version) are motor speed control pins.

For previous Romeo board, the pins used to control the motor is Pin 7,8 (Direction), Pin 6,9 (Speed). You can find the information at the right side of the Motor Control Pin Jumpers.

Sample Code：

```
//Standard PWM DC control
int E1 = 5;     //M1 Speed Control
int E2 = 6;     //M2 Speed Control
int M1 = 4;    //M1 Direction Control
int M2 = 7;    //M1 Direction Control

///For previous Romeo, please use these pins.
//int E1 = 6;     //M1 Speed Control
//int E2 = 9;     //M2 Speed Control
//int M1 = 7;    //M1 Direction Control
//int M2 = 8;    //M1 Direction Control



void stop(void)                    //Stop

{
  digitalWrite(E1,LOW);
  digitalWrite(E2,LOW);
}
void advance(char a,char b)          //Move forward
{
  analogWrite (E1,a);      //PWM Speed Control
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}
void back_off (char a,char b)          //Move backward
```

```
{
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}
void turn_L (char a,char b)             //Turn Left
{
  analogWrite (E1,a);
  digitalWrite(M1,LOW);
  analogWrite (E2,b);
  digitalWrite(M2,HIGH);
}
void turn_R (char a,char b)             //Turn Right
{
  analogWrite (E1,a);
  digitalWrite(M1,HIGH);
  analogWrite (E2,b);
  digitalWrite(M2,LOW);
}
```

```
void setup(void)
```

```
{
  int i;
  for(i=4;i<=7;i++)
    pinMode(i, OUTPUT);
  Serial.begin(19200);      //Set Baud Rate
  Serial.println("Run keyboard control");
}
void loop(void)
{
  if(Serial.available()){
    char val = Serial.read();
    if(val != -1)
    {
      switch(val)
      {
      case 'w'://Move Forward
        advance (255,255);   //move forward in max speed
        break;
      case 's'://Move Backward
        back_off (255,255);   //move back in max speed
        break;
      case 'a'://Turn Left
        turn_L (100,100);
        break;
      case 'd'://Turn Right
        turn_R (100,100);
        break;
      case 'z':
        Serial.println("Hello");
        break;
      case 'x':
        stop();
        break;
      }
    }
    else stop();
  }
}
```

PLL Control Mode

The Romeo also supports PLLPhase locked loop control mode.

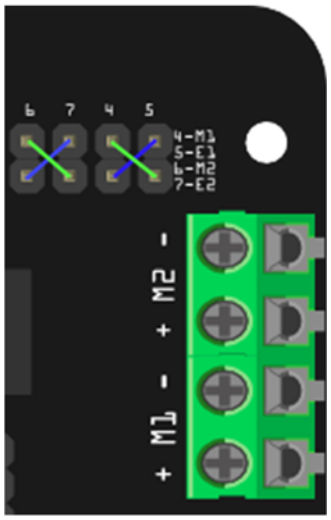


Fig5: PLL Motor Control Pin Allocation Configuration

Sample Code：

```
//Standard DLL Speed control

int E1 = 4;     //M1 Speed Control
int E2 = 7;     //M2 Speed Control
int M1 = 5;    //M1 Direction Control
int M2 = 6;    //M1 Direction Control

///For previous Romeo, please use these pins.
//int E1 = 6;     //M1 Speed Control
//int E2 = 9;     //M2 Speed Control
//int M1 = 7;    //M1 Direction Control
//int M2 = 8;    //M1 Direction Control

//When m1p/m2p is 127, it stops the motor
//when m1p/m2p is 255, it gives the maximum speed for one direction
//When m1p/m2p is 0, it gives the maximum speed for reverse direction

void DriveMotorP(byte m1p, byte m2p)//Drive Motor Power Mode
```

```
{

  digitalWrite(E1, HIGH);
  analogWrite(M1, (m1p));

  digitalWrite(E2, HIGH);
  analogWrite(M2, (m2p));

}

void setup(void)
{
  int i;
  for(i=6;i<=9;i++)
    pinMode(i, OUTPUT);
  Serial.begin(19200);      //Set Baud Rate
}
void loop(void)
{
  if(Serial.available()){
    char val = Serial.read();
    if(val!=-1)
    {
      switch(val)
      {
      case 'w'://Move Forward
        DriveMotorP(0xff,0xff); // Max speed
        break;
      case 'x'://Move Backward
        DriveMotorP(0x00,0x00);
        ; // Max speed
        break;
      case 's'://Stop
        DriveMotorP(0x7f,0x7f);
        break;

      }
    }
  }
}
```