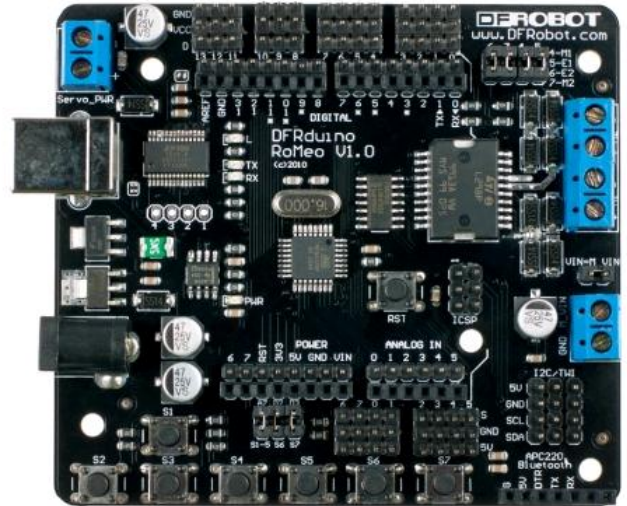
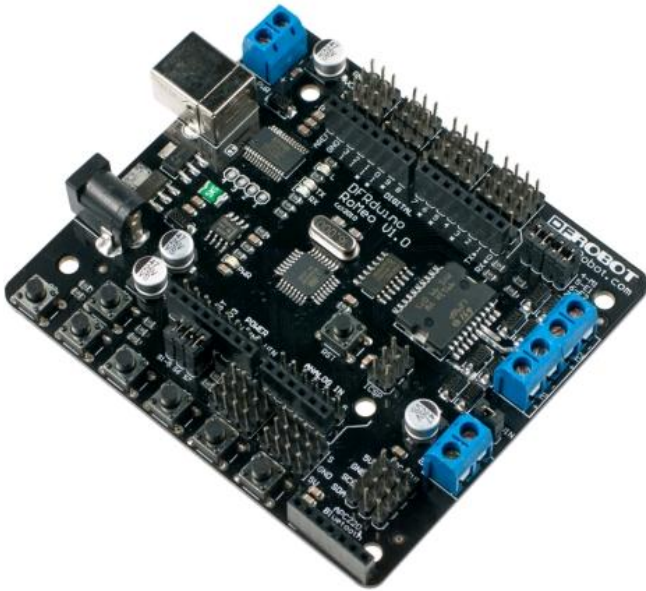


DFRduino Romeo, todo en un controlador (RoMeo V1.0)

Por favor, lea atentamente este manual antes de encender el dispositivo.



DFRduino Romeo

Romeo es un microcontrolador de todo-en-uno diseñado especialmente para la aplicación de la robótica. Beneficiarse de la plataforma Arduino de código abierto, con el apoyo de miles de códigos de fuente abierta, y se puede ampliar fácilmente con la mayoría de los complementos Arduino. El integrado controlador de 2 motores DC y la toma inalámbrica proporciona una forma mucho más fácil para comenzar su proyecto de robótica.

Especificaciones.

- Atmega 328
- 14 E/S digitales
- 6 Canales PWM (Pin 11,10,9,6,5,3)
- 8 canales E/S analógicos de 10 bits
- Interfaz USB
- Auto detección/conmutación de entrada potencia
- Cabecera ICSP para descarga directa del programa
- Interfaz serie TTL
- Soporte AREF
- Conexiones pin macho y pin hembra
- Conexiones integradas para modulo APC220 RF y módulo DF-Bluetooth
- Tres puertos I2C
- Control de dos motores CC en doble sentido, 2A corriente máxima
- 7 Pulsadores
- Alimentación CC: con alimentación USB o externo y 7V ~ 12V CC para motores
- Salida de CC: 5V / 3,3V CC y Salida de alimentación externa
- dimensión: 90x80mm. Peso: 60 gramos.

DFRduino Romeo Pinout

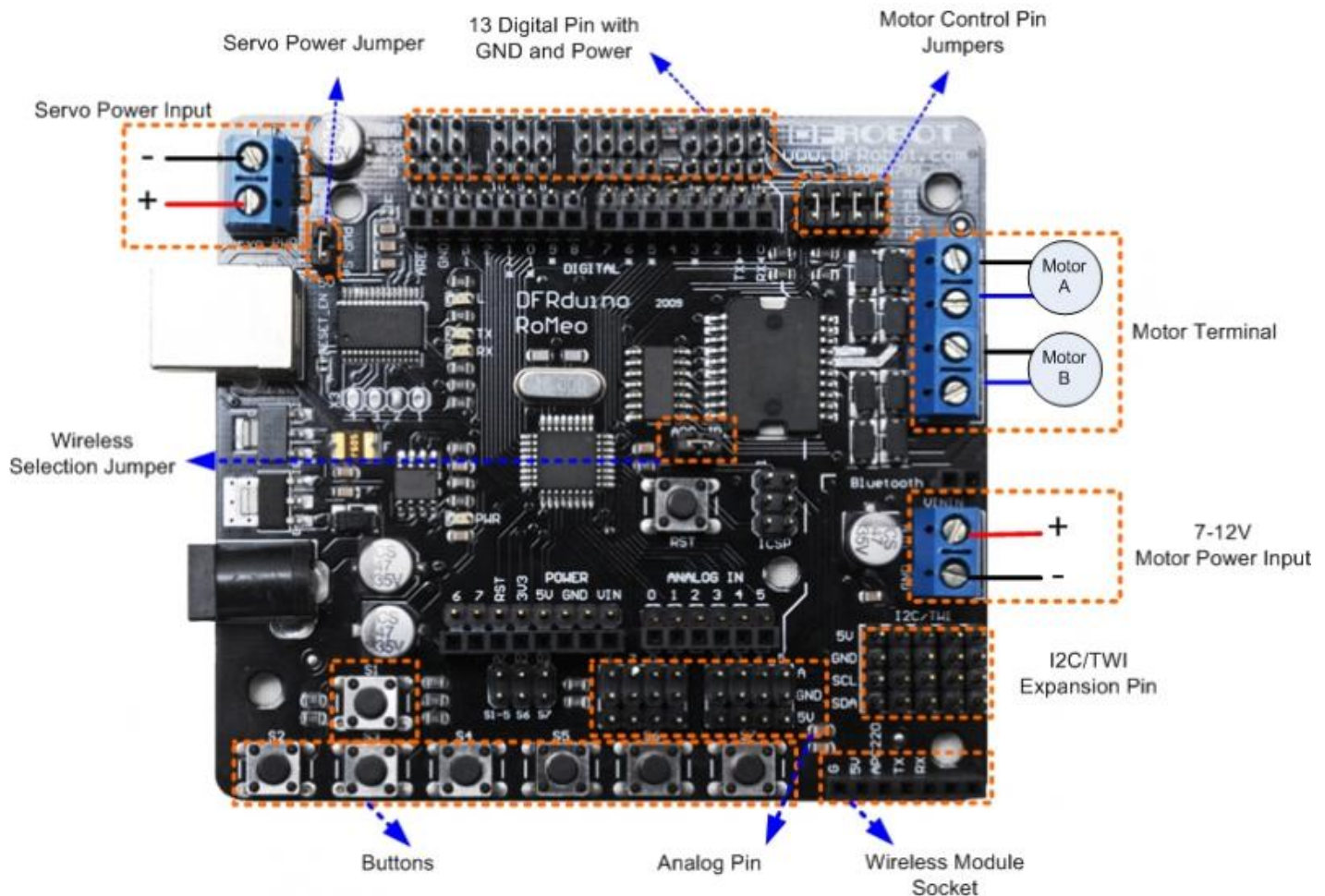


Figura 1 Romeo (Versión anterior.)

La imagen de arriba muestra todas las líneas de E / S y conectores del Romeo, que incluye:

- Una alimentación regulada del motor de terminales de entrada (6v a 12v)
- Una entrada no regulada Servo Terminal (4v a 7.2v)
- Una interfaz en serie para el encabezado del módulo módulo APC220/Bluetooth
- Dos puertos para motores DC - Soporta corriente del motor a 2A, cada terminal
- Una I2C/TWI Puerto - SDA, SCL, 5V, GND
- Un puerto analógico de 8 entradas analógicas - una entrada está ligada internamente a la tensión de alimentación
- Uno de uso general de E / S del puerto con 13 líneas I / O - 4,5,6,7 se puede usar para controlar motores directamente.
- Un pulsador de reset.
- Jumpers para activar / desactivar Control de Motores
- Jumpers para activar / desactivar los pulsadores.
- Jumper para activar / desactivar la alimentación externa de motores.

Antes de empezar

La aplicación de energía

Usted debe asegurarse de que la alimentación eléctrica al terminal de alimentación este con la polaridad correcta. La inversión de la polaridad puede dañar el Romeo. *No nos hacemos responsables de los daños, ni tampoco garantía contra daños.* Asegúrese de tomar el tiempo para aplicar la alimentación correctamente. De lo contrario, podrían ser costosos para usted!

De alimentación de USB: Sólo tiene que conectar el cable USB, y Romeo es capaz de trabajar. Tenga en cuenta que el USB sólo puede suministrar 500 mA de corriente. Esta entrada es capaz de satisfacer los requisitos para la aplicación con LED. Sin embargo, no es suficiente para alimentar motores de corriente continua o servos.

De alimentación de entrada de corriente para motores : Sólo tiene que conectar el cable de negativo de la fuente al terminal de tornillo marcado "GND", y luego conectar el cable positivo de la fuente al terminal de tornillo marcado "VIN".

NOTA: La tensión máxima de alimentación no puede exceder de 14 V DC.

Software

Romeo puede ser programado por Arduino IDE 0014 o versión superior. Se puede descargar en <http://arduino.cc/en/Main/Software> , por favor seleccione "Arduino Nano", como el hardware.

Romeo configuración

Alimentación para Servos.

Como la mayoría de los servos necesitan más corriente que la fuente de alimentación USB puede proporcionar. Un conector de alimentación externa suministra la energía que el servo necesita.

La V1.0 Romeo utiliza un conmutador automático para la selección de fuente de alimentación. Cuando la fuente de alimentación externa se ha aplicado, la alimentación del servo se activa automáticamente por la fuente de alimentación externa en lugar del USB.

Control de motores Pin Jumper

La aplicación de los puentes de Control de Motores Pines 4,5,6,7 destinará estos pines para el control de motores.

La eliminación de los puentes deja los pines 4,5,6,7, libres.

Bluetooth

Permitirá la comunicación Romeo a través de su módulo inalámbrico, como el módulo de APC220 DF y Bluetooth. Si un módulo inalámbrico está conectado.

Tutorial

Pulsadores.

Romeo tiene 7 pulsadores S1 a S7 (Figura 2). S1 a S5 para entrada de uso analógico con el pin 7, y S6, S7 que utilizan las entradas digital pin 2 y pin 3.

"Botón Mapa Pin"	
Alfiler	Función
Digital Pin 2	Botón S6
Digital Pin 3	Botón S7
Analógica Pin 7	Botón S1-S5



(Figura 2)

Ejemplo 1: uso de pulsadores

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

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); //vamos a utilizar LED a la salida de un latido del corazón
}
void loop()
{
    adc_key_in = analogRead(7); // leer el valor del sensor
    digitalWrite(13, HIGH);
    key = get_key(adc_key_in); // convertir en pulsar la tecla
        if (key != oldkey) // si se oprime una tecla se detecta
        {
            delay(50); // esperar que el tiempo de rebote
            adc_key_in = analogRead(0); //leer el valor del sensor
            key = get_key(adc_key_in); //convertir en pulsar la tecla
            if (key != oldkey)
```

```

    {
      oldkey = key;
      if (key >=0){
        Serial.println(msgs[key]);
      }
    }
  }
  digitalWrite(13, LOW);
}
// Convertir el valor ADC con el número clave
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 clave válida presionado
  return k;
}

```

Para activar S6 y S7,, se aplican los puentes que indica en el círculo rojo. **S6** utiliza Digital **Pin2**, **S7** utiliza Digital **Pin3**. Una vez que estos puentes se han aplicado, el pin 2 y 3 estarán ocupados (Figura 3).



 Figura 3 botón Activar Puentes

Ejemplo 2: uso de pulsadores:

```
int ledPin = 13;
int key_s6 = 2;
int val=0;
void setup()
{
  pinMode(ledPin, OUTPUT);      // Establecer Pin13 a modo de salida
  pinMode(key_s6, INPUT);      // Establecer Pin12 a modo de entrada
}
void loop()
{
  if(digitalRead(key_s6)==0)   //
  {
    while(!digitalRead(key_s6));
    val++;
  }
  if(val==1)
  {
    digitalWrite(ledPin, HIGH); //
  }
  if(val==2)
  {
    val=0;
    digitalWrite(ledPin, LOW);  //
  }
}
```

Ejemplo 3: uso de pulsadores:

//Código de la función: Presionando S6, enciende LED, Pulsando S7, apaga el LED.

```
int ledPin = 13;           //
int key_s6 = 2;           //
int key_s7 = 3;           //
void setup()
{
  pinMode(ledPin, OUTPUT); //
  pinMode(key_s6, INPUT);  //
  pinMode(key_s7, INPUT);  //
}
void loop()
{
  if(digitalRead(key_s6)==0) //
  {
    digitalWrite(ledPin, HIGH); //
  }
  if(digitalRead(key_s7)==0) //
  {
    digitalWrite(ledPin, LOW); //
  }
}
```

Doble control de velocidad y dirección de giro de motores de CC

Configuración de Hardware

Puede conectar dos motores al Motor Terminal Server. Y aplicar tensión a través del conector de alimentación (Figura 4).



Figure 4 Romeo Motor Connection Diagram

Pin Allocation

"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**

"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

Mode

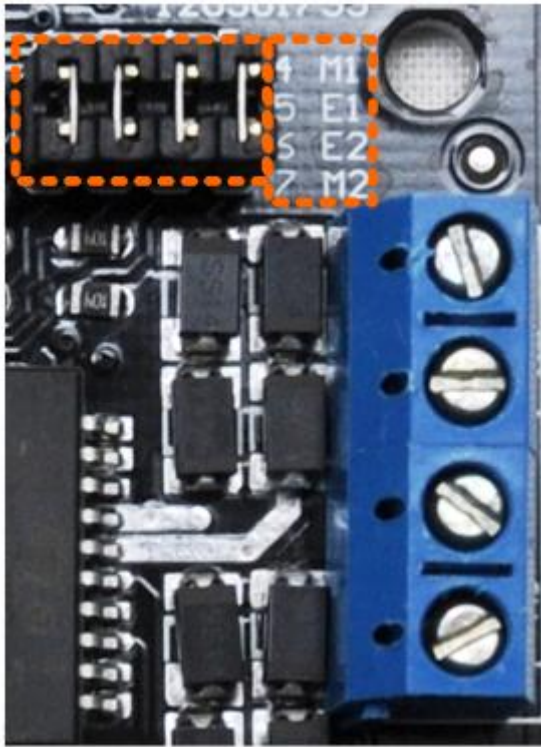


Figure 5 PWM Motor Control Pin Allocation

El PWM DC de control del motor se lleva a cabo mediante la manipulación digital de dos pines IO y dos pines PWM. Como se ilustra en el diagrama anterior (Figura 5), Pin **4,7** son pines de control de **dirección de giro** del motor, Pin **5,6** son pines de control de **velocidad**.

Ejemplo 4: uso de control de motores PWM.

//Standard PWM DC control

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

void stop(void)           //Stop
{
    digitalWrite(E1,LOW);
    digitalWrite(E2,LOW);
}
void advance(char a,char b) //Avanza
{
    analogWrite (E1,a);     //Velocidad de control PWM
```

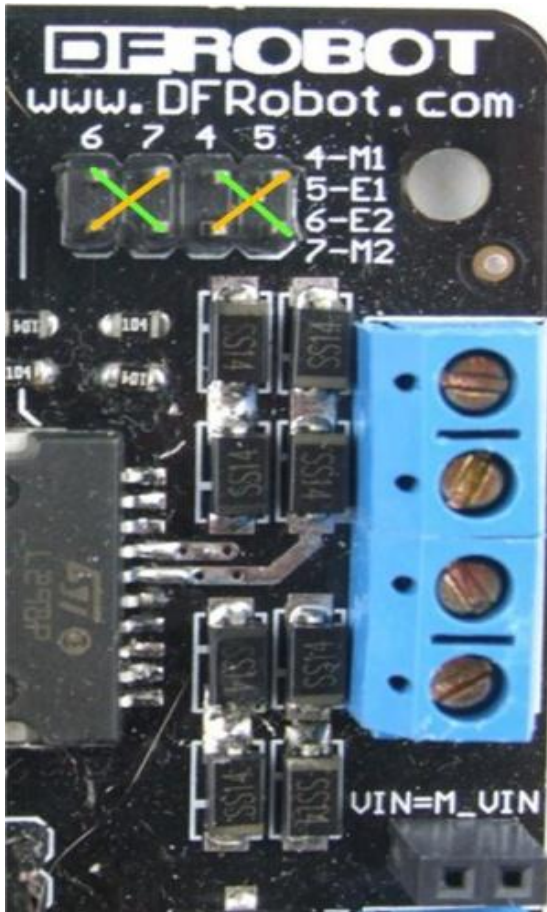
```

        digitalWrite(M1,HIGH);
        analogWrite (E2,b);
        digitalWrite(M2,HIGH);
    }
void back_off (char a,char b)           //Se mueve hacia atrás
    {
        analogWrite (E1,a);
        digitalWrite(M1,LOW);
        analogWrite (E2,b);
        digitalWrite(M2,LOW);
    }
void turn_L (char a,char b)            //Gira a la izquierda
    {
        analogWrite (E1,a);
        digitalWrite(M1,LOW);
        analogWrite (E2,b);
        digitalWrite(M2,HIGH);
    }
void turn_R (char a,char b)            //Gira a la derecha
    {
        analogWrite (E1,a);
        digitalWrite(M1,HIGH);
        analogWrite (E2,b);
        digitalWrite(M2,LOW);
    }
void setup(void)
{
    int i;
    for(i=6;i<=9;i++)
        pinMode(i, OUTPUT);
    Serial.begin(19200);           //Establecer la velocidad en baudios
}
void loop(void)
{
    char val = Serial.read();
    if(val!=-1)
        {
            switch(val)
            {
                case 'w'://avanza
                    advance (100,100); //Velocidad de control PWM
                    break;
                case 's'://retrocede
                    back_off (100,100);
                    break;
                case 'a'://Gira izquierda
                    turn_L (100,100);
                    break;
                case 'd'://Gira derecha
                    turn_R (100,100);
                    break;
            }
            delay(40);
        }
    else stop();
}

```

Modo de control PLL

Romeo también soporta modo de control PLL [Phase Locked Loop](#) .



Ejemplo 5 uso de control de motores PLL

```
//DLL estándar de control de velocidad

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

// Cuando m1p/m2p es de 127, se detiene el motor
// Cuando m1p/m2p es de 255, velocidad máxima avanzar
// Cuando m1p/m2p es 0, velocidad máxima retroceder

void DriveMotorP(byte m1p, byte m2p) // el modo de unidad de alimentación del motor
{
```

```
    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);        //Establecer la velocidad en baudios
}
void loop(void)
{
    char val = Serial.read();
    if(val!=-1)
    {
        switch(val)
        {
            case 'w'://Avanza
                DriveMotorP(0xff,0xff);
                break;
            case 'x'://Retrocede
                DriveMotorP(0x00,0x00);
                break;
            case 's'://Stop
                DriveMotorP(0x7f,0x7f);
                break;

        }
        delay(40);
    }
}

-----
```