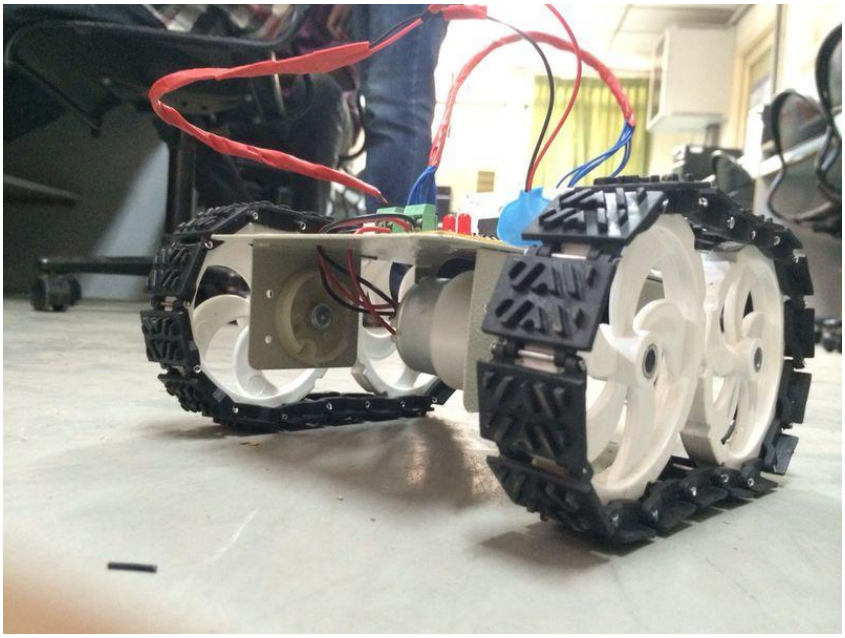
PIBOT



Introduction

This project is designed as a way of learning physical computing (the combination of computer programming with electronic circuits) in a fun way. When first starting out with electronics then getting a LED to flash can be quite exciting, and getting it to respond to programming done on a computer is another big leap, but once you have mastered those skills then it can be easy to lose interest. What we need is something that is exciting, fun to make, something you can show off to friends. The next challenge is that it needs to be affordable. It needs to be within a reasonable price that children (or their parents) can pay for an educational toy; unfortunately for most of us this rules out Lego Mindstorms and many commercial robots. I therefore started looking for an inexpensive robot vehicle kit that could be used for others learning about robotics. The result was the Pibot shown on the front cover. The reactions I have had when taking the robot out in public has shown that children do indeed get excited when they see a robot, especially when you let them have a go at controlling it using a mobile phone or tablet.

Design specification for Pi-Bot:

This aim of the project is to create a robot vehicle that can be controlled from a computer or mobile phone. This is being made as fun project that can help demonstrate the principles involved in designing and building a project. The vehicle should be relatively cheap and simple to build so that it can be replicated by a school or college student.

• It should be simple to make using tools available in a standard workshop.

• Vehicle should be able to move around on any floor.

• It should be possible to control the vehicle using a computer or other computing device.

• Maximum cost is Rs.6500/- (excluding batteries and optional camera).

• It should be able to have a camera mounted to take photos of the environment.

Live view or video recording would be an advantage (not required).

• It should provide some flexibility for adding additional features.

• It should be self powered.

• Communication should be wireless so that no cables are required connecting to the vehicle.

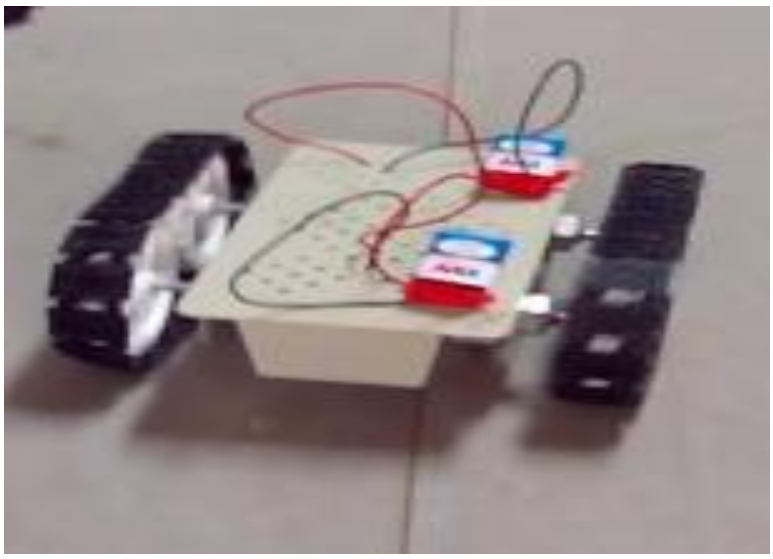
• The vehicle should be small enough so that it can be taken to events to give demonstrations.

Deciding on main components

The two main components that were decided upon were the Robot Chassis for the body and wheels, with a Raspberry Pi to control it. Before finalising the design it is a good idea to consider alternatives. I was not able to find an alternative robot chassis that would fit in with the specification (mainly because of the cost aspect), although since starting there are some clones of the chassis and at the time of writing there is a simpler single layer design looking for funding through Indiegogo. I also considered an Arduino compared with a Raspberry Pi for the processor. The decision to use the Raspberry Pi came down to supporting wireless control (it is possible with Arduino, but adds significant additional cost) and that programming a Raspberry Pi is generally considered easier than programming an Arduino. Having decided on the Raspberry Pi another decision to be made was what wireless method to use. The two most common ways of communicating using wireless are Wi-Fi (wireless networking) or Bluetooth. Neither of these are included in the Raspberry Pi, but can be added easily with a low cost USB adapter. Wi-Fi was chosen because of it's flexibility (it can also be used to configure the Raspberry Pi) and flexibility for sending images from the camera. In use I have found one disadvantage to using Wi-Fi, which is that it is prone to interference from other Wi-Fi networks. The robot uses a small USB dongle which does not have the signal strength to compete with strong wireless access points (WAP) if they are in the same room. I found that I had to keep the tablet I was connecting with close to the robot if it was using the same channel as a nearby WAP. The Wi-Fi configuration can be changed to use a different channel, which may need to be changed when taking the robot to a different site. More detailed decisions needed to be made later on in the design process, but having decided on this main components made the remaining decisions easier later on.

Robot chassis kit

The robot chassis is a commercial robot vehicle kit available from several electronic and robot suppliers. The chassis appears to have been created with the Arduino in mind as the power connector from the battery pack matches with the Arduino power supply. This is something we will address later when we look at connecting the power supply to the Raspberry Pi. Instructions for assembling the kit are provided and are fairly easy to follow. When complete it should look like the photo's below



Looking at the underside we can see how the robot vehicle will move around. There are two main drives wheels each of which is connected to its own motor. There is then caster which is effectively a large ball-bearing which can move in any direction.

Raspberry Pi

The robot chassis appears to have been designed for use with the Arduino, which is an open source programmable controller. Whilst the Arduino can control the motors and make simple decisions based on input sensors it does not have the processing capability as the Raspberry Pi, which is what we are using in this project. The Raspberry Pi is a small low cost computer designed specifically for education. It is based around an ARM processor which can run the open source Linux operating system and includes a GPIO (General Purpose Input Output) connector which can be used to interface to electronic circuits that can control the robot. The Raspberry Pi can also be connected to a camera which gives the robot the ability to see it's surroundings, or can be used to send images back to the operator. There are currently two different models of the Raspberr y Pi. The model A is more basic with only one USB port, no wired Ethernet port and only 256Mb of memory, the model B includes additional features including an additional USB port, ethernet port and an increase to the amount of memory to 512MB. The model A is also less expensive.The Raspberry Pi model A is recommended for this project as we won't be using a wired Ethernet port and the model A uses less power than the model B. The fact that the model A uses less power is useful when running on batteries. It can still run on a Raspberry Pi model B if you have one already, but you may need to consider a different power supply (see later). It's not possible for the Raspberry Pi to control the motors directly, so we are also going to need an electronic circuit.

When designing a circuit it's a good idea to start by using a prototyping breadboard so we'll be adding a breadboard onto the robot chassis. The Raspberry Pi and breadboard can be installed in various different positions on the robot chassis. These can be installed on either the top or bottom plates of the chassis. For now I recommend using the top layer for the breadboard as it will make it easier to make any changes to the electronic circuit. You will need to consider the length of the cables required as well as any connectors on the Raspberry Pi that you would like to connect to. For example the camera comes with only a short ribbon cable to connect it to the Raspberry Pi, which is the reason I placed the Raspberry Pi close to the camera. I also put the Raspberry Pi with the GPIO connector closest to the breadboard so that only a short lead is required to connect to the breadboard. If you are going to be using a Wi-Fi dongle you need to ensure there is space near the USB ports for the dongle (such as by having the USB port point towards the side or the chassis) and if you want to add a speaker (so the robot can speak) then you should ensure that there is space by the 3.5mm audio socket to plug in a speaker cable. Finally whilst you won't be connecting the robot directly to a TV or monitor when it's in use it may be a good idea to leave space near the HDMI connector to be able to connect a screen when doing the initial set-up. Access to the HDMI port is not actually required as the set-up can be done on a different Raspberr y Pi / computer or the Raspberry Pi could be temporarily disconnected when a new image is created.

Image of Raspberry pi:

Raspbian Linux operating system

The recommended operating system to run on the Raspberry Pi is Raspbian Linux. This is a distribution specifically created for the Raspberry Pi based upon Debian Linux. It can be purchased pre-installed on a SD card from the main suppliers (look for recent version of NOOBs SD cards), or can be downloaded from the Raspberry Pi website for install onto an SDcard http://www.raspberrypi.org/downloads .

Linux is a free open source operating system. It is free for anyone to use and its source code is

available. Much of the software within the distribution is provided under the GPL (GNU General Public License) which means that if you modify and share the code then you need to make your modifications available under the same conditions. If you are familiar with other operating systems such as Windows, then you will find the Linux graphical desktop (sometimes referred to as a GUI – Graphical User Interf ace) familiar. It has a look and feel more reminiscent of older versions, which is due to the low specification processor needing a leaner operating system. Modern Linux distributions running on a powerful computer provide a more feature rich experience

One thing that is different to some other operating systems is that you can run the full operating system without needing to have a graphical environment loaded and you can run commands on the computer remotely using the command line. This may be a little strange for those used to clicking on icons, but provides a lot of control and is ver y powerful. It is particularly useful in projects such as this robot where we can control and program the robot without needing to connect it to a monitor or TV.

It may be useful to spend some time getting familiar with Linux before embarking on the rest of this project. To find out more about the Linux operating system see http://www.penguintutor.com/linux/about-linux or one of the many guides on Linux or the Raspberry Pi.

Initial configuration of the Raspberry Pi

We will need to make some configuration changes to the Raspberry Pi configuration to allow us to use certain features.

Download the NOOBS image

In this section we will set-up the Raspberry Pi with the Raspbian operating system. If you are already familiar with the Raspberry Pi and have set-up the operating system then you can skip this section, but ensure you read the section on creating a Wi- Fi hotspot if you would like to be able to control the robot outside of the range of your home wireless network.

For this project we will be using the Raspbian operating system. This is easiest to install using the NOOBS software image. You can purchase an SD card with the operating system pre-loaded or you can download the software for free and install it onto a standard SD card. I assume you have an existing SD card and will be downloading the image from the Internet.

You will need a 4GB SD card (or micro- SD card in an SD adapter). If it is a new pre-formatted card then you should be able to use it as it is, but otherwise you should format the card using the SD card association's formatting tool: https://www.sdcard.org/downloads/formatter_4/

Formatting the card and running the NOOBS initial configuration will delete all the data on the SD card. Make sure you have made a copy of any data you want to keep before formatting the card.

Download the NOOBS image from the Raspberry Pi website:

http://www.raspberrypi.org/downloads

You need to select the standard NOOBS image rather than the Lite version (especially if you are

using a model A which has no ethernet port). The image is over 1GB in size and is usually quicker to download using a torrent client, although there is a standard web link on the website as well. This guide is based on NOOBS version 1.3.2, although it should work for future versions as well

Once you have downloaded the file (it should have a file ending with .zip) you should extract the files and copy all the files from the zip to the SD card.

Note: Ensure you copy the contents from inside the zip file – not the zip file itself.

Connect the Raspberry Pi for first time install

The easiest way to perform the initial configuration of the Raspberry Pi is to connect it temporarily to a TV or screen and keyboard. The Raspberry Pi can be connected to a TV using a standard HDMI cable, or if you have a monitor that has a DVI connector then you can connect it to that using a HDMI to DVI connector. Alternatively you can use the RCA composite connector to connect to an analogue TV; if using an analogue TV you will need to press a button during start-up so that it knows to use the RCA connector (press 3 for PAL as used in most of Europe or 4 for NTSC used in the USA).

Your SD card should be inserted in the slot on the Raspberry Pi and a keyboard in the USB port. The initial configuration can be performed with just a keyboard which is good as we don't have enough USB ports on the Model A to connect a mouse as well. If you do want to use a mouse then you may need to use a USB hub.

Finally insert a micro-USB power supply (used for most mobile phones) into the power socket and the NOOBS first start screen will appear on the screen. Choose to install the Raspbian operating system which is the recommended choice (press space) and then press i to start the install. It will take a while to repartition the SD card and install the operating system, after which it will restart with the Linux operating system After the operating system boots for the first time the Raspberry Pi Software Configuration Tool will run (raspi-config). This tool gives a menu which makes it easier to configure some features of the operating system. I have listed some of the useful options below:

Change User Password – Recommended to keep your system secure. The default username is pi and the password is raspberry.

Enable Boot to Desktop – This is not required for a robot. It provides a graphical interface similar to that used by Windows

Internationalisation Options – If you are in the UK then you don't normally need to change this. If you are in a different country or use a non-UK keyboard then you may want to change this to your local country.

Enable Cam era – This is required if you want to use the Raspberry Pi camera module

Advanced Options – You can reduce the memory allocation to 16MB and provide a unique hostname for your Raspberry Pi. It is normally a good idea to check for an updated version as well, but that is only possible when connected to the Internet.

Once you have made the changes choose finish and then reboot the Raspberry Pi.

Configure the static IP address

We now need to reconfigure the network with a static IP address. Whilst it is possible to use the same IP address range as you have on your existing Wi-Fi network we can avoid some potential problems by using a different network range. This will help if you try and connect from a computer that is connected to your local network (using wired ethernet) and you are connecting to the robot using wireless. There are a number of network ranges that are reserved for local use (chances are that your normally wireless network uses one of these ranges). I have chosen the network 10.5.5.0 / 24 as this is less commonly used. We will set the robot to 10.5.5.1 which is the first usable IP address in that range.

sudo nano /etc/network/interfaces

Find the entry for the wireless LAN which we changed previously (wlan0).

Remove the iface wlan0 entry and the wireless network information and replace with:

iface wlan0 inet static

Logging on to the Raspberry Pi using SSH

Now that the Raspberry Pi has a working network connection it should be possible to remove the connections to the screen and keyboard and continue the rest of the configuration over the network. The usual way of connecting to a Linux computer is using SSH. This uses a text based client that will run a shell on the "server" (see what is a server? below). It allows you to run commands in the same way as we did for the configuration above, but across the network instead of needing to be on the Raspberry Pi. SSH is an encrypted protocol so that it is not possible for others on the network to see what commands are being issued and more importantly the username and password used to login. You can get an ssh client for most computers, including tablets and smart phones.

What is a server?

You may think of a server as being a computer sat in a dedicated data centre hosting a website. Whilst that can be the case, any computer can actually act as a server for certain functions, even a desktop computer or a mobile phone. When we talk about a server it is whichever computer is running the code that allows a client (the opposite of a server) to connect to it. The Raspbian operating system includes the OpenSSH server which is enabled by default to allow a client to connect to the Raspberry Pi.

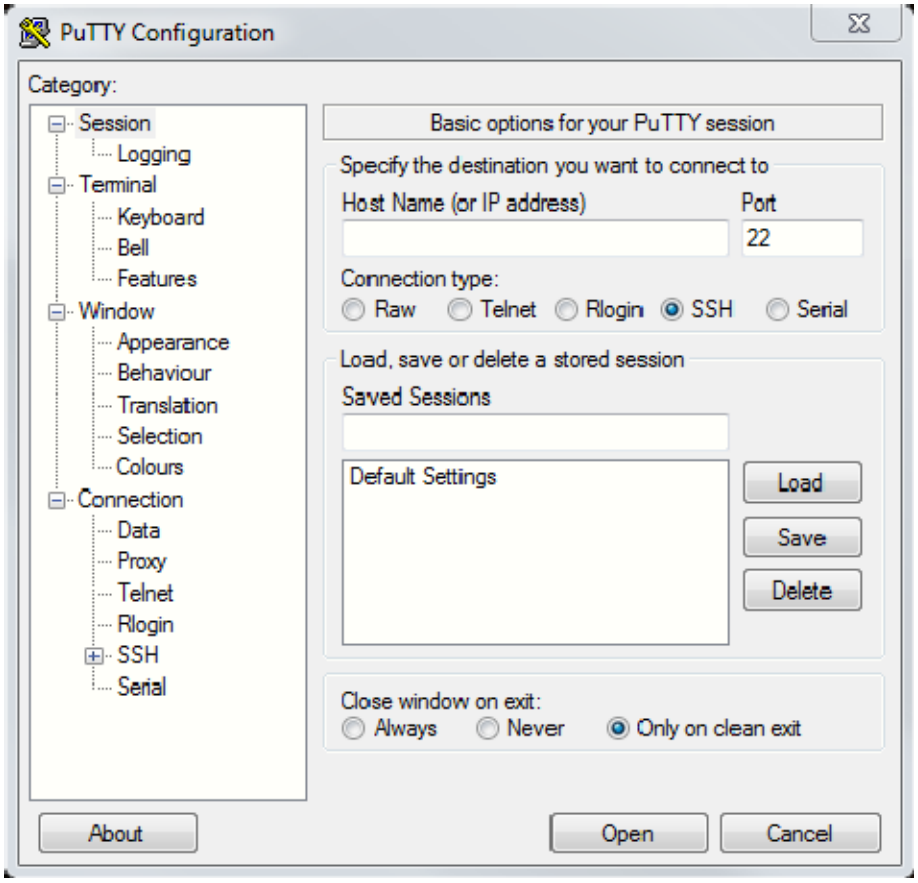
SSH has a lot more functionality than we are using here. SSH can be configured to use key based authentication providing a secure way of connecting from one computer to another without needing to login manually (which is useful for automation). SSH can also be used to create tunnels to direct network traffic over an encr ypted session over an insecure network. For more details see: http://www.openssh.com/

U***sing SSH from a Windows computer***

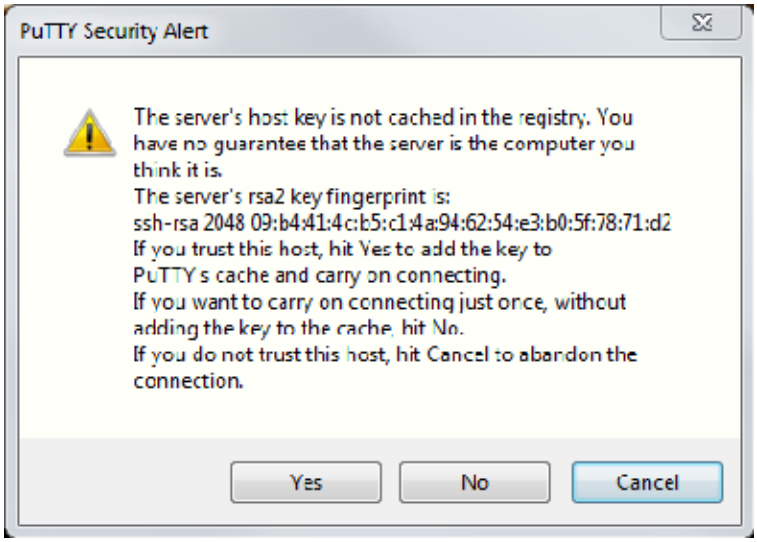
Unlike other operating systems Microsoft Windows does not normally include an ssh client. There are a number of free ssh clients for Windows. PuTTY is one of the most popular clients. It is a graphical application, but still provides the same text based interface that is used by the standard OpenSSH client used in Linux.

You can download it from: http://www.chiark.greenend.org.uk/~sgtatham/putty/

You can either download the executable file putty.exe, which can be saved onto your local computer, or as an installer which includes a file transfer client and other tools. When you run PuTTY you will be presented with the following startup screen. You can enter the IP address of the Raspberry Pi 10.5.5.1 in “Host Name (or IP address)”. The port can be left at the default of 22 and then click open



This is the first time PuTTY has connected to this computer and so it provides a security alert with some details about the computer so you can decide if that is the correct computer. If you want to check you are connecting to the correct computer then you can run the “ssh-keygen -l” command when physically connected to the Raspberry Pi. You will need to provide the appropriate file where SSH has stored the key. In reality if this is your local network and you know this is a computer you have not connected to before then you can just accept the connection using 'Yes'. If in future you get a warning that the key has changed then you should take extra care in case someone is masquerading as your computer, although if you have re-installed the computer that will also generate the same warning.



You will now be presented with a login prompt where you can login with the username and password for the Raspberry Pi. After logging on any commands entered will be run on the remote computer "Pibot" and not the computer that I am typing on. When using ssh get used to looking at the computer name on the command prompt before issuing a command to make sure you are on the right computer first.

Adding the Raspberry Pi to the robot chassis

We now have the robot chassis built and Raspberry Pi configured so that we can connect to it over the network; it's now time to put the two together. The Raspberry Pi can be installed anywhere on the chassis. The key things to consider are the length of the camera cable (if using a Raspberry Pi camera) and the cable from the GPIO to the breadboard. In the robot vehicle shown the Raspberry Pi is at the front (or at least what I designated as the front) so that it could use the standard cables. If you would rather install it elsewhere then you may need to get longer cables. There are two holes on the Raspberry Pi board that be used to secure it using the PCB stands provided with the robot chassis.

Power for the Raspberry Pi and motors

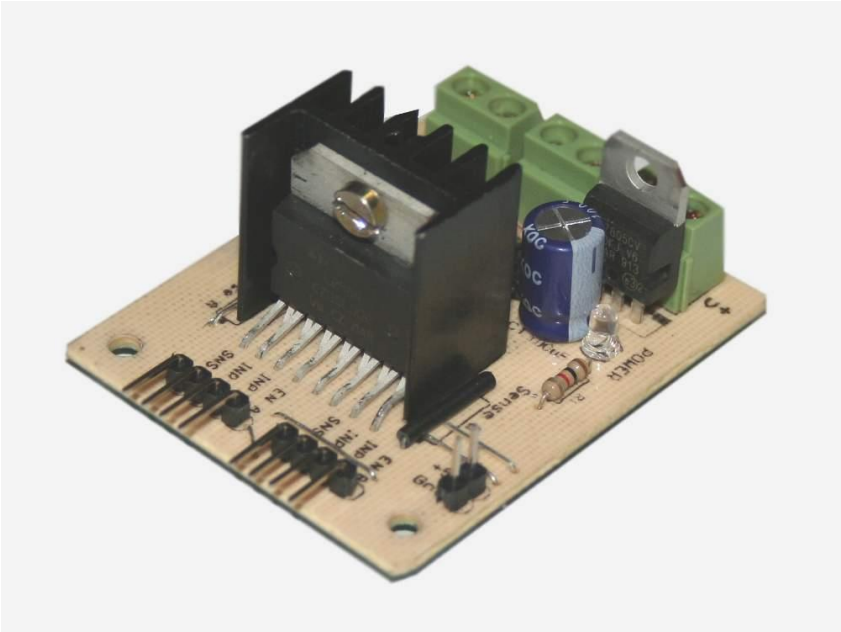
The Raspberry Pi needs a power supply of 5V. This is the standard supply from a USB socket and for the power used for most modern mobile phones. The tolerance of the supply is ± 0.25V which means that the power supply should be between 4.75V and 5.25V. If the supply falls below 4.75V then it is likely to have problems, typically the USB devices stop working first, but other problems may occur (such as unexpected reboots). More than 5.25V and there is a risk of damage to the Raspberry Pi. The power supply needs to be able to supply a running current of around 300mA for the model A or 700mA if using a model B. This is the main reason that the model A is better suited for this project, to reduce the load on the power supply. The motors supplied with the chassis are designed for a maximum of 6V. If you factor in the voltage drop within the electronic circuit then ideally it should have a higher power supply. Whilst we haven't decided on how to switch the motors yet you could consider about 1 volt voltage drop in the circuit. So we should therefore be looking for a power supply of about 7V, which is required to run the motors at full power. Through testing I have found that the particular motors I have will run on a 5V supply even with our control circuit. If using 5V the motors will be running with less power than if the supply to the motors is 6V, but in tests it was shown to be sufficient for moving the robot around on a relatively flat surface.

L298N Dual H-Bridge Motor Driver

The L298N Dual H-Bridge Motor Driver is a great value and can be used with a variety of robot controllers. It features a powerful L298N motor driver module with a heavy duty heat sink. It is powerful enough to drive motors from 5-35V at up to 2A peak.

An onboard 5V regulator is provided that can be used to power other parts of your robot's circuitry such as an Arduino microcontroller.

Usage



Follow the steps below to configure the motor controller board to work as a typical robot motor driver for use with two DC motors.

1. Attach your robot's motors to the green Motor A and Motor B screw terminals.
2. Connect the ENA and ENB to PWM capable digital outputs on your robot's microcontroller.
3. Connect the IN1, 2, 3 and 4 pins to any digital outputs your robot's microcontroller.
4. Apply 5-16V to the board by connecting positive (+) to the blue VMS screw terminal and ground (-) to the blue GND screw terminal.
5. See below for details on controlling the motors with your robot's microcontroller.

All inputs are TTL compatible. Do not enable the onboard 5V regulator if you plan to supply more than 16V to your motors. Refer to the details below

Hardware Details

Screw	Terminal	Pin	Assignments
Pin	**Color**	**Name**	**Description**
1	Green	Motor A -	Output to Motor A (-)
2	Green	Motor A+	Output to Motor A (+)
3	Blue	VMS	Input 4-35V motor power supply (+)
4	Blue	GND	Ground (-)
5	Blue	5V	5V regulated power (+)
6	Green	Motor B -	Output to Motor B (-)
7	Green	Motor B+	Output to Motor B (+)

Note that the 5V regulated power on pin 5 above is an output when the 5V_EN jumper is in place. Otherwise you must input 5V regulated power at pin 5 so that the circuit can operate properly. Do not enable the onboard 5V regulator if you are supplying more than 16V to motors on pin 3 or the regulator will burn out.

Header pin assignments		
Pin	**Name**	**Description**
1	ENA	Input to enable Motor A
2	IN1	Input to control Motor A
3	IN1	Input to control Motor A
4	ENA	Input to enable Motor B
5	IN1	Input to control Motor B
6	IN2	Input to control Motor B

Jumpers	
Name	**Description**
5V_EN	Enable the onboard 5V regulator
U1	Enable Motor A input pin IN2 pull-up resistor (10K)
U2	Enable Motor A input pin IN2 pull-up resistor (10K)
U3	Enable Motor B input pin IN3 pull-up resistor (10K)
U4	Enable Motor B input pin IN4 pull-up resistor (10K)
CSA	Ties the Motor A current sense to ground
CSB	Ties the Motor B current sense to ground

Note: The CSA and CSB current sense feature is disabled when the jumpers are present. To use the current sense feature, remove the jumpers and attach to the header pins. Leave the jumper connected when not using current sense.

Software

Speed Control

The speed of the motors can be adjusted by connecting PWM outputs from your robot's microcontroller to the ENA and ENB input pins on the motor driver board. The ENA pin controls Motor A and the ENB pin controls Motor B. When these pins are HIGH, power is output to the motor. By using PWM, your are turning power on and off very quickly to adjust the speed of the motor. The longer the PWM duty cycle is, the faster the motor will turn. We recommend always using a PWM duty cycle of 90% or less.

Direction Control

The direction that the motors turn is controlled using the IN1, IN2, IN3 and IN4 input pins on the motor driver board. Connect these pins to digital outputs on your robots microcontroller. To make Motor A go forward, set IN1=HIGH and IN2=LOW. To make Motor A go backward set IN1=LOW and IN2=HIGH. The same method is used to control Motor B: set IN3=HIGH and IN4=LOW to o forward and set IN3=LOW and IN4=HIGH to go backwards. Note that "forward" and "backwards" refer to the direction of the motors themselves. If your robot does not move in the expected direction, reverse the motor polarity by swapping the green screw terminals for Motor A + and – and/or Motor B + and -.

Stopping

To remove power from the motors, simply set ENA=LOW for Motor A and ENB=LOW for Motor B. This will result in the motors stopping slowly and naturally from friction. To perform a quick braking operation, set both IN1=LOW and IN2=LOW for Motor A and IN3=LOW and IN4=LOW for Motor B. The motors will come to an instant stop.

Motor Driver Truth Tables

Here are some handy tables to show the various modes of operation.

Motor A truth			
ENA	**IN1**	**IN2**	**Description**
0	N/A	N/A	Motor A is off
1	0	0	Motor A is off (floating)
1	0	1	Motor A is on and turning backwards
1	1	0	Motor A is on and turning forwards
1	1	1	Motor A is stopped (brakes)

Motor B truth table			
ENB	**IN3**	**IN4**	**Description**
0	N/A	N/A	Motor B is off
1	0	0	Motor B is off (floating)
1	0	1	Motor B is on and turning backwards
1	1	0	Motor B is on and turning forwards
1	1	1	Motor B is stopped (brakes)

Programming with python

Forward.py:

```
import RPi.GPIO as gpio
gpio.setmode(gpio.BOARD)
gpio.setup(7,gpio.OUT)
gpio.output(7,1)
gpio.output(11,0)
gpio.output(13,1)
gpio.output(15,0)
```

Backward.py:

```
import RPi.GPIO as gpio
gpio.setmode(gpio.BOARD)
gpio.setup(7,gpio.OUT)
gpio.output(7,0)
gpio.output(11,1)
gpio.output(13,0)
gpio.output(15,1)
```

Left.py:

```
import RPi.GPIO as gpio
gpio.setmode(gpio.BOARD)
gpio.setup(7,gpio.OUT)
gpio.output(7,1)
```

```
gpio.output(11,1)
gpio.output(13,0)
gpio.output(15,1)
```

Right.py:

```
import RPi.GPIO as gpio
gpio.setmode(gpio.BOARD)
gpio.setup(7,gpio.OUT)
gpio.output(7,1)
gpio.output(11,1)
gpio.output(13,0)
gpio.output(15,1)
```

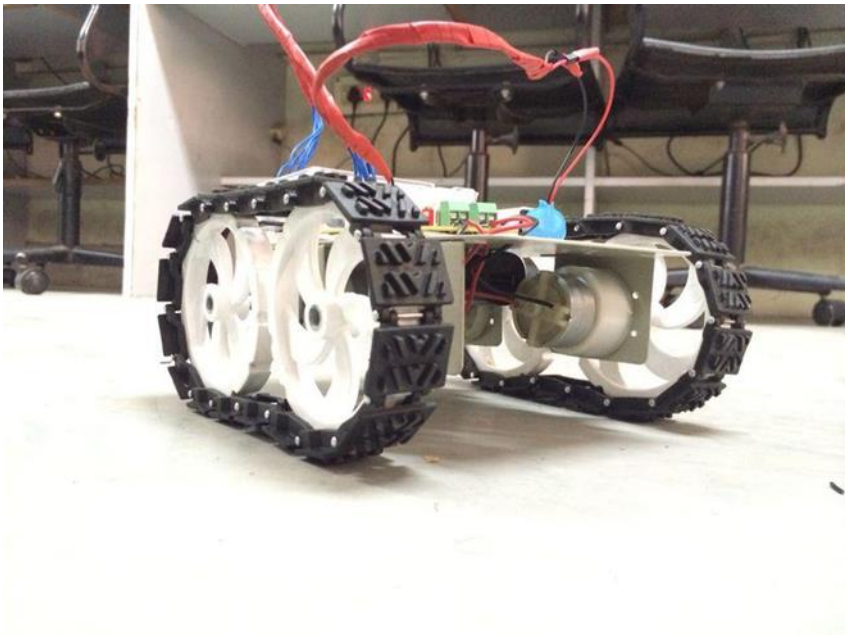
Stop.py:

```
import RPi.GPIO as gpio
gpio.setmode(gpio.BOARD)
gpio.setup(7,gpio.OUT)
gpio.output(7,0)
gpio.output(11,0)
gpio.output(13,0)
gpio.output(15,0)
```

Testing the robot

It should now be possible to connect to the robot using a mobile phone, tablet or computer using the wireless network and an ssh client. Then cd to the directory with the program stored and run the program with

```
sudo ./forward.py
sudo ./backward.py
sudo ./left.py
sudo ./right.py
sudo ./stop.py
```



Software requirements specification for Pibot

1.0 Introduction

The software created is to provide the control of a robot vehicle based around the Raspberry Pi bot. This software is intended as a demonstration of how the vehicle can be controlled, it can then be used as a basis for future development of a program with improved functionality. The software will allow a user to control the robot vehicle using key-presses and will control the motors appropriately. It should allow change in speed and direction.

1.1 System overview

The system comprises of a Raspberry Pi (model A or B) running Raspbian Linux. The GPIO is connected to a L2989D motor controller board.

2. 0 Overall description

The application should be a standalone application that can be used by the user to control the vehicle.

2.1 System interfaces

The software must interface with the Raspbian operating system.

2.2 User interfaces

The program should be text based so that it can be run in a terminal window. The users will

interact with the software through key presses within the terminal application.

2.3 Hardware interfaces

The GPIO ports connect to the L2989D motor controller [a link could be provided showing details of the motor controller in this case linking to elsewhere in this document]. The GPIO pins are connected directly to the output pins on the Raspberry Pi processor.

2.4 Software interfaces

The software is not required to interface with any other software for this version. The software should not prevent the Raspberry Pi camera function's from being used outside of the application. [A future development may be to interface with the Raspberry Pi camera software, but it is not included in this version]

2.5 Communication interfaces

No direct network communication is required, but the software must be able to work from within a remote ssh terminal.

2.6 Memory constraints

The application must be able to run on a Model B Raspberry Pi with 512Mb of memory. This includes other applications that may be running including the Raspberry Pi camera applications.

3.0 Specific requirements

3.1 Performance requirements

The software should be responsive to user interactions to allow immediate control of the vehicle. A significant delay (more than 0.5secs) will make control very difficult.

3.2 Software system attributes

There are no specific reliability concerns in this version. This is intended as a demonstration only and so in the event of a control failure there is only minimal risk of damage to itself or other objects. The electronic hardware already incorporates protection to prevent invalid states causing a short circuit. [See the H-Bridge section for details of how this could be a potential risk of causing a short circuit.]

As this is not network based there is no security restrictions required to prevent unauthorized access. If using ssh then the security credentials of the network configuration and the ssh authentication are required to protect from unauthorized use.

3.3 Other requirements

This software is designed as a teaching aid to accompany the Pibot project. It should therefore be written so that those learning programming can understand how this works.