Components

In this section is showed the components of the initial prototype of the DIT-100 model of drone.

Structure of the DIT-100

View product

This is the structure of the quad-rotor where are going to be fixed all its components.

Electronic Speed Control (ESC)

View product

The ESC is the device that allows to control a brushless motor. It can vary the motor's speed depending on the PWM signal received, and depending on the device it also can vary the direction, by means of generate a three phase electric power.

This is an interesting project with a lot of documentation about pilot unmanned aerial vehicles, and it can be found information about how to use the ESC to control brushless motors:

wiki.openpilot.org

The fundamentals:

1. The signal received by the ESC is a PWM signal. In the case of the ESC that is going to be used the frequency of the PWM signal is of 400 Hz.
2. The ESC, depending on the PWM signal received, generates an output to control the motor with a frequency of about 8 kHz, in general.

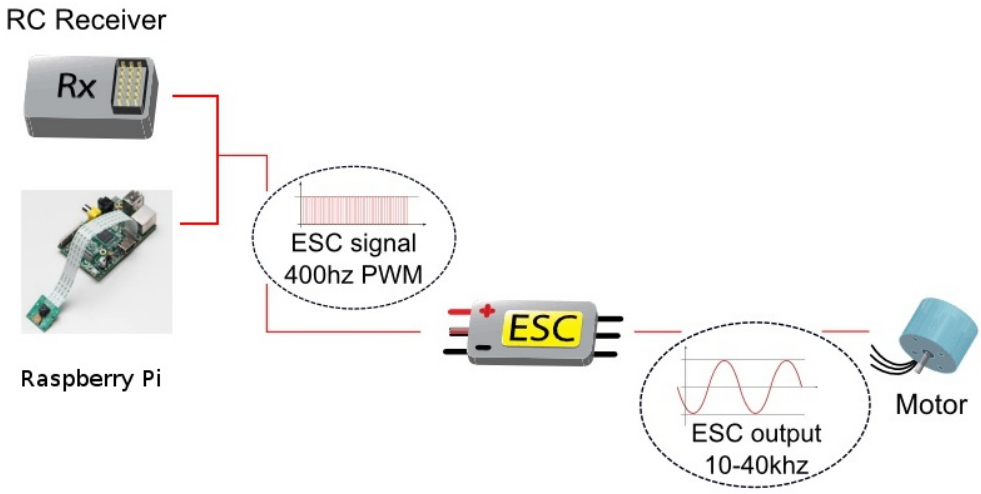
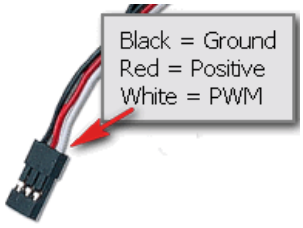


Image obtained from http://wiki.openpilot.org

3. The duty cycle of the PWM signal that receives the ESC vary in the range of 1 ms to 2 ms, as shows the next figure. The minimum duty cycle is of 1sms and the maximum is of 2 ms.
4. The ESC has a connection cable through which the PWM signal is received. In some cases the cable provides the supply necessary to power up the receiver or other used servos, by means of an internal regulator. This feature is not included in the ESC of type OPTO.



How to choose an ESC:

In general, given a motor, with a maximum value of amperage, the ESC has to be selected to support this amount of amperage plus about a 25% or more. For example, for a motor with a maximum amperage of 15 A, the ESC has to support at least 18 A. More information about the choice of an ESC can be found here.

Motors

View product (800kv)

View product (1200kv)

The motors recommended to build multi-rotors, are brushless motors.

Parameters of the motor:

Kv: velocity constant of the motor measured in RPM/v. It indicates the RPM of the motor without load supplied by the peak of voltage supported. For example, if the motor is of 5700kv, supplied with 11.1V, the motor will have a nominal velocity of 63270 RPM (5700 RPM/V x 11.1V).

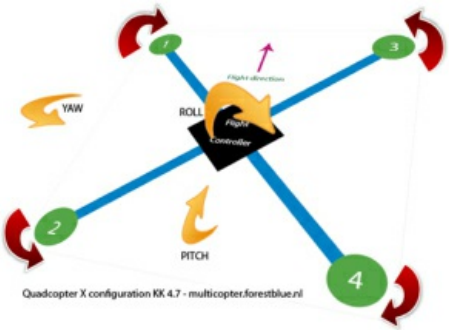
In general, following this indications, a low value of kv allows more thrust, because the motor provides more torque, although less velocity. A high value allows more velocity but less thrust.

Electric power: work per unit of time at a determined RPM value. It is measured in watts .

How to choose a brushless motor:

The motor has to be selected depending on the weight of the flight model.

Configuration of the motors on the frame:



Propellers for the motors

View product (8045)

View product (1045)

In a general way, following the indications found in Internet, the bigger the propellers the higher is the provided thrust, obviously depending on the features of the motors.

Battery

View product (2200mAh)

View product (5000mAh)

Here it can be found a very good description of the lithium batteries.

Parameters of the battery

Voltage:

The voltage of the batteries is indicated with a nomenclature that indicates the number of cells and the position of them. Below is shown a list of this nomenclature for a better understanding:

- 3.7 volt battery = 1 cell x 3.7 volts (1S)
- 7.4 volt battery = 2 cells x 3.7 volts (2S)
- 11.1 volt battery = 3 cells x 3.7 volts (3S)
- 14.8 volt battery = 4 cells x 3.7 volts (4S)
- 18.5 volt battery = 5 cells x 3.7 volts (5S)
- 22.2 volt battery = 6 cells x 3.7 volts (6S)
- 29.6 volt battery = 8 cells x 3.7 volts (8S)

- 37.0 volt battery = 10 cells x 3.7 volts (10S)
- 44.4 volt battery = 12 cells x 3.7 volts (12S)

Capacity:

Capacity indicates how much power the battery pack can hold and is indicated in milliamp hours (mAh). This is, the amount of load that can be put on the battery for 1 hour at which time the battery will be fully discharged. To calculate the time that takes to discharge a battery the following formula is used:

$$t = \frac{Q_P}{I^k}$$

Discharge rate (C):

Discharge rate indicates how fast a battery can be discharged safely. For example, given a battery of 1000 mAh with a discharge rate of 20C, it can be obtained from this battery a constant load of 20000 mA (20C x 2000 mAh = 20000 mA = 20 A). So, providing the maximum load the battery will discharge in t = 1000 mAh / 20000 mA = 0,05 h = 3 min, using the previous formula.

For charging the same concept is applied although the rate is usually lower.

Internal resistance:

The three previous parameters are standard in the industry. To these is added the internal resistance which is verifiable and one of the best parameters to check the state of the batteries. As mentioned in the page of reference, most of the batteries with high capacity and with high values of discharge rate should have an internal resistance between 2 and 6 milliohms, when they are new.

Things to be taking into account

Heat:

To prolong the life of the batteries as much as possible, something important is to control their temperature. Check that the battery does not heat too much. If it is possible to handle the battery when the whole system is working it says that the battery is the properly and supports the necessary load.

Discharge:

Discharge over the limits of the batteries reduce their life and can even burn themselves. A rule to avoid this is to not discharge them more than the 80% of its capacity.

Charge:

The cell of a LiPo battery is fully charged when reaches the 4,2 volts, in general. **To charge a battery over this value can cause damage in it**, reducing its life time. Due to this it is necessary to charge the batteries with such a device that allows a balanced charging.

Tips for a secure charging:

1. The charge has to be made in a secure fire resistant zone.
2. It is recommended to wait at least 15 minutes after using a LiPo to let it cool before charging it.
3. Never leave the battery charging alone.
4. Have a extinguisher at hand.

Electronics

Raspberry Pi board

View product

Preparing the operating system to boot from an USB:

This is interesting to do because the SD cards brakes up in a few months, at least in our case.

1. Download the operating system image (In this case raspbian)
2. Copy the image into an SD card. For example:
   sudo dd if=2013-09-25-wheezy-raspbian.img of=/dev/sdc
   Then, put the file system into an USB:
   sudo rsync -avz $PATH_SD $PATH_USB
3. Modify the file /boot/cmdline.txt (in the SD card) to replace the root file system for the USB unit. The content should be:
   `dwc_otg.lpm_enable=0 console=ttyAMA0,115200 kgdboc=ttyAMA0,115200 console=tty1 root=/dev/sda1 rootfstype=ext4 elevator=deadline rootwait rootdelay=5`
4. Modify the file /etc/fstab (in the USB) to replace the unit where to mount '/' for the unit of the USB:
   /dev/sda1 / ext3 defaults,noatime 0 1
   #/dev/mmcblk0p2 / ext4 defaults,noatime 0 1

Notes: In the version 2 of the Raspberry Pi, when an USB wifi is connected and the Raspberry Pi is working, it reboots. This is due to the lack of fuses, which the version 1 of the Raspberry Pi has. So, the wifi has to be connected before boot the Raspberry Pi. [Info]

Documentation:

- How to use the GPIOs on Raspberry Pi

Wifi adaptor

View product

To activate the wifi (wpa) on the Raspberry Pi is necessary to modify the file /etc/network/interfaces as follows:
auto wlan0
iface wlan0 inet dhcp
wpa-ssid *red*
wpa-psk *password*

IMU (gyroscope, accelerometer, magnetometer and altimeter)

View product – Manufacturer web page

This product is the Polulu AltIMU-10. It is an inertial measurement unit (IMU) and an altimeter. The components are:

1. The gyroscope L3GD20
2. The accelerometer and the magnetometer LSM303DLHC
3. The altimeter LPS331AP

The whole unit is accessible through I^2C interface. This operates with a voltage in the range of 2,5 V to 5,5 V.

Considerations about the connections:

Pins:

- **SCL:** I²C clock line
- **SDA:** I²C data line
- **GND:** I²C ground line
- **VIN:** main 2.5 V to 5.5 V power supply connection
- **VDD:** This line has a double functionality. First, if VIN is supplied greater than 3,3 V, VDD is a regulated 3.3 V output that can supply up to approximately 150 mA to external components. Second, when interfacing with a 2.5 V to 3.3 V system, VIN can be left disconnected and power can be supplied directly to VDD. Never supply voltage to VDD when VIN is connected, and never supply more than 3.6 V to VDD.

I²C communication:

Each component of the unit has an independent slave address. As follows, it is shown the relation table between each component and its slave address:

Sensor	Slave address
Gyroscope (L3GD20)	1101011b
Accelerometer (LSM303DLHC)	0011001b
Magnetometer (LSM303DLHC)	0011110b
Altimeter (LPS331AP)	1011101b

Sample code:

- L3G4200D y L3GD20 (Gyroscope)
- LSM303 (Accelerometer and magnetometer)
- LPS331 (Altimeter)
- MinIMU-9 + Arduino AHRS (Test program)
- OpenIMU

Configuration notes:

- The pressure sensor, gyro, accelerometer, and magnetometer are all off by default. You have to turn them on by setting the correct configuration registers.
- It is possible to read or write multiple pressure sensor, gyro, or accelerometer registers in a single I²C command by asserting the most significant bit of the register address to enable address auto-increment.
- The magnetometer will not update its data until all 6 data bytes have been read during a single I²C transfer. All the bytes can be read in the same transfer using the magnetometer's automatic sub-address updating feature (this feature is enabled by default).
- The pressure sensor has a 24-bit pressure reading. The gyro, accelerometer, and magnetometer have all output readings in a 16-bit *format* (obtained by combining the values in two 8-bit registers for each axis), but only the gyro readings contain 16 bits of *precision*. The accelerometer and magnetometer readings contain a maximum of 12 bits of precision; for the accelerometer, at least the lowest 4 bits of the output values are always 0, and for the magnetometer, the highest 4 bits of the output values are always 0.
- The accelerometer gives low-resolution 10-bit readings by default (the lowest 6 bits of the output are always 0). To get the full 12-bit resolution, you must set the HR (high resolution) bit in the CTRL_REG4_A register.
- The LSM303DLHC combines an accelerometer and a magnetometer made by separate manufacturers into

one IC, so there are fairly significant differences in their features, functionality, and interfaces. The interfaces of the LPS331AP and L3GD20 are similar to that of the accelerometer in the LSM303DLHC.

Documentation:

- Datasheet LSM303DLHC (Accelerometer and magnetometer)
- Datasheet L3GD20 (Gyroscope)
- Datasheet LPS331AP (Altimeter)
- Schematic (AltIMU-10)
- Operation guide
- Operation guide of the accelerometer and the magnetometer (AN3192)
- Operation guide of the accelerometer combined with the gyroscope
- Operation guide of the altimeter (AN4159)

Arduino libraries:

- L3G4200D y L3GD20
- LSM303
- LPS331
- MinIMU-9 + Arduino AHRS
- OpenIMU

Python libraries for the Raspberry Pi:

- L3GD20
- LSM303DLHC

Usage examples:

- Sigway controlled by a Raspberry Pi using the AltIMU-10 (code)
- Test program using the MinIMU-9

Configuration of the Raspberry Pi:

Once the IMU is connected, to verify that it was done correctly it is necessary to configure the Raspberry Pi and install some tolls first:

1. Install the necessary tools to use the I²C:
   sudo apt-get install i2c-tools
2. Make the following modifications to add the necessary modules to manage the I²C bus:
   - In the file /etc/modprobe.d/raspi-blacklist.conf comment the line blacklist i2c-bcm2708
   - In the file /etc/modules add the lines i2c-dev and i2c-bcm2708
3. Reboot the Raspberry Pi

After the reboot, execute the next command which shows the devices connected to the I²C bus and their directions:

sudo i2cdetect -y 1

The result should be something like this:

```
pi@raspberrypi ~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- 19 -- -- -- -- 1e --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- 5d -- --
60: -- -- -- -- -- -- -- -- -- -- -- 6b -- -- -- --
70: -- -- -- -- -- -- -- --
pi@raspberrypi ~ $
```

To access to the I²C bus through Python code is necessary to install the python-smbus library:

sudo apt-get install python-smbus

To modify the I²C bus velocity it is necessary to create a file named /etc/modprobe.d/i2c.conf with next content:

options i2c_bcm2708 baudrate=200000

Note: baudrate is in bits

For the calibration of the accelerometer and the magnetometer it is used the matplotlib library. This library draws the values of these devices in such a way that the calibration can be seen visually. To install the library:

sudo apt-get install python-matplotlib

Root permissions are necessary to access to the I²C bus by default. So, to do it as a normal user, this user can be added to the I²C group as follows:

```
sudo usermod -a -G i2c pi
```

Implementation notes:

– Accelerometer and magnetometer:

in the AN3192 document, it is explained the necessary theory to calculate the three angles of interest (roll, pitch y heading) which are obtained with the accelerometer and the magnetometer.

– Altimeter:

To calculate the altitude, given the current pressure and temperature, is used the Hypsometric equation:

$$h = z_2 - z_1 = \frac{R \cdot \bar{T}}{g} \cdot \ln\left(\frac{p_1}{p_2}\right)$$

where:

h = thickness of the layer [m]

z = geometric height [m]

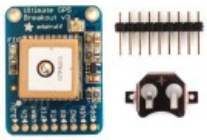
R = specific gas constant for dry air

$\bar{T}$ = mean temperature in Kelvin [K]

g = gravitational acceleration [m/s^2]

p = pressure [Pa]

GPS



View product

Documentation:

- MTK3329/MTK3339 command list
- Datasheet – version 3

Tutorials:

- Using GPS with the Raspberry Pi

Configuration in the Raspberry Pi:

By default, the Debian image for the Raspberry Pi uses the UART as a debugging serial line. So, it is necessary to modify the configuration to leave free the UART for the GPS. For this:

1. Modify the file /boot/cmdline.txt to eliminate the content referring to the serial line (ttyAMA0). The result should be something like this:
   `dwc_otg.lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 rootfstype=ext4 rootwait`
2. Modify the file /etc/inittab, to comment the line referring to the serial line (ttyAMA0). The result should be something like this:
   #Spawn a getty on Raspberry Pi serial line#T0:23:respawn:/sbin/getty -L ttyAMA0 115200 vt100
3. Reboot the Raspberry Pi.
   Once the Raspberry Pi reboots, and the GPS is connected appropriately the data can be obtained as follows:
   sudo cat /dev/ttyAMA0Y
   And the data flow should be something like this:
   $GPGGA,000655.800,,,,,0,0,,,M,,M,,*46$
   GPGSA,A,1,,,,,,,,,,,,,*1E
   $GPRMC,000655.800,V,,,,,0.00,0.00,060180,,,N*4C
4. Install the necessary libraries and tools to manage the GPS:
   sudo apt-get install gpsd gpsd-clients python-gps

Raspberry Pi camera

View product

Battery monitor

View product

As it is mentioned, it is important to control the levels of capacity of the LiPo batteries. In general, it is recommended **not to discharge them over the 80% and not to charge them over the 100%**. Above all, it is very important to control the discharge of the batteries because exceeding the limit causes copper deposits to form damaging the batteries. This damage can lead to the burning of the batteries. For this is important that the system monitors the battery capacity.

It is possible to know the capacity through the provided voltage. In general, one full loaded cell provides 4,2 V. If the battery is at 80% discharged it will give an approximate open circuit voltage of 3.72 to 3.74 volts.

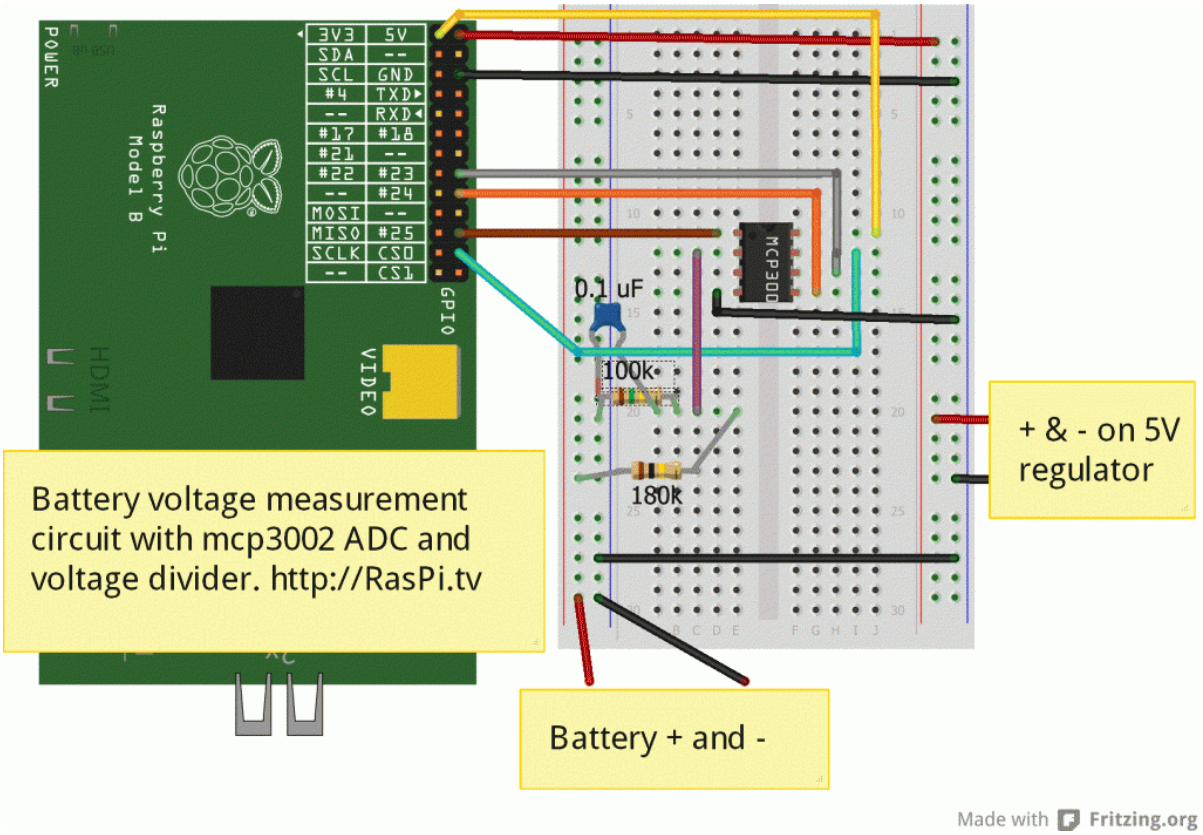
Implementation of a battery monitor using a Raspberry Pi:

Web page with the description [PDF]

Documentation:

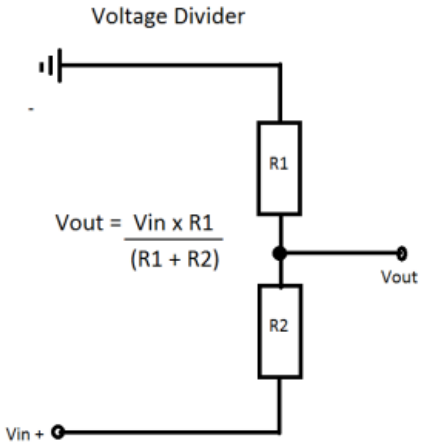
- Datasheet MCP3008
    1. There is an error in the chip documentation. The bit sequence in it is as follows:
       Null Bit – B9 – B8 – B7 – B6 – B5 – B4 – B3 – B2 – B1 – B0
       But in tests and comparing with found code on Internet it seems that the real sequence is as follows:
       B9 – B8 – B7 – B6 – B5 – B4 – B3 – B2 – B1 – B0 – Null Bit
- Web page where it is explained how to interface the ADC mpc3008

In our case it is necessary to monitor a range of values between 3,6 and 4,2, being the critical value 3,72 V (20%). The schematic circuit of the battery monitor is as follows:



Picture taken from http://raspi.tv/2013/controlled-shutdown-duration-test-of-pi-model-a-with-2-cell-lipo

Where the resistances work as a voltage divider, as it is indicated in the next figure:



Picture taken from
http://raspi.tv/2013/controlled-shutdown-duration-test-of-pi-model-a-with-2-cell-lipo

So, in this case the values for the resistances and the capacitor are:

1. R1 = 100000
2. R2 = 180000
3. Capacitor = 0.1 uF

Maximum value at the output of the divider (case 4,2):

Vout = (Vin x R1)/(R1 + R2) = (4,2 x 100000)/(100000 + 180000) = 1,5 (theoretically)

Maximum value at the output of the divider (case 8,4):

Vout = (Vin x R1)/(R1 + R2) = (8,4 x 100000)/(100000 + 180000) = 3 (theoretically)

Maximum value at the output of the divider (case 12,6):

Vout = (Vin x R1)/(R1 + R2) = (12,6 x 100000)/(100000 + 180000) = 4,5 (theoretically)

So, because the higher input is of 4,5 volts, the minimum supplied voltage for the chip has to be higher than this. This is to supply the chip with 5 V from the GPIO of the Raspberry Pi.

Once everything is connected, with the code of the designed library is possible to obtain the voltage at the input of the voltage divider. Just indicating the values of the resistances and applying the next formula:

Vin = (Vout x (R1 + R2)) / R1

Configuration on the Raspberry Pi:

The battery monitor is managed trough the SPI. To make the SPI accessible through the GPIO the next modifications have to be done:

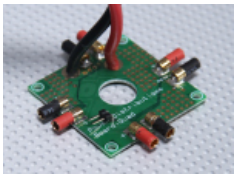
- In the file /etc/modprobe.d/raspi-blacklist.conf comment the line blacklist spi-bcm2708
- Then, reboot the Raspberry Pi.
- Once the Raspberry Pi reboots, execute ls /dev/spidev* and the output should be:

```
pi@raspberrypi ~/workspace/drones $ ls /dev/spidev*
/dev/spidev0.0  /dev/spidev0.1
pi@raspberrypi ~/workspace/drones $
```

To access to the SPI device from Python, it is necessary to install the `spidev library`:

```
sudo apt-get install python-dev
wget https://raw.github.com/doceme/py-spidev/master/setup.py
wget https://raw.github.com/doceme/py-spidev/master/spidev_module.c
sudo python setup.py install
```

Power Distribution Board



View product

PWM controller

View product

Documentation:

- Datasheet PCA9685

Expansion board for the GPIO

View product

Charger

View product

For the selected charger it is necessary a power supply.

Power supply

View product

To work without batteries, which makes the debug process easy, the power supply has to be at least of 33 A. This is because one motor, with the features indicated, at the higher RPM value consumes about 8 A and the Raspberry Pi

consumes 1A, so the whole system 4 motors * 8A + 1 Raspberry PI * 1A = 33 A. Here you can find a power supply that can supply this amount of current.

Compártelo:

Twitter Facebook Google

Cargando...

Deja un comentario

Introduce tu comentario aquí...