Let's Make Robots

ALL LMR ARE BELONG TO US!

How to build your first Robot !

Full, step by step Instructions

How to make your first Robot

Let's Make Robots

ALL LMR ARE BELONG TO US!

By Frits “fritsl” Lyneborg

http://letsmakerobots.com/user/4

PDF by servello

For more info, latest version and where to find the materials needed:
http://letsmakerobots.com/start

This is an updated version, the original can be found here: The original instructions

This tutorial is the easiest way in the world to give you a fast start into building robots.

Learn how to make a robot, no skills needed, very, very easy.

Time to build: 2 hours
URL to more information: http://picasaweb.google.com/fritslyneborg/Let...
Cost to build: $85

- Actuators / output devices: 2 geared motors
- CPU: Picaxe
- Power source: 4 AA batteries
- Programming language: Picaxe basic
- Sensors / input devices: Sharp IR
- Target environment: indoor

Welcome :)

Everything here is so easy, that after you have gone through it, you can make a robot in a couple of hours. Why can't you do that now?

Because there are so many little things you need to know. This is an attempt to let you know exactly all these little things, and nothing more. Fast, and based on 2 years of experience of what people need to know to get started. If you hurry, you can run through this, and be robot builder in a couple of hours. But expect to use a good weekend - Learning takes time - even though it is very easy, it just takes some time, all the little things to get to know :)

There are other "How to get started building robots" out there. This one is focusing on getting you around everything extremely fast. You need no knowledge of ... anything. And you will learn everything... well, the basics of everything ;)

All images in high-res here.

Attachment: shbotvideoprogram.bas (5.43 KB)

Resources:

PICAXE Manual Part 1 - Getting Started (.pdf)	403KB
PICAXE Manual Part 2 - BASIC Commands (.pdf)	405KB
PICAXE Manual Part 3 - Interfacing Circuits (.pdf)	309KB
PICAXE Editor Software v5	

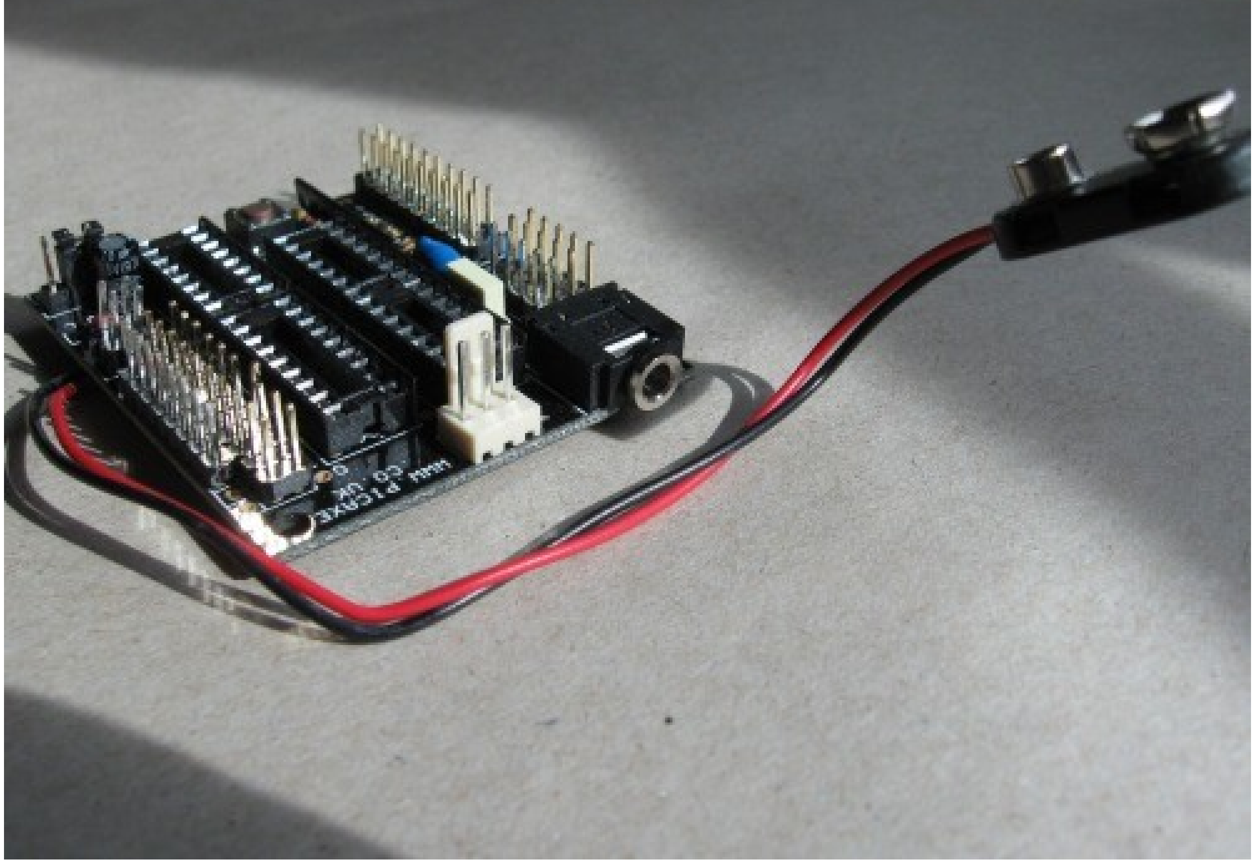
If you are ready to begin, let's start with . . .

Materials needed

Everything you need can be found in webshops, and via Google, and you can get it where you are, in any country.

1 PICAXE-28 Project Board

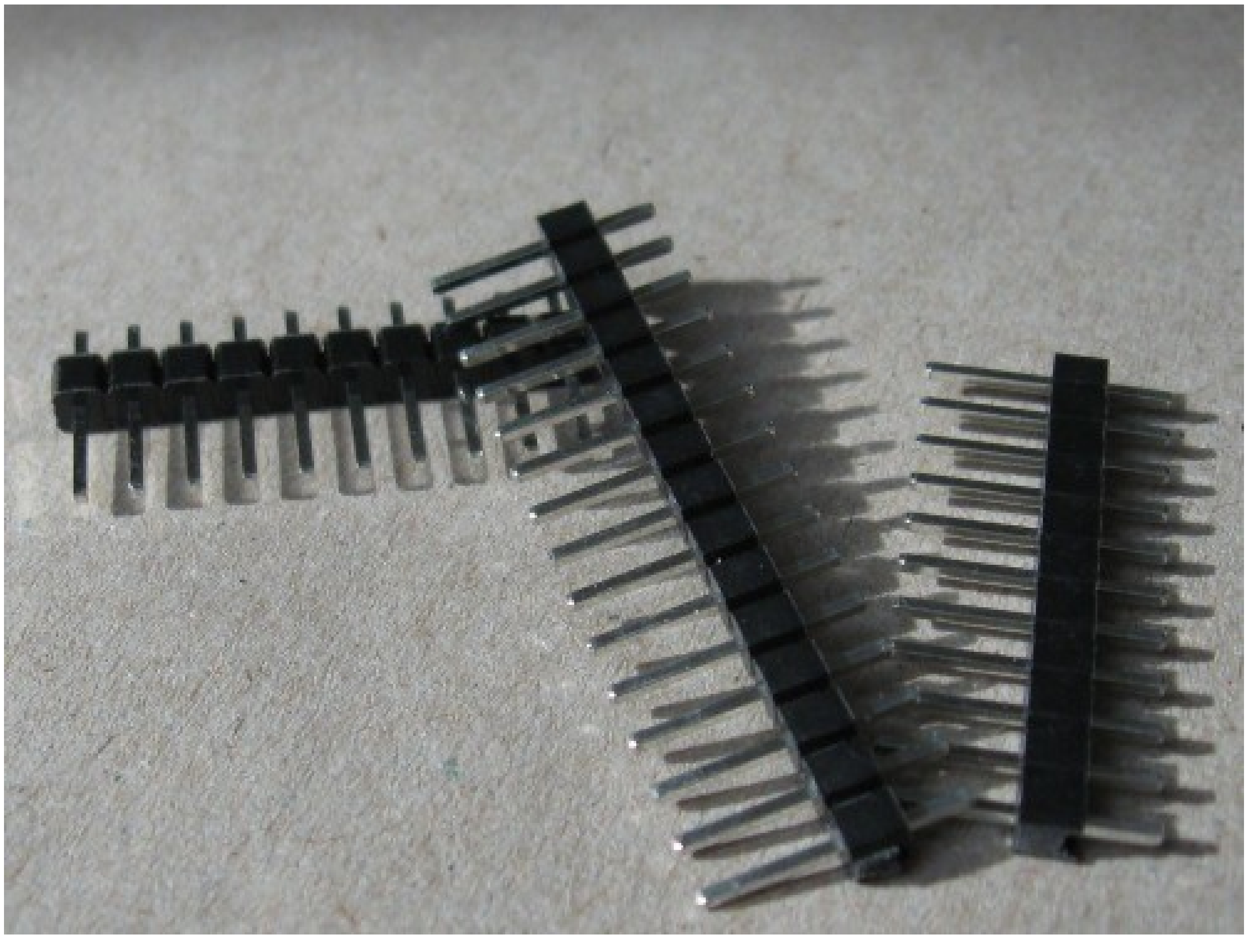
The 28 pin project board is like a game of Mario Bros; Fun and full of extras and hidden features, making you want to play over and again. It is an extremely good board to get you started, and it can be used for a fantacillion different projects, don't get me started :)



Male "snap off" Header Pins, at least 10 pins on a strip

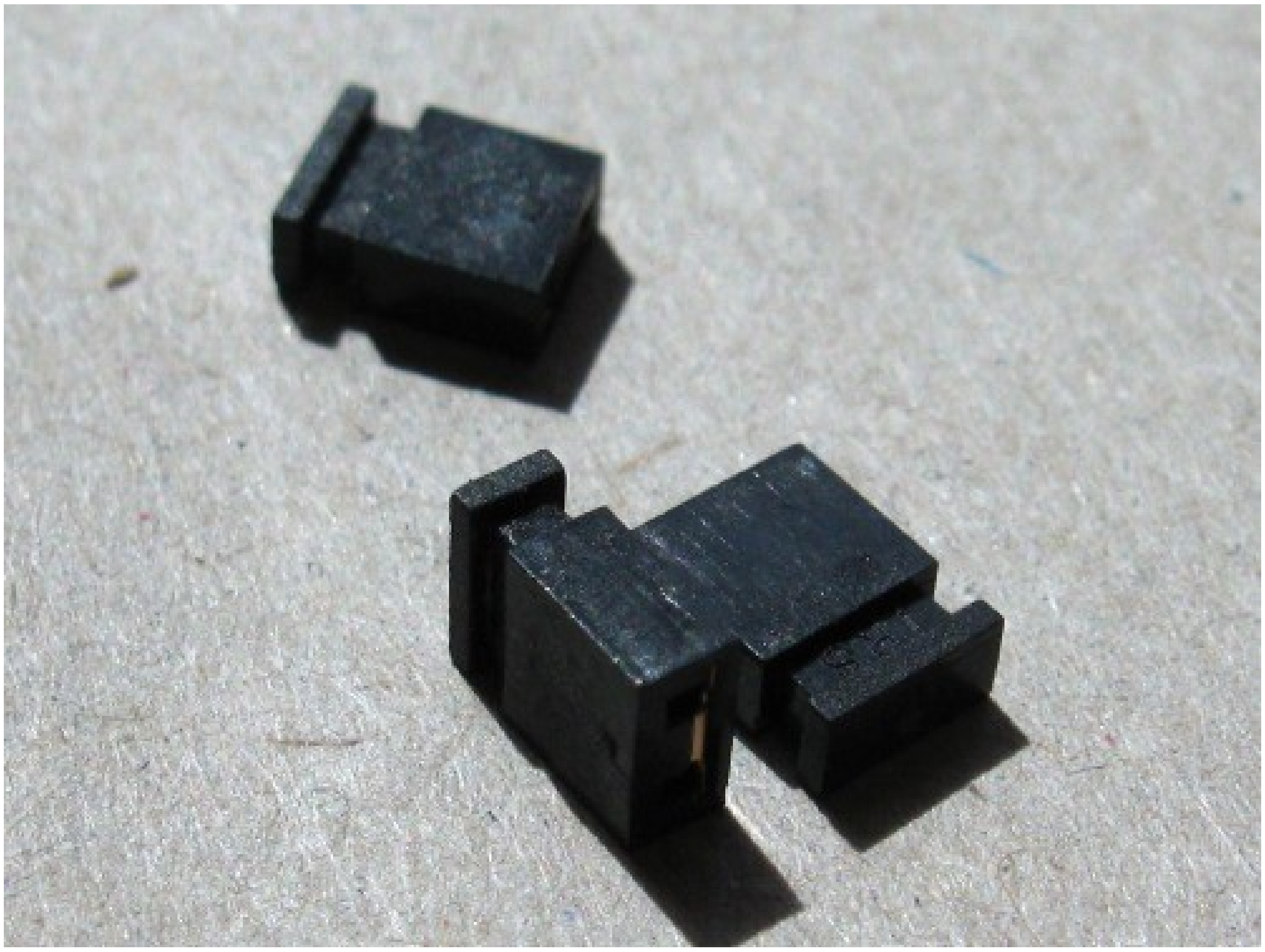
Many times the boards that you buy just have holes in them, and that makes it hard to plug something in / on. One way to overcome this, is to solder wires into the holes. Another is to add these pins, so you can plug on wires, like with the servo and female headers shown below. "Why don't they just put pins in all the holes from the factory", you may ask. Well, I don't know. Maybe to give us the option. It is also possible to solder female headers onto the board, perhaps this is why.

You get these in long rows, and simply break them apart with your fingers.



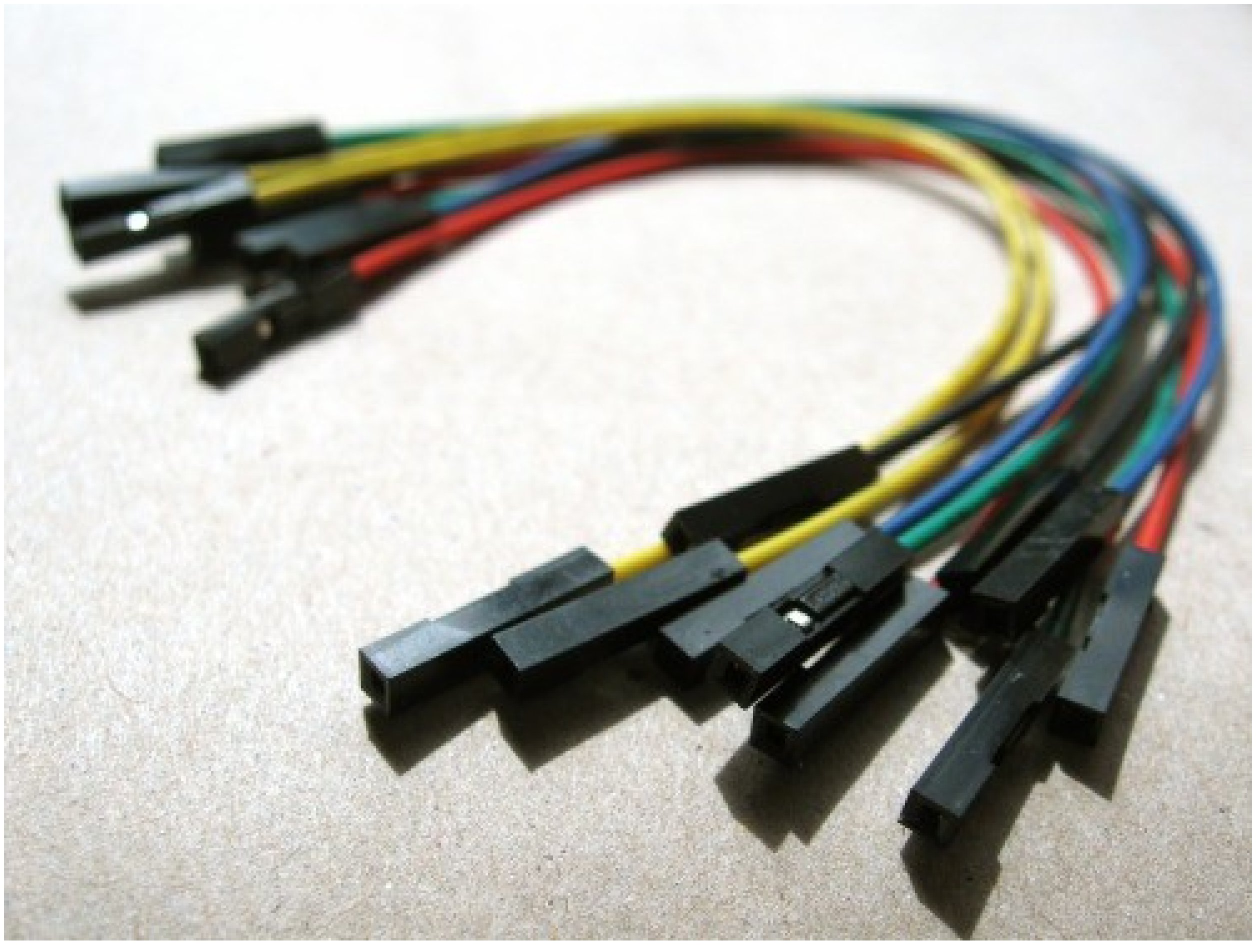
3 Shorting Blocks, Top Closed

Put these over 2 pins next to each other, and there is a connection between them!



5 or more Female-Female Header Jumper cables

Yes. These are nice. When I started this hobby a couple of years ago, these where really hard to get. Now they are everywhere, and that is really good. Most things in this new robot-hobby of yours have pins (or you solder some in ;) - and by using these jumpers, you can make quick connections without soldering. Nice!



1 USB PICAXE Programming Cable

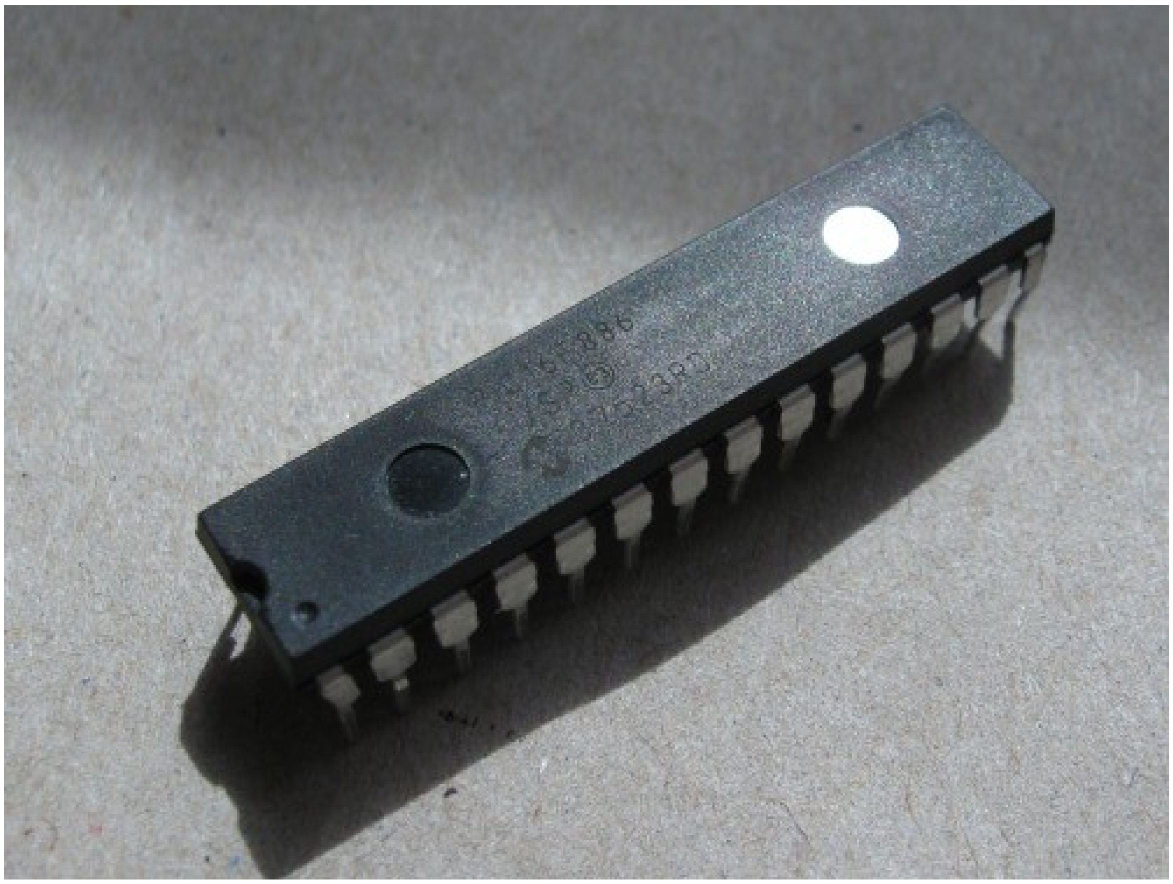
You write your robots programs on your computer. Plug this cable into the robot, and transfer the program. Unplug, and the robot runs the program by itself.

1 PICAXE-28X1 IC

This chip is a Microprocessor. That is often explained as "A computer in a chip". It can be placed in the board described above, after that, it can be programmed from your computer via the programming cable.

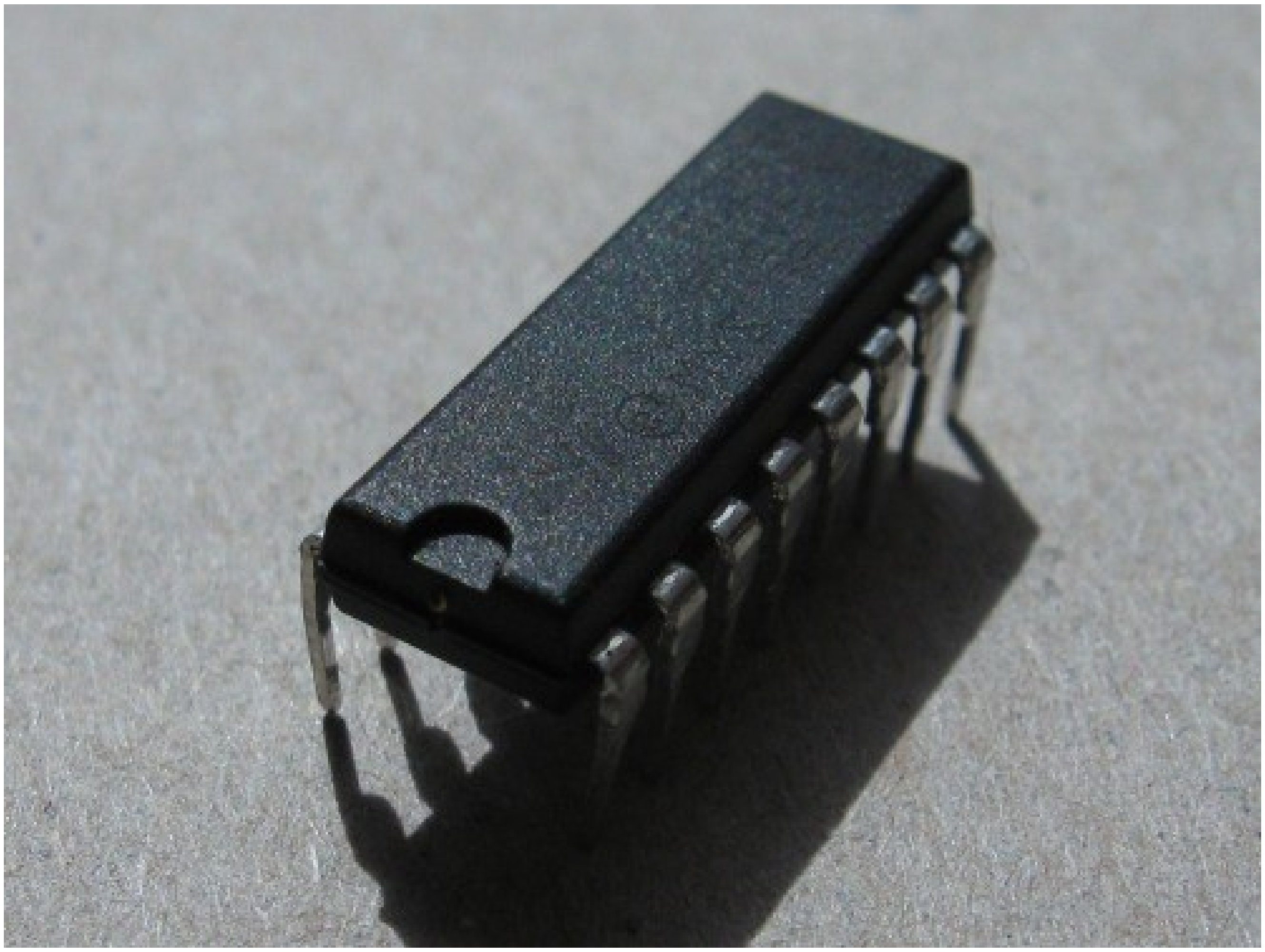
Your program can tell the controller to "listen for inputs", "think about them", perhaps make some calculations or look in some datas, and make outputs to something like the motor driver below.

It is chosen here, because it is quite strong, yet very easy to program, as you will see below.



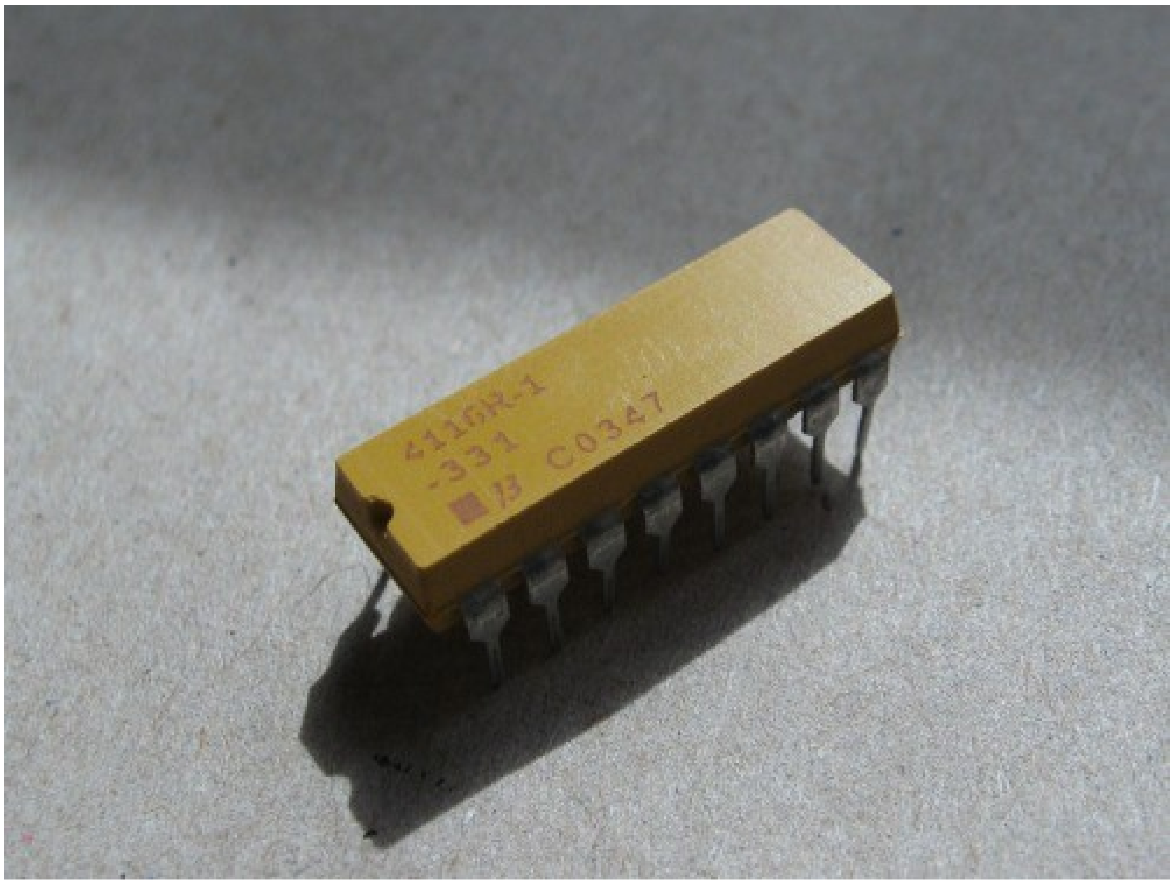
1 L293D Motor Driver IC

I will describe this later, when we install it below :)



1 DIL 330 x 8 resistor array

The Yellow chip! It is very dull, just a row of little resistors. You will be using it to set your board up for servos.



1 Standard servo

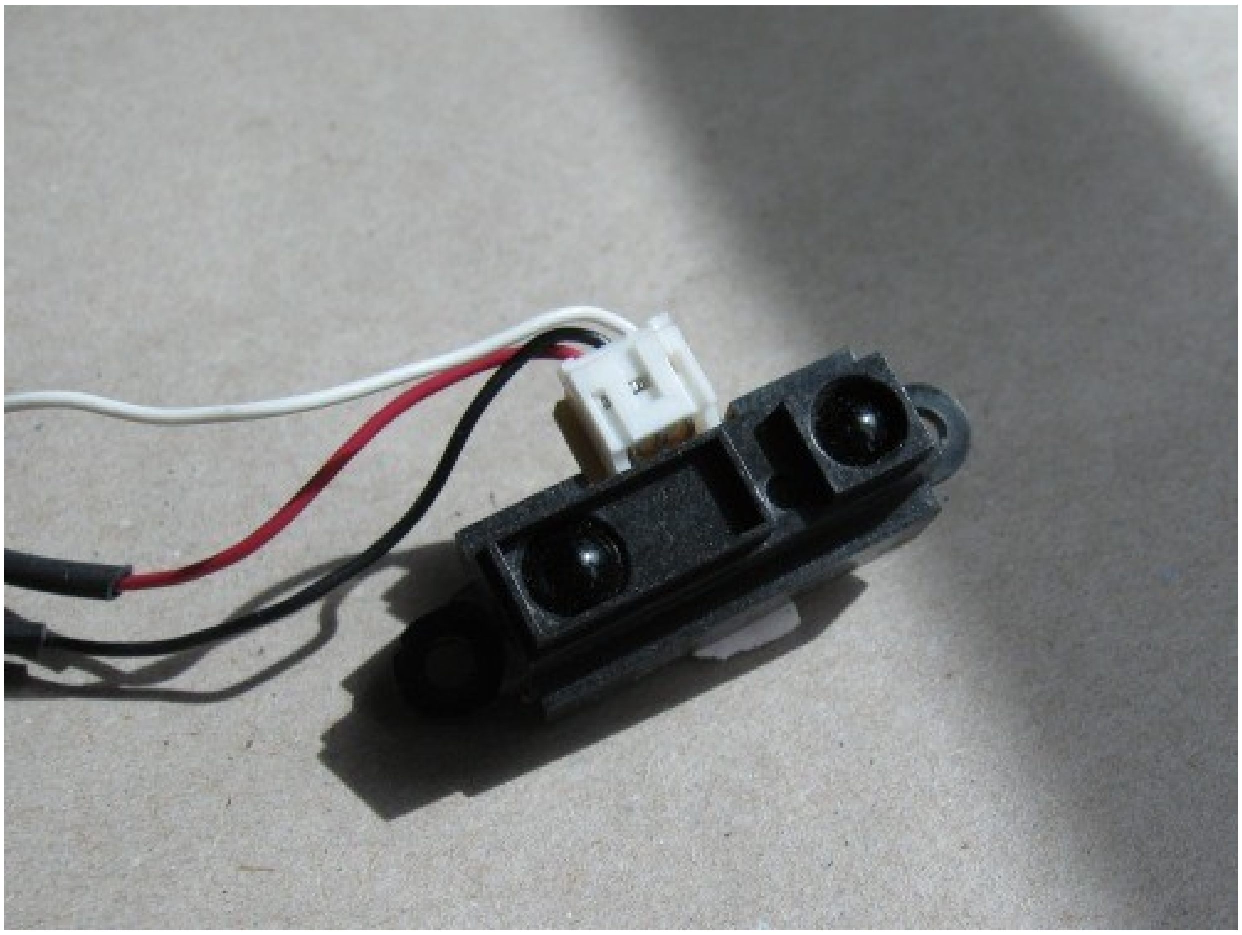
A Servo is a cornerstone in most robotic appliances. To put it short it is a little box with wires to it, and a shaft that can turn some 200 degrees, from side to side.

The microcontroller can decide to where the shaft should turn, and stay there. Like go to "3 o'clock". That is it pretty handy; You can program something to physically move to a certain position. Next thing (after this project) could be to let one servo lift another servo. You would then have what is referred to as 2 DOF ("Degrees of Freedom"). But let's start with one ;)

You may wonder why my servo has that white pin, where yours might have a flat disc, a cross, or something. It does not matter, servos comes with all kinds of "servo horns". We just need something there to glue the head on to!

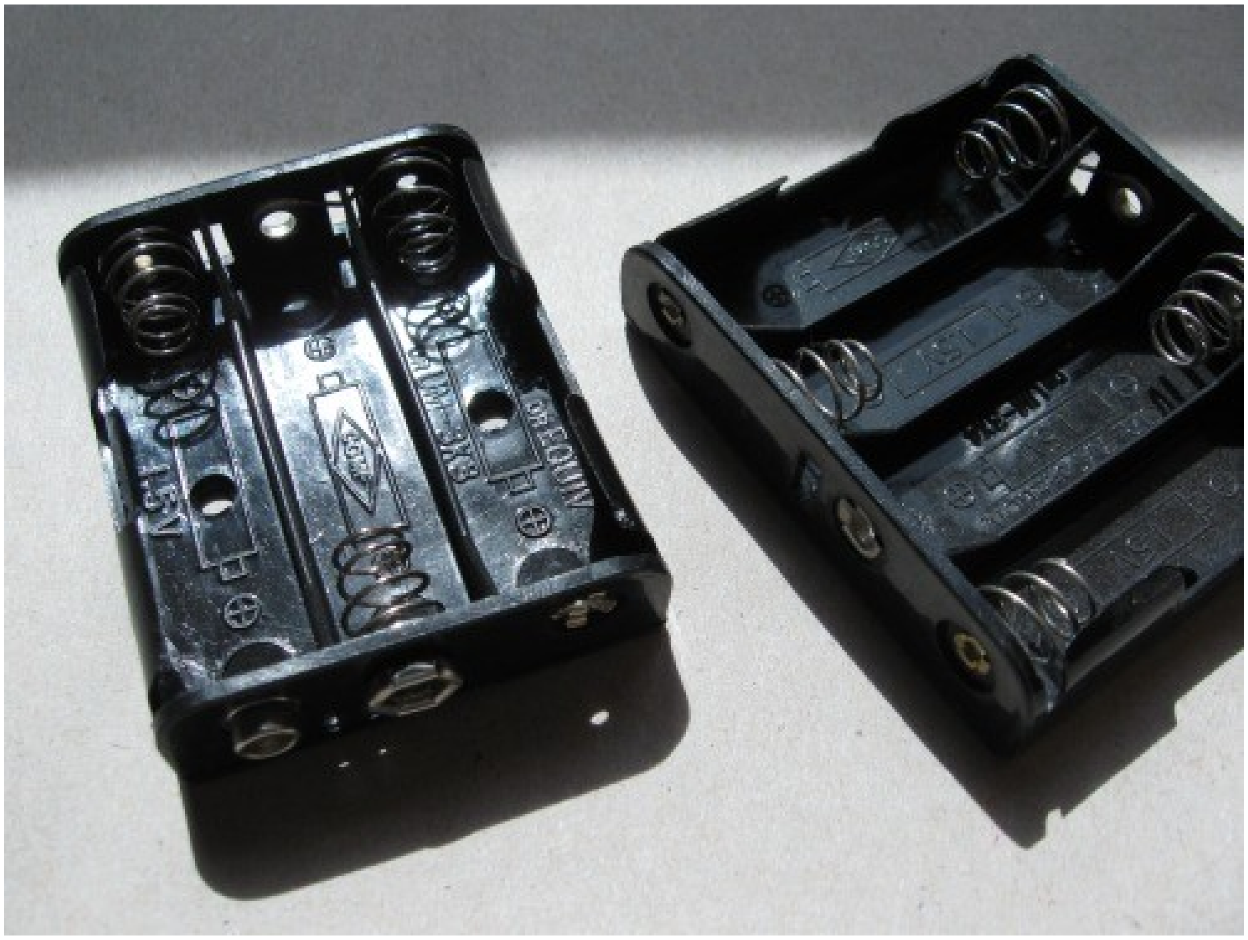
1 Sharp Analogue InfraRed Range Finding System (AERS) with cable

The one "eye" sends infra red light. The other sees the reflection of this (if there is one), and the unit returns the distance to the object in front of it. It has 3 wires (make sure you get the cable for it, or it can be a little hard to hook up). You give it power on 2 of the wires, and the third one plugs into the microcontroller, and tells it the distance.



1 4 x AA Battery Holder if you are using rechargeable batteries
- or -
1 3 x AA Battery Holder if you are using non-rechargeable batteries

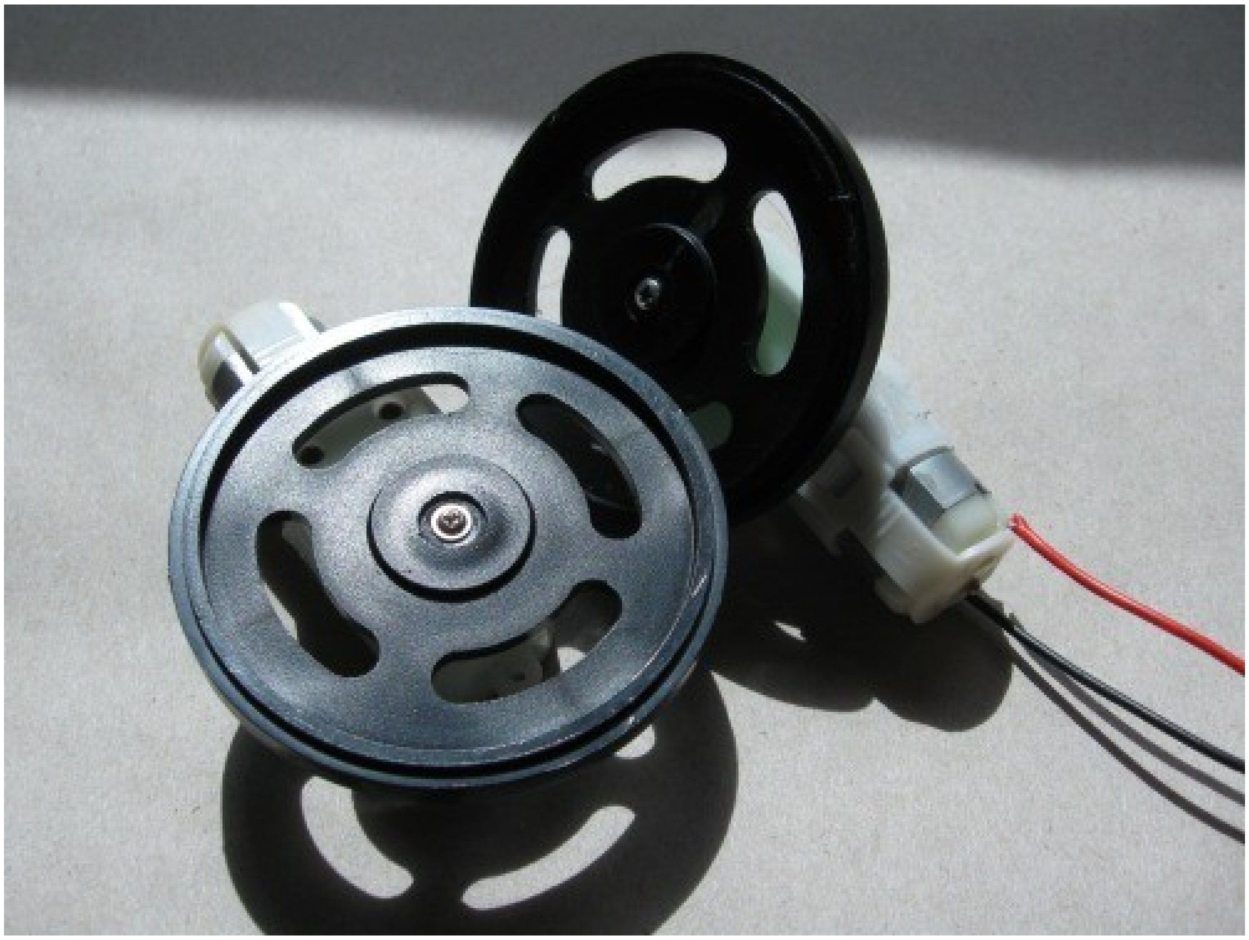
(See more below, regarding batteries, and why the difference - Point is that you need as close to 5V as possible, one way or the other, and you can use something completely different in terns of batteries if you want. As long as it is just about 5 Volts.)



2 Geared motors and wheels to fit

It is very important that your motors have gears. You want a slow robot; Go for high ratios, like 120:1 or higher, as a slow robot is so much more fun in the beginning, because you can see what it is doing.

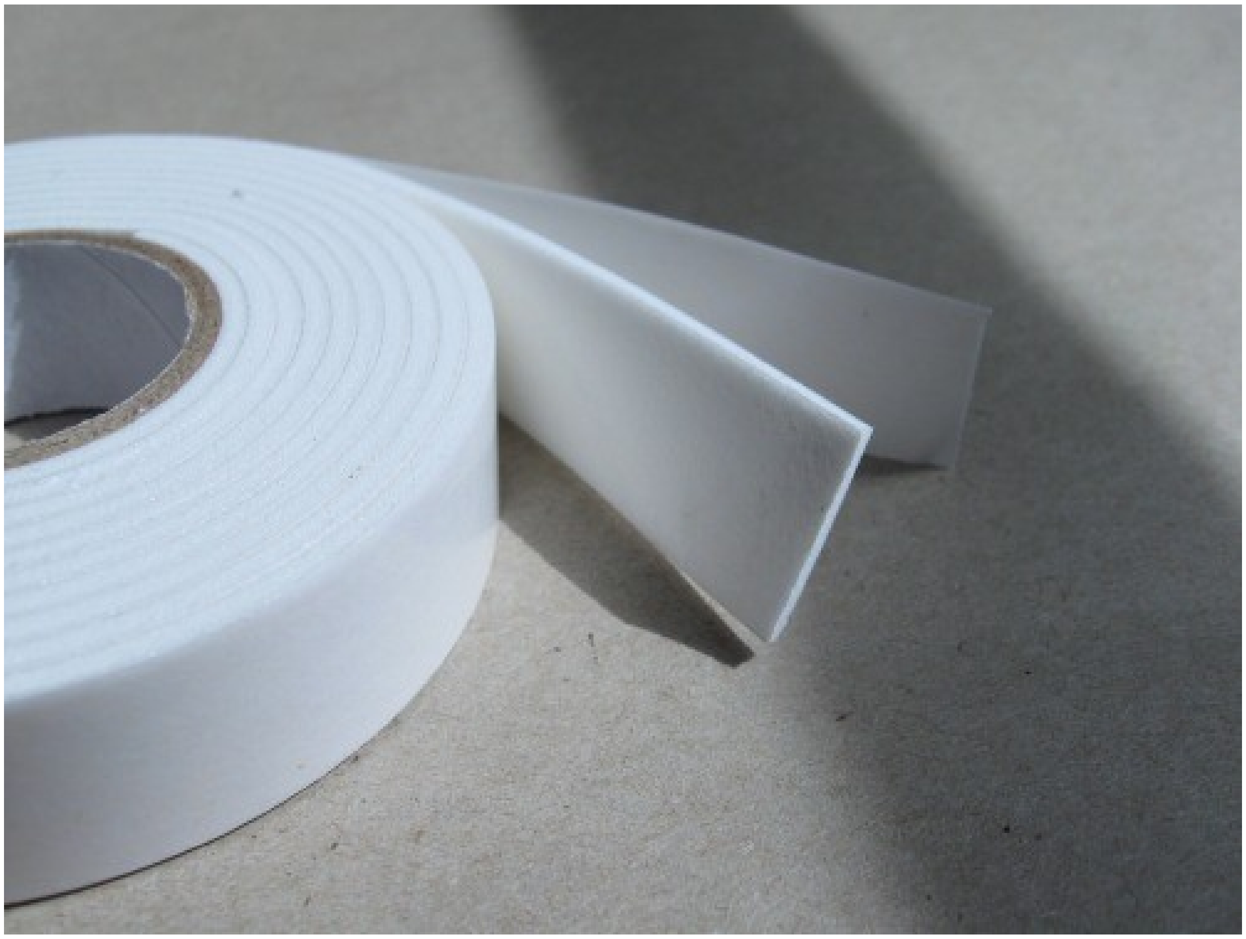
Apart from that, there is not much to say. Well, that would be, that there are many ways of moving and steering. This way of only using 2 wheels, is sometimes referred to as "skid steering". And it is worth remarking that if you'd like to add belt tracks later on, the basics are the same ;)



1 Roll of double sided foam tape

Oh yes! If there is something you cannot fix with this tape, it is because you are not using enough! It is a very, very fast way of sticking 2 items together. In fact we will be using it to make this entire robot! Depending on the make, of course, it is also reasonably easy to take apart again.

Paint stirring sticks, this tape, and a melt glue gun is the backbone for most of my fun with robots :)



1 Heat shrink tube (5 mm approx)

Sometimes you do need to solder 2 wires together. For instance the Sharp IR Range finder; It comes with straight up wires on the plug. What you do, is cut one of the female cables (above) in 2 parts, solder them together.. but before that, you cut a little piece of this heat shrink tube to slide over the place without insulation. Then with a lighter, you can quickly heat up the tube, and it shrinks to fit.

That is so much smarter than using tape ;)

Also needed:

Batteries

Either 3 AA Non rechargeable, or 4 AA Rechargeable.

This robot needs 5 Volts. Mainly because the Sharp IR, really feels best on 5.0V, that's what it's made for. Motors and servo would like more, microcontroller could live with 6.0V, but keeping it simple is the core here, so we feed the whole robot with as close to 5.0V as possible. And rather too little than too much, so we make sure not to fry anything, now that this is your first robot ;)

Now, you may know, that normal batteries provide 1.5V. However, you may not know that rechargeable batteries only provide 1.2V!

No matter if you knew that or not, 3 times 1.5V from normal batteries, is 4.5V. If we use 4 times 1.5V we would get 6.0V, which might be a little scary to use on the Sharp, rated for 5.0V.

4 times 1.2V from rechargeables is 4.8V, which is nice and close to 5V. And then it is much cheaper in the long run. So I strongly recommend you to get some rechargeables and a charger.

Tip: The best rechargeables have the highest capacity, measured in "mAh". The 2500 mAh AA-size is a fine battery.

A Soldering iron and solder

If you are new to soldering, this might interest you.

A lighter and a cutter

Lighter for heat shrinking, cutter to.. cut.

Tip: If you want to use the cutter to remove plastic from cables, turn it this way; Imagine that you where sticking the cable right into the cutter from where you are now, into the table it is laying on. That way. And not from the table, and out to where you are. Then gently close around the wire, and pull the plastic off.

A computer with an internet connection and a free USB port

Can be Mac, Linux or PC. The software needed for this is free.

Nice-to-have tools, though not essential:

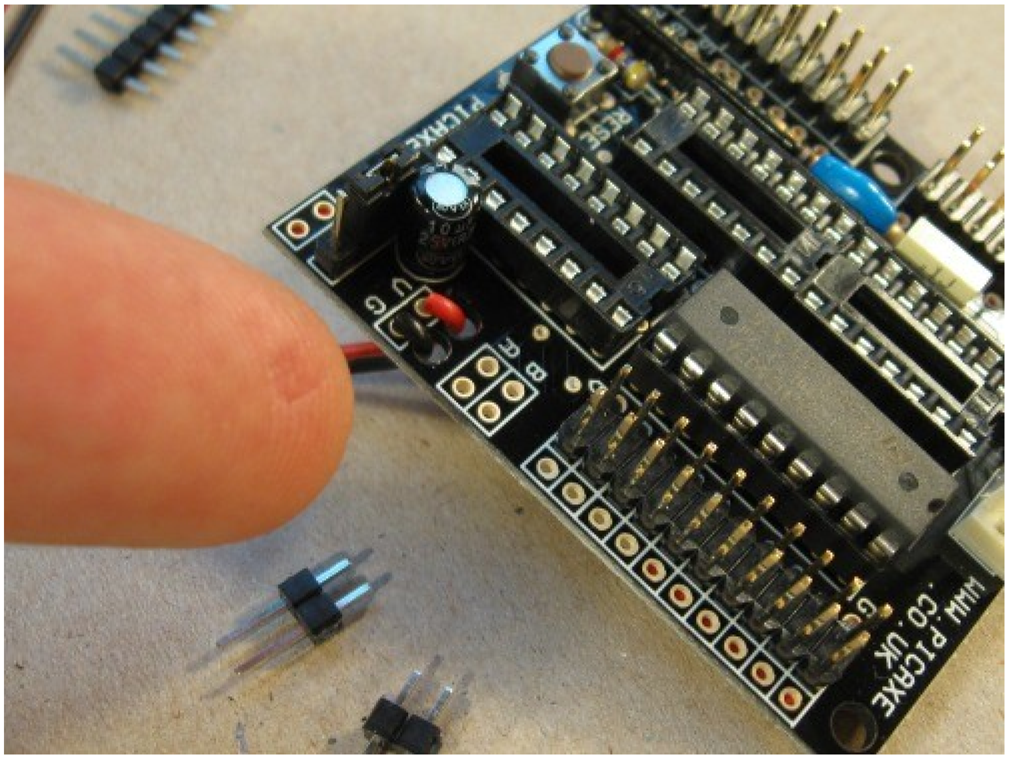
A multimeter (aka measure-thingey), a wire stripper, and a screwdriver

Ready? Let's make a robot :)

Pins in holes!

First unwrap your board (I am sure you already did that :), and then see that it may have some red stuff underneath. That is just there from when they made it; They insert the components on the upper side, and dip the boards lower side into hot solder.. and areas where they don't want solder to be stuck, they have placed that red stuff. So just take it off :)

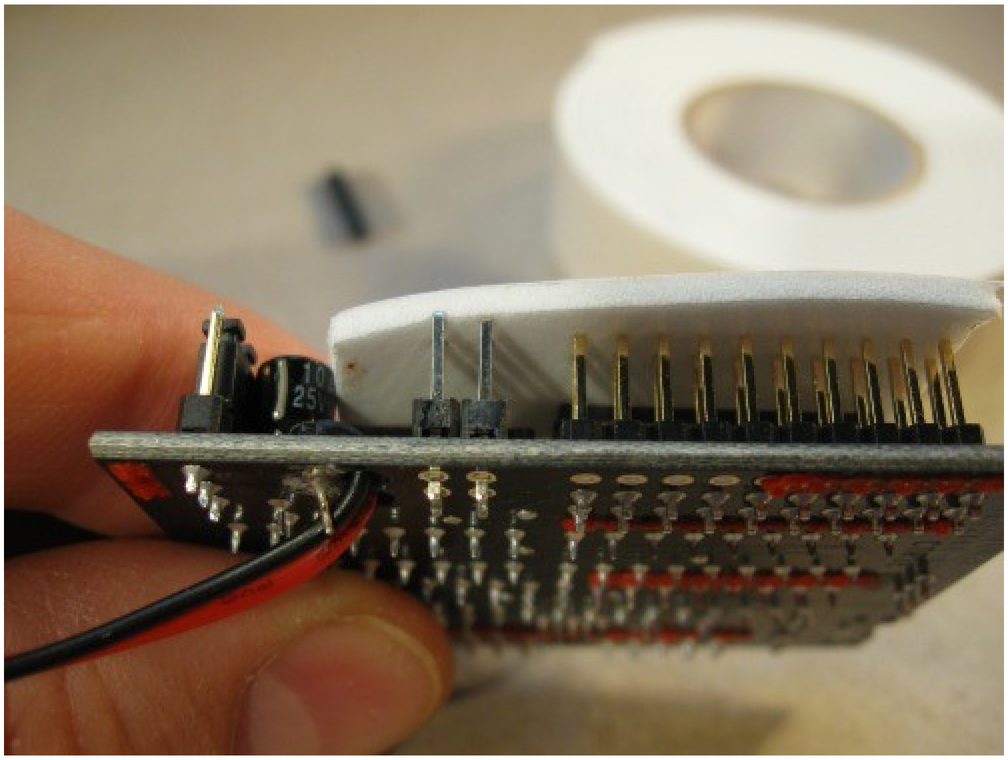
Let's look at somewhere where we for sure are going to add some pins.. Yes, the motor-outs.



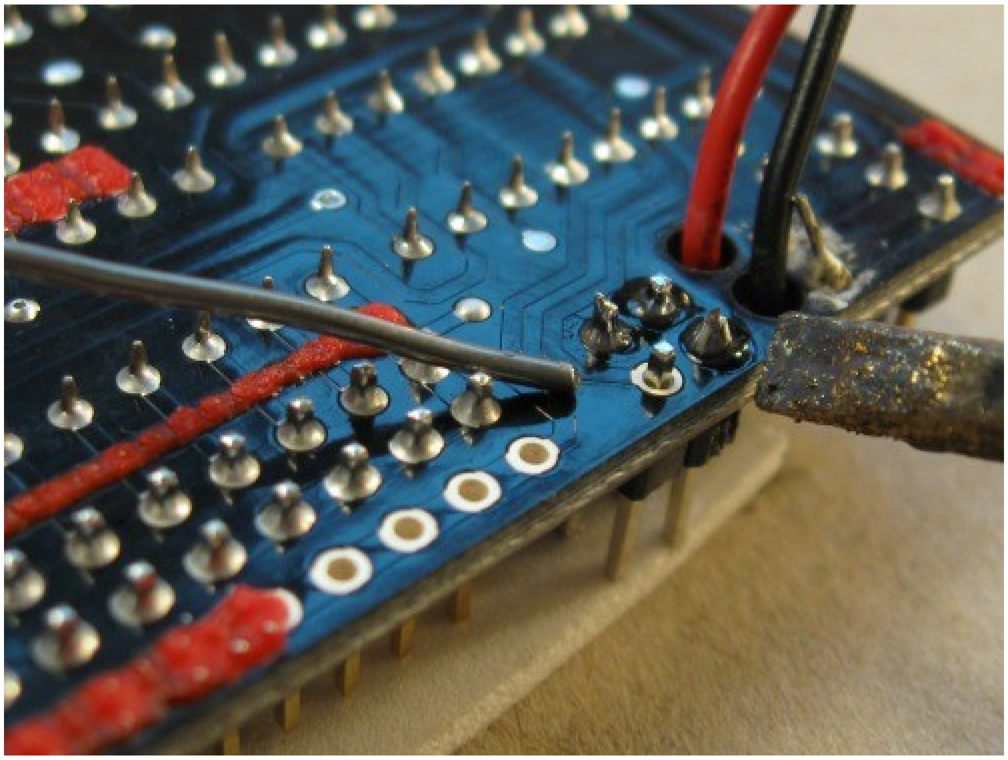
The A & B on the board. We will get back to them, but for now, snap off 2 times 2 pins, and plug them in.

It does not matter if you snap off single pins or anything like that. They are simply little metal rods in plastic. Short side down into board.

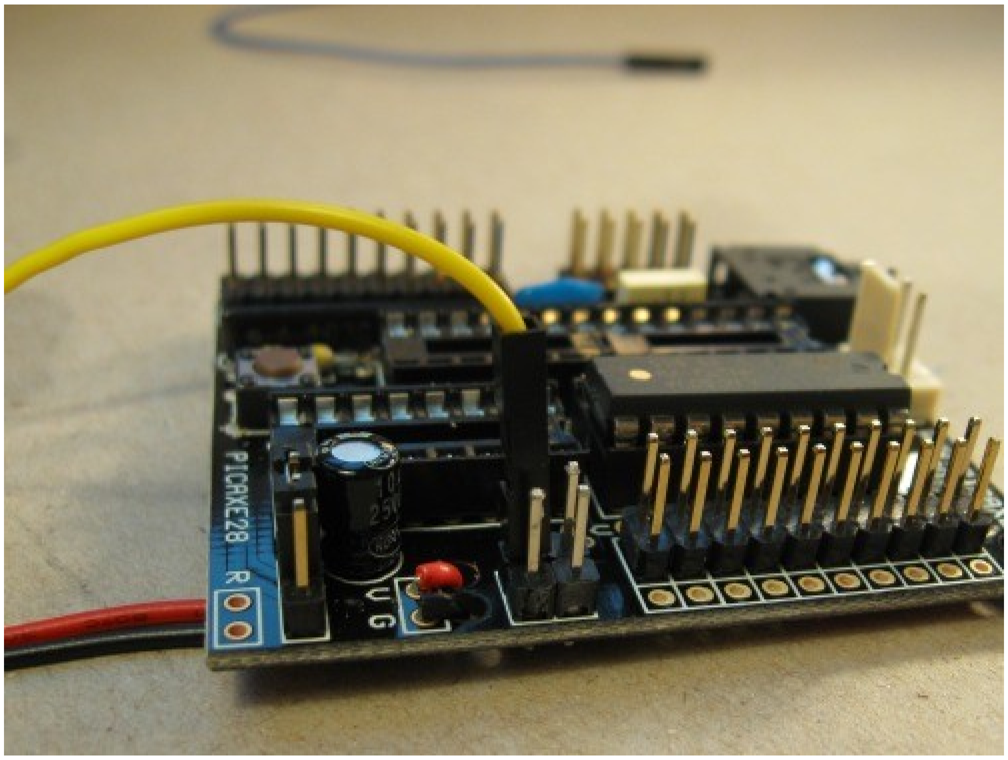
Use some foam tape to hold the pins in place.



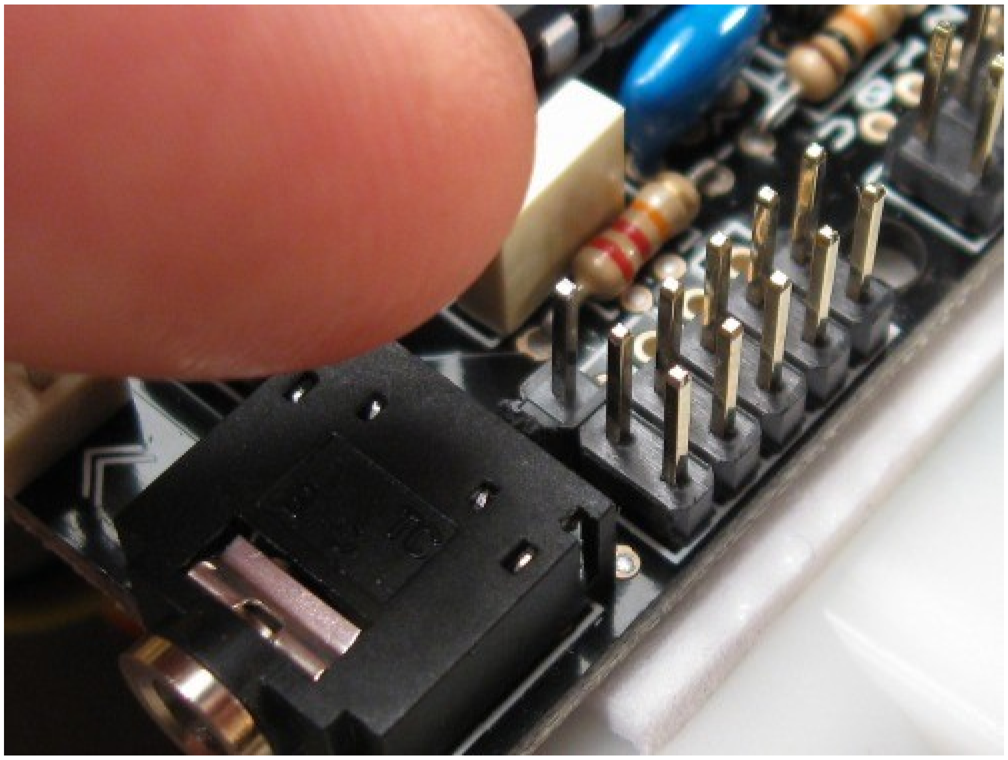
Turn the board around and solder...



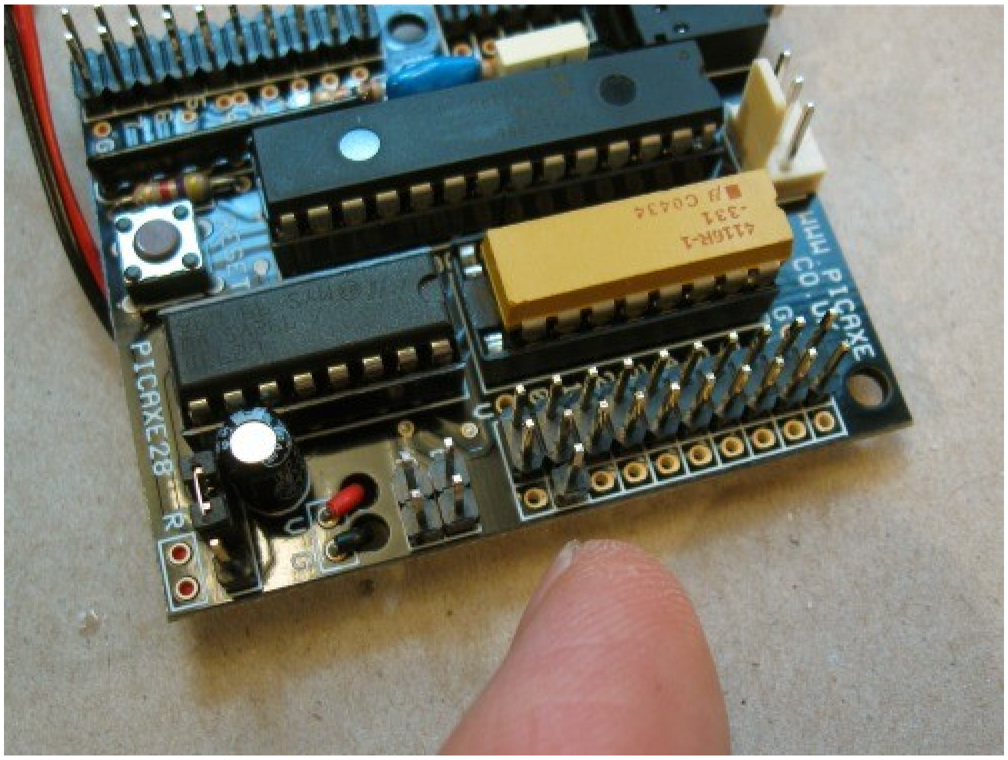
And now, tatataaaa.. You can plug in any standard female header, where you used to have a hole :)



Nice, and while you are at it, also solder a pin into analogue port 0, that we are going to use for the Sharp:



Then solder a pin into output 0..



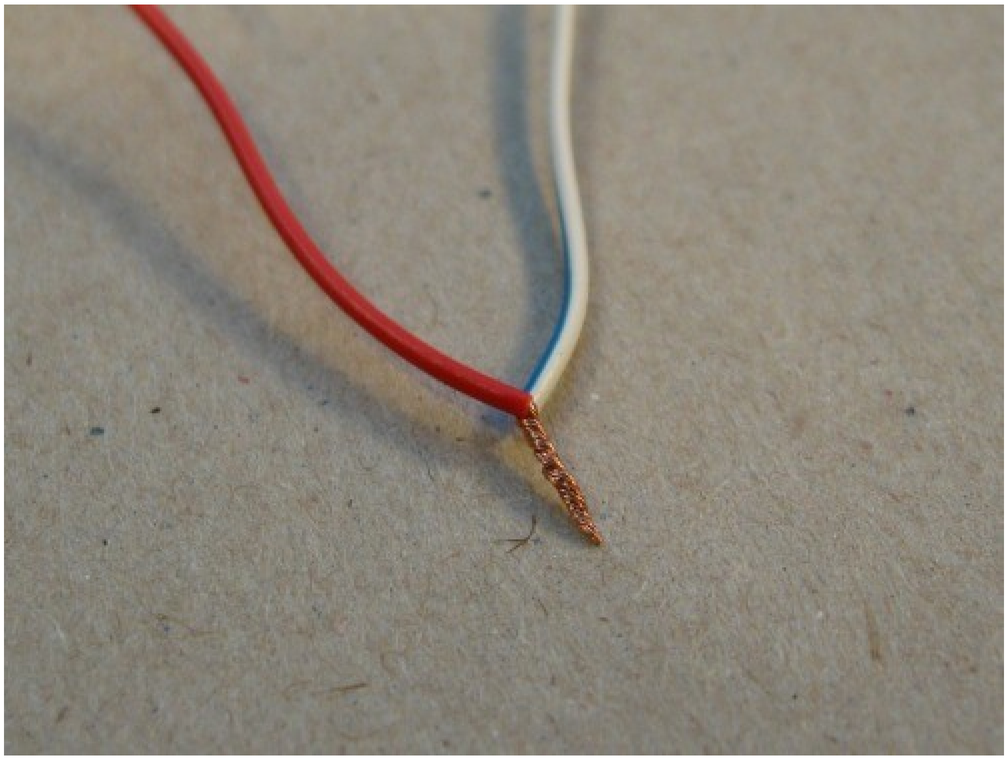
And you are done with pin-soldering.

If it was me, I'd just solder pins in all the holes, but you may want to leave that for later. You have soldered all the pins that we need for this project now.

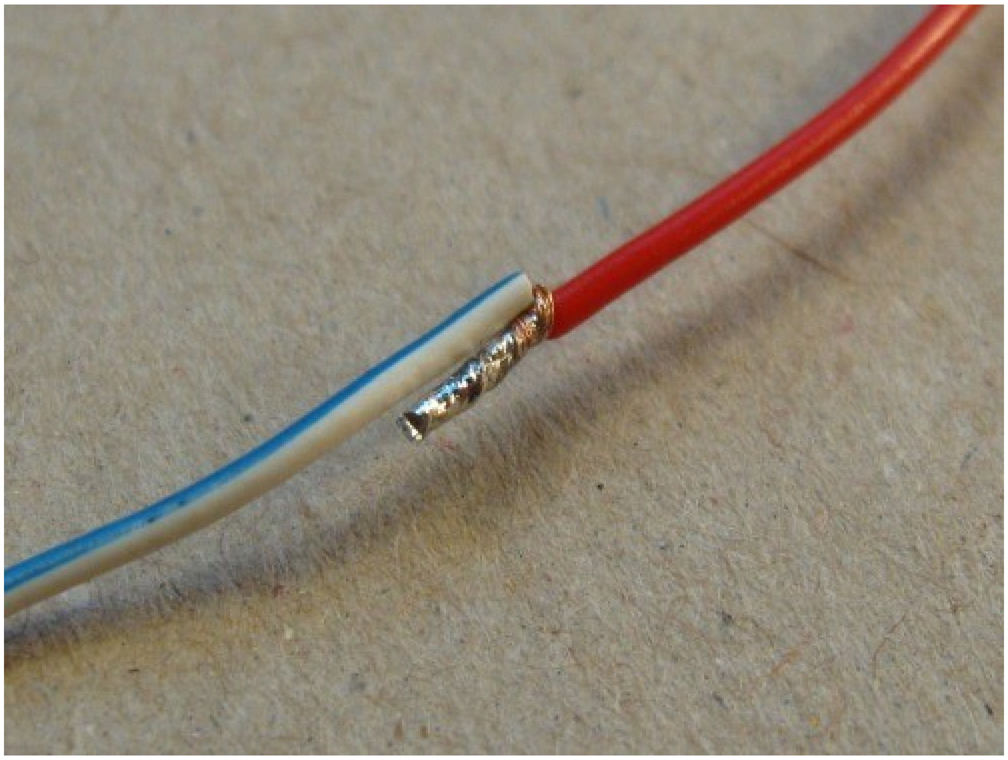
Next, general instructions: Extensions and alternations of wires and cables:

Connecting 2 wires "The right way" is almost a religion to some. Here is how I do it :)

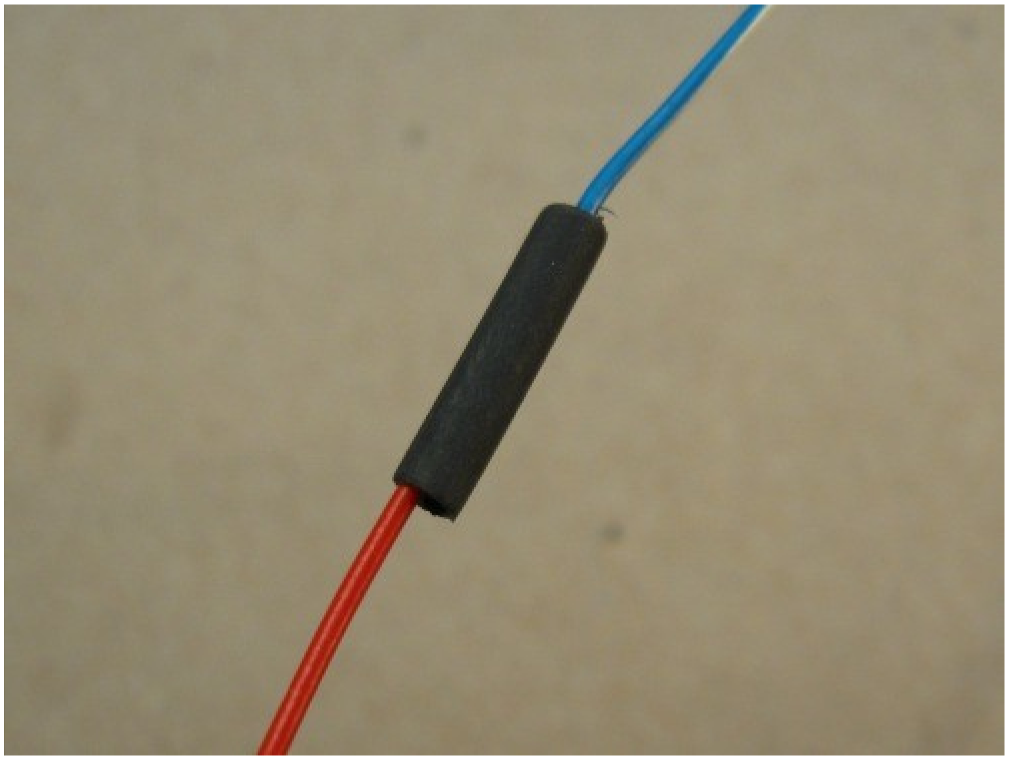
First, I simply twist together the 2 wires



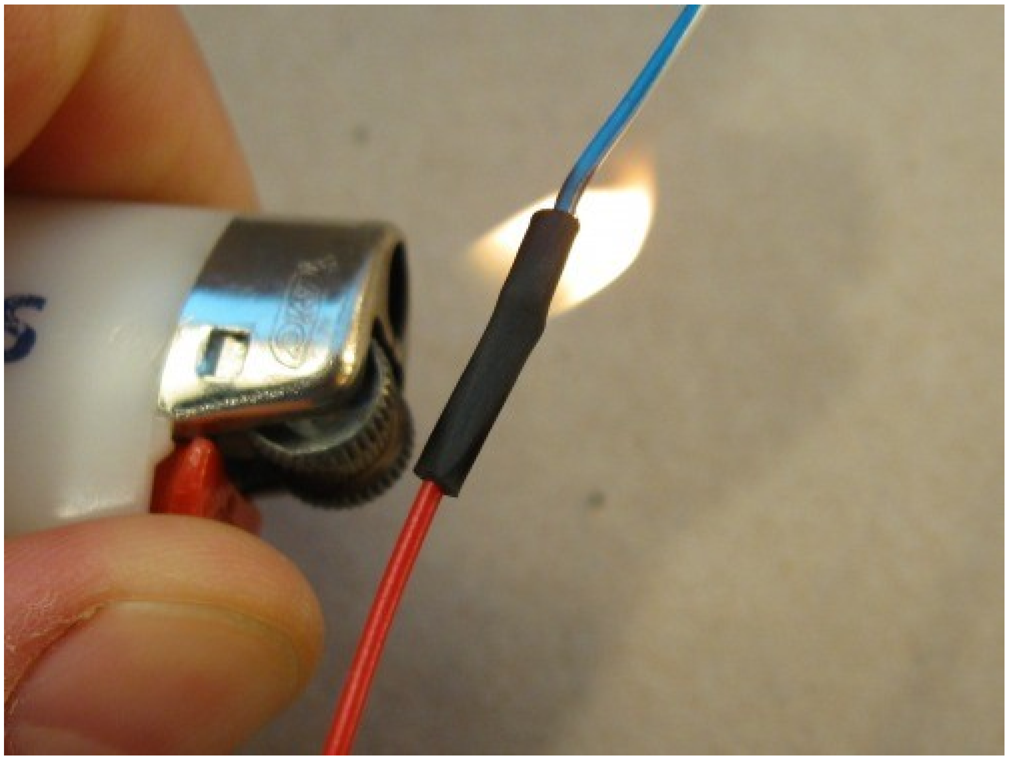
Then I solder them together, cut some off, if it is too long, and bend it along the side of one of them.



However, BEFORE I do this, I make sure that I have cut off a little piece of heat shrinking tube, and placed it over one of the wires. Then I slide that over...



A lighter quickly heats it up. This makes it shrink, hence the name, and it is a perfect insulation.



I don't think you realize how hard it was to take that picture all by myself :) It had to be in focus on the right spot, you know. And yes, the wire got a little burned :p Good shot though, if I may say so myself.

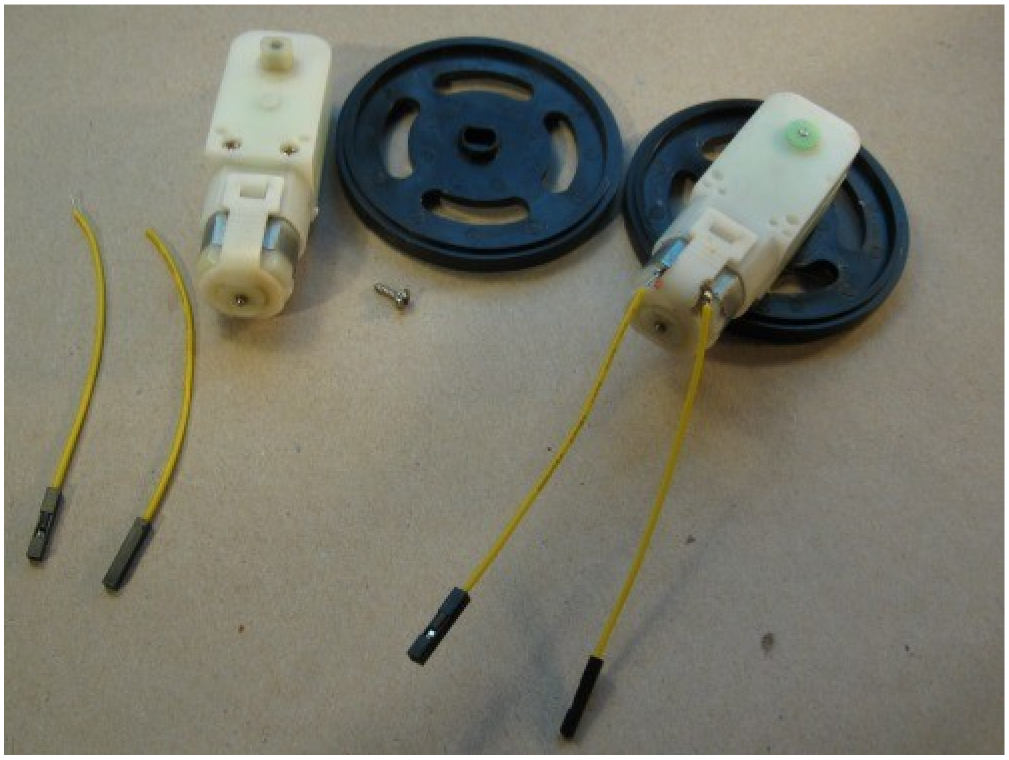
From now on, I expect you to just extend wires that are too short, hook up headers on wires when needed, and if you need to connect something to the board, where there is only a hole... you simply add a pin :)

Get it together!

Fixing up the motors

Mount the wheels on the geared motors. You may have a completely different set than I do here, but as long as they are geared motors that run fine on a few volts, and some nice wheels, you will be all right.

When you have the wheels on the motors, cut one of the female-to-female wires in halves, take away some of the plastic from the end of the wire, and solder it on. And do the same for the other motor.



Make sure no solder or wires touches the metal on the motor :)

Some wheels come with optional rubber tires. It can be a good idea to wait with putting on this rubber, because if the robot is stuck, it can just slide, which is nice when testing and developing.

Chips in the board!

The Picaxe 28X1 Microcontroller that you have here, and the board, is a pretty amazing and very powerful little package.

I remember how amazed I was that I could actually make this control everything I could imagine, I hope you will get that sensation at some point as well; Seriously, you can make this thing do all sorts of stuff with all sorts of electronics. Even if you know nothing about anything, and electronics is as strange to you, as it is to me.

You can also make it handle your servos, motors, calculations, monitoring distance.. everything a robot needs. And that is what we are going to set it up for now

The microcontroller is the long chip. That is the one you program, and then there are inputs and outputs on the board that it can use.

Have a look at this page: 28 pin Project Board (AXE020), Picaxe for dummies

Now, I do not expect you to read that page *now*, because I have promised you that you will building building the robot as fast as possible :) However, it is important that you read that page at some point, to learn about the board, and the microcontroller. Promise me to get back to that, make a bookmark for next step ;)

OK, enough talk, insert the long black chip, that is the microcontroller.

Make sure to turn it the right way: It has a marking in one end, and so does the socket. They should match.

Now, when you bought the board, it should already have a black chip in it, in the slot where I have placed the yellow chip, in the picture below.

Take up the black chip, and as I did, replace it with the yellow one. It does not have enough pins, but see picture for what end to leave open. (the inner side)

The yellow chip is sitting between the microcontroller and the topmost row of pins on the picture. That row has (as you will know when you read about the board, your bookmark, remember?) the outputs.

We are going to hook up the servo to one of these. Servos are sending a lot of electrical noise back on the line. The Yellow chip is a series of 330 Ohm resistors, that will reduce the amount of noise that is sent back to the microcontroller. It is simply straight lines across, with some resistance between, making the signals travelling both ways a little weaker. So it is there to protect the microcontroller.

Frankly, I have never heard of anyone frying a microcontroller because of servo noise, but since manuals tells us to do this, and the board is prepared for it, we might as well.

I have also heard of black versions of this chip. How boring, but none the less, and yes; You can use it, no matter the colour, if it has the same functionality.

The black chip that was in its place, is a so-called Darlington driver. If you need more info than that, please read the manuals :) But it is a good chip, if you are not hooking up servos right on the board. It is amplifying the signals, so for instance you can hook up a speaker right on it (where we now will be placing a servo) - and using the command "Sound", you can make it beep quite loud, play tunes etcetera. You have got to try that as well! Just not now ;) Disadvantage of using the microcontroller and this board for everything is, that when you want to play with servos, you take out the Darlington, and the fun it provides. But there is more, wait and see!

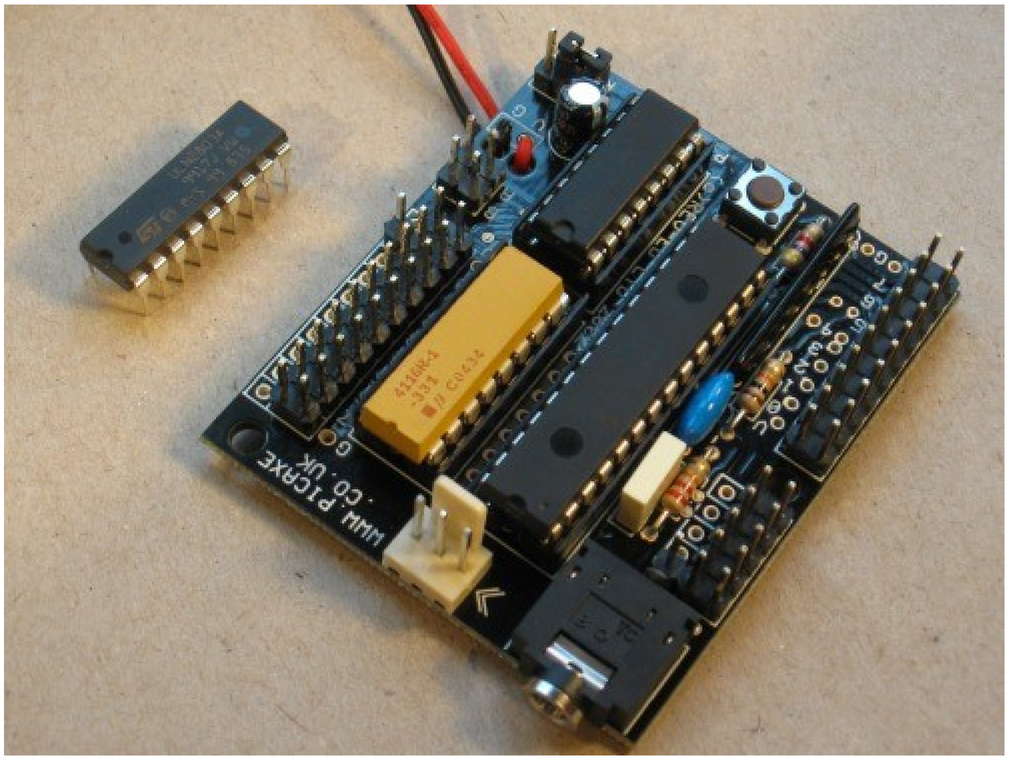
Last chip is the motor controller, throw that in as well!

When your microcontroller is placed in your board, it can switch power on/off. You could use that to drive motors. However, motors are "rough", and could fry the microcontroller's outputs. And another thing is that if the microcontroller can only turn power on/off, then.. how do you drive backwards?

A motor driver takes care of all this;
This little motor-driver-in-a-chip can drive a pair of small motors (600 mA each, for the tech interested), without frying the microcontroller. And furthermore; It can make the motors go backwards. Which is handy when facing a wall :)

Your nice board has a place for a motor driver that can take a pair of small motors, and make them drive both forward and reverse.

The board is set up, so the microcontroller's outputs 4, 5, 6, and 7 are fed into the motor controller, and out comes 2 fine pairs of wires that you can hook up to a pair of motors: Motor A and Motor B. And you just soldered pins into them, how nice.



Tip: New chips in

New and unused chips usually have the two rows of legs a little too wide apart. So gently push down the side of the chip towards a table to correct it, before inserting it into a slot. You will understand me once you try to place a new chip in a socket ;)

Tip: Old chips out

It is easy to get a chip out of a socket, by inserting a flat screwdriver below it, push it under, to the far end, and gently push it up.

Fact: Later in your life, you will want the microcontroller to just be a microcontroller. You then buy extra other boards for something like servo control and motor control. These boards will get their commands from the microcontroller. And then your robot can do everything much more powerful, and simultaneously.

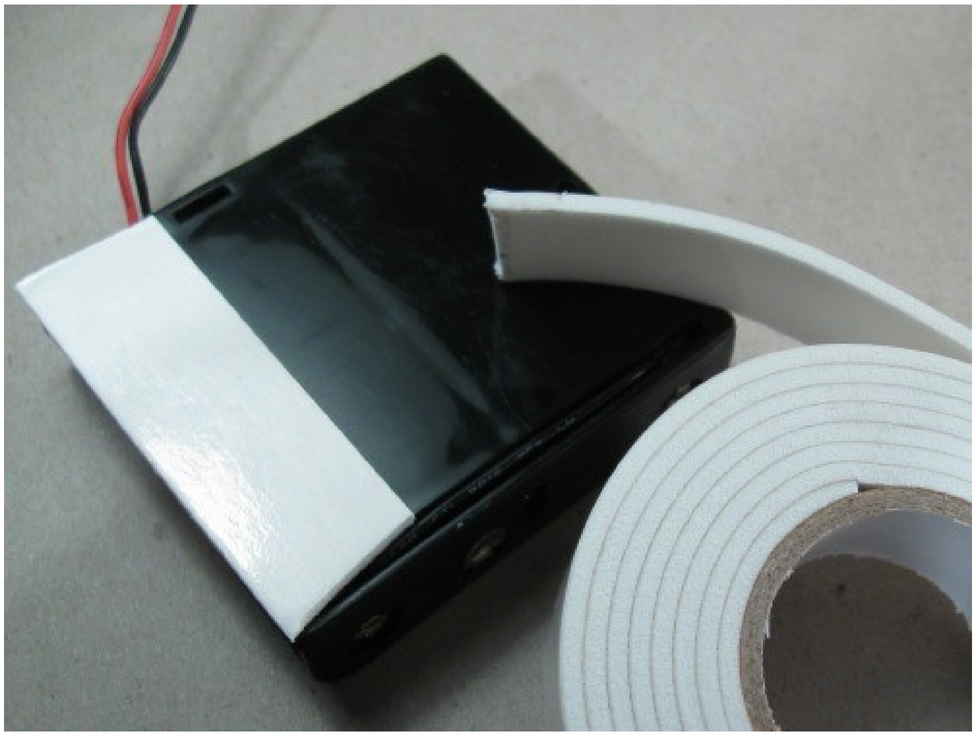
However, it is pretty amazing what you can make this set-up do, as you have it right here! Many, many cool robots and other project use far simpler or just this set-up.

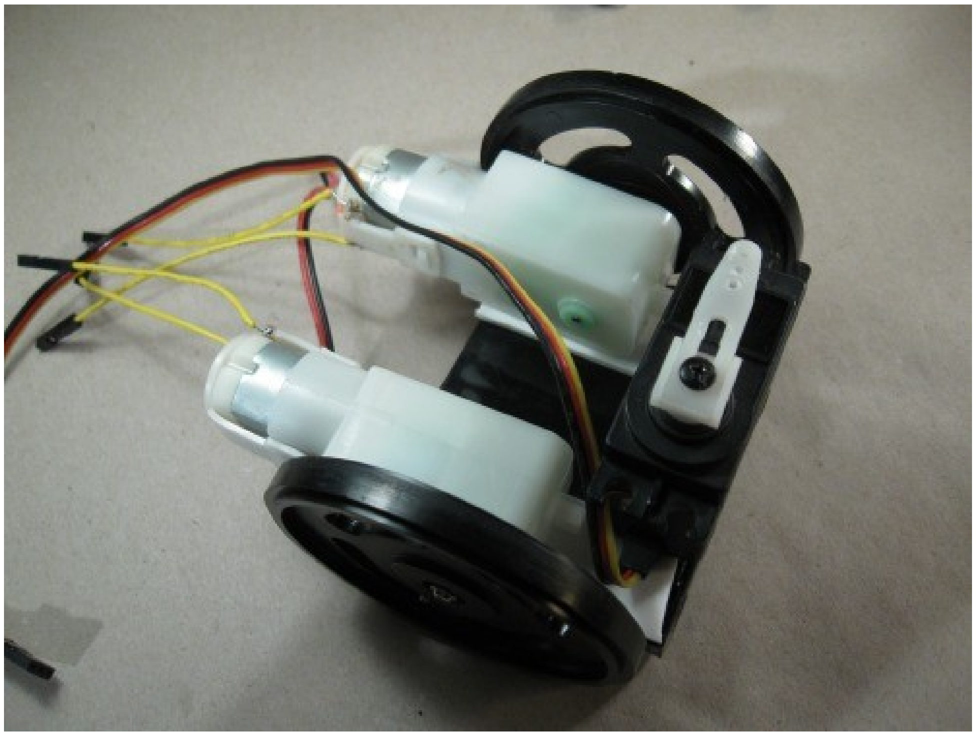
Make the body.. without a body!

In order to keep this as simple as possible, we are just glueing all the parts together, and call that a robot! Yes.

If you prefer, of course you can make it more advanced than this. Only thing to note as such is that we use 2 central wheels, each with one motor. Driving both forward makes the robot drive forward, and if one is going backwards with one forward, it turns on a plate. If one is still, and the other is driving, it is more like "sliding" to one side.

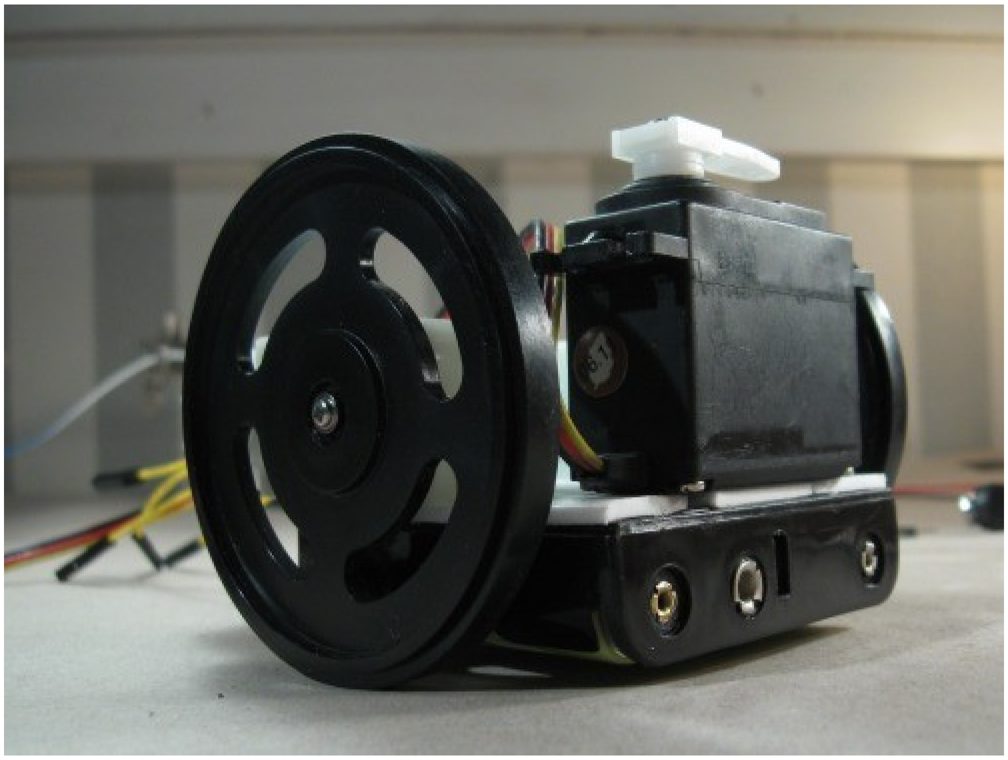
What you can do, is simply to stick on the motors to the battery holder, using the foamy tape. If the batteries are in there, and the wheels are big enough to have them placed below the axle, the whole thing will simply balance on its own. Quite strange, actually, when I think about it :)


Somehow also leave some room for the servo in front. Or stick it on to the front of it all.

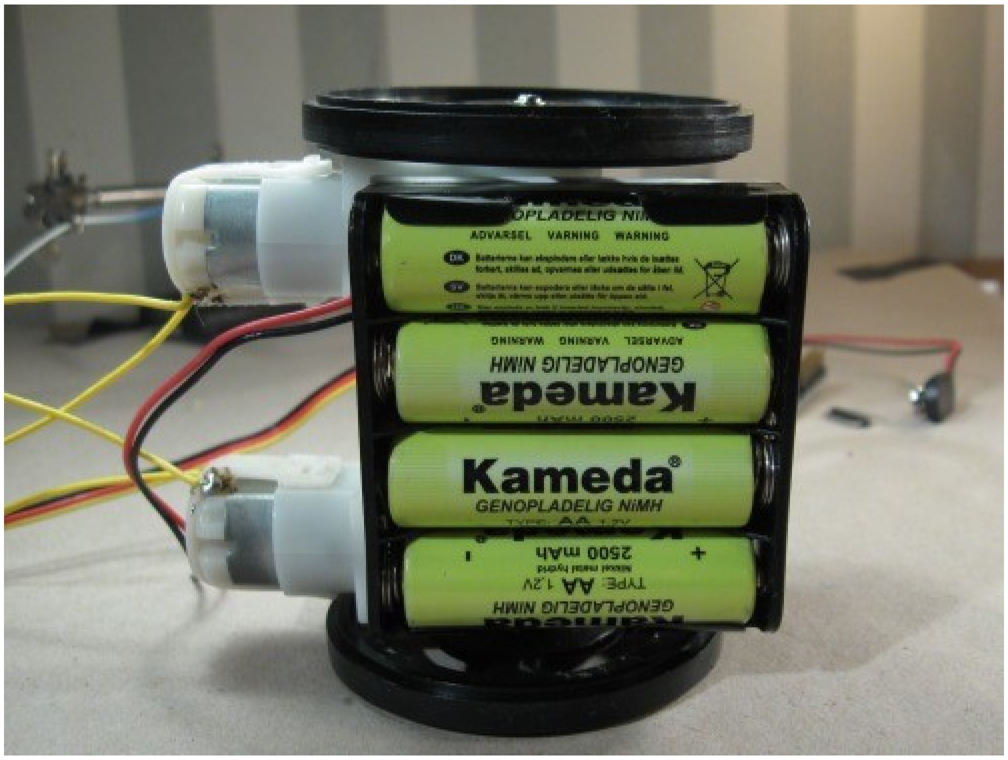


Most important is that wheels touch ground, and the rest is more or less in balance. It does not matter if it is tipping a little backwards.

Feel free to make your own design, of course :) If you do not like the balancing part, or cannot make it work, you can just add some smooth "pads", or extra wheels. A pearl, or an old LED could make nice little "third wheels", that usually are placed in the rear of the robot.



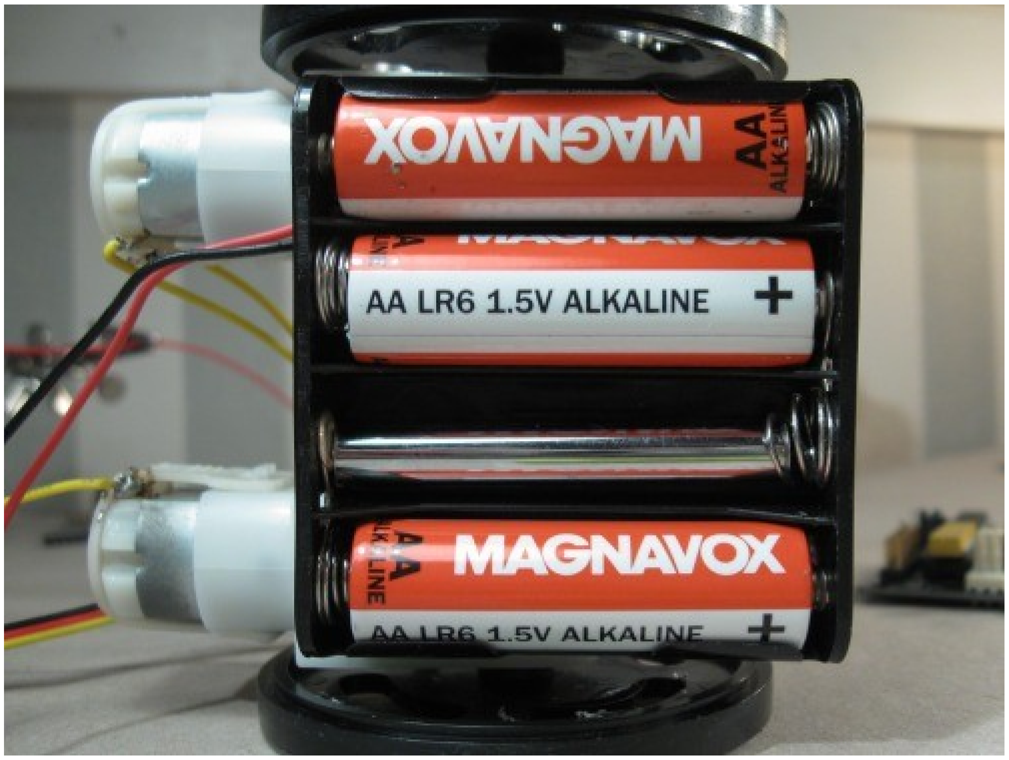
Now, as you can see, this time, I used the 4-battery holder. Because that is the biggest one, which makes it easier to stick it all on to it.



- But if you are using non-rechargeables, and only should use 3 batteries, here is a tip:

Find an old telescopic antenna, from a radio or something.

Cut off a piece (Here is a tip on how to cut it), and insert it instead of one of the batteries. Bingo ;)

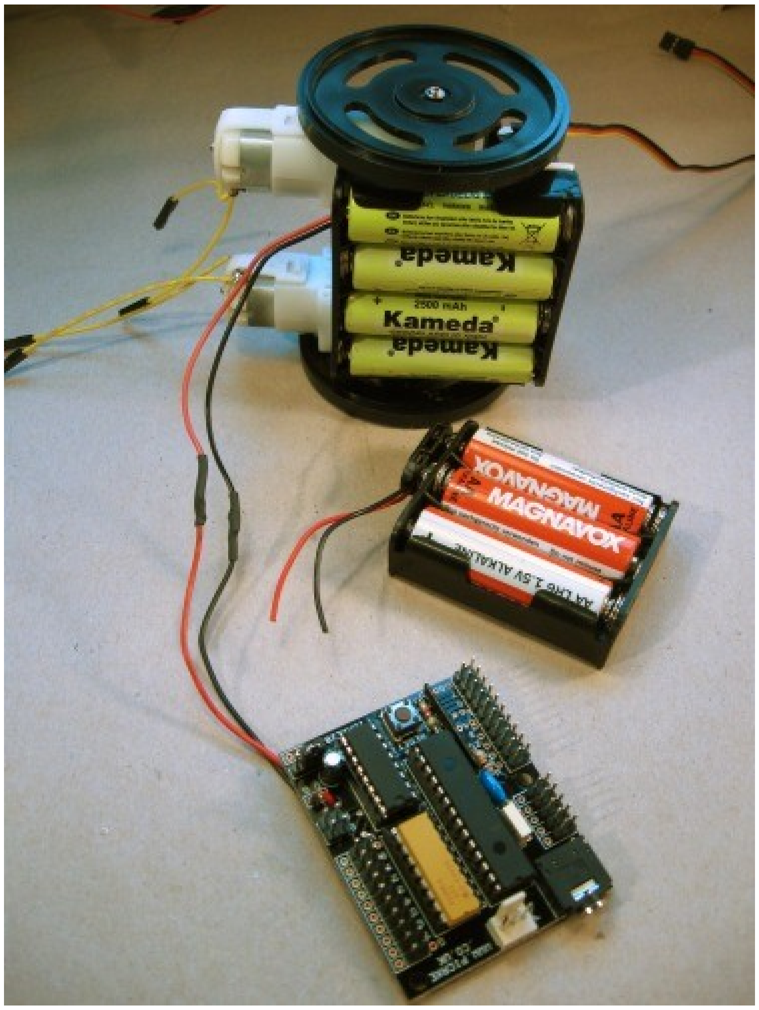


OK, next thing is to place the board on the robot, and hook everything up (apart from the Sharp, wait with that).

First: Take out the batteries again (or at least one of them). Just to make sure you don't fry something by accident. We don't have an On/Of on this robot; Batteries in, and it is alive. But we want it dead now, so one battery out! (and not like on next picture, doh!)

Some battery holders have wires (like the one I am using), and some have a clip that fits right onto the clip on the board, as illustrated in the 3 battery holder below. If you have a clip, then hook it up, you are done. If you have wires like me, cut off the clip from the board, and connect black with black and red with red. (and use shorter wires than I did ;)

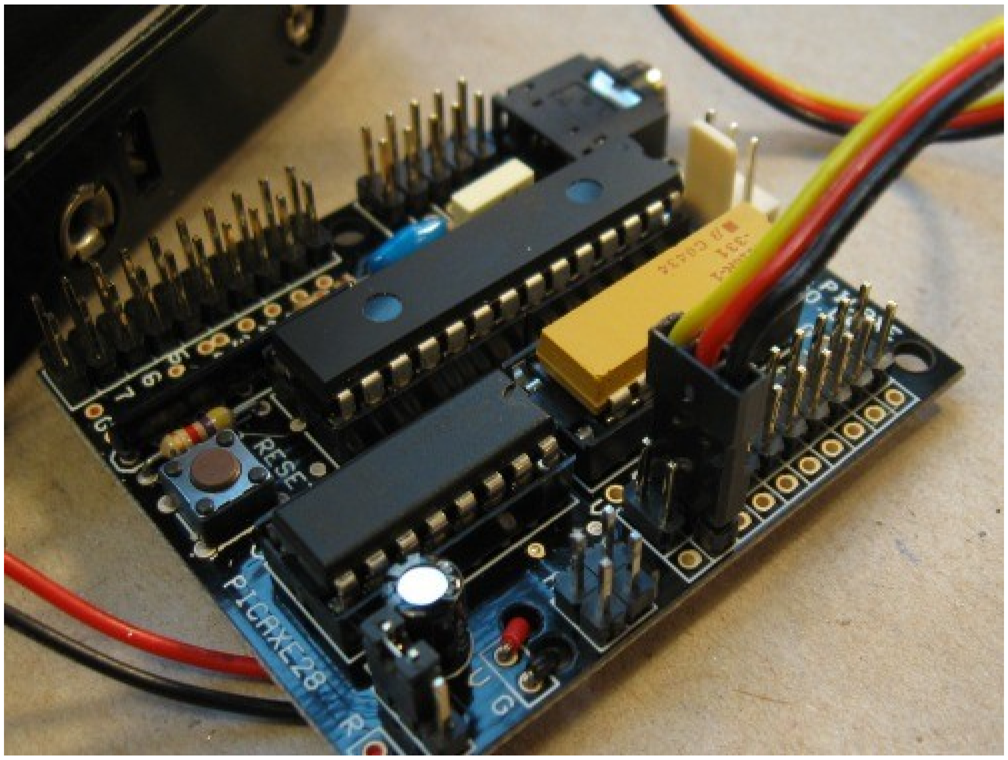
The + from the battery should go the where the RED is hooked up on the board, from the factory.



Hook up stuff to the board

Hook up the servo

Your servos wires colours may be different, but the hints are; Brown or Black (Ground) to the outside, Red (Volt) in the middle, and yellow or white (Signal) on the inside of the board. These descriptions may make most sense to you, if you have read about the board, as you promised me to do earlier. But for now you can just note the colours, and make sure to plug in your servo the right way around :)



Mount the board, and hook up the motors

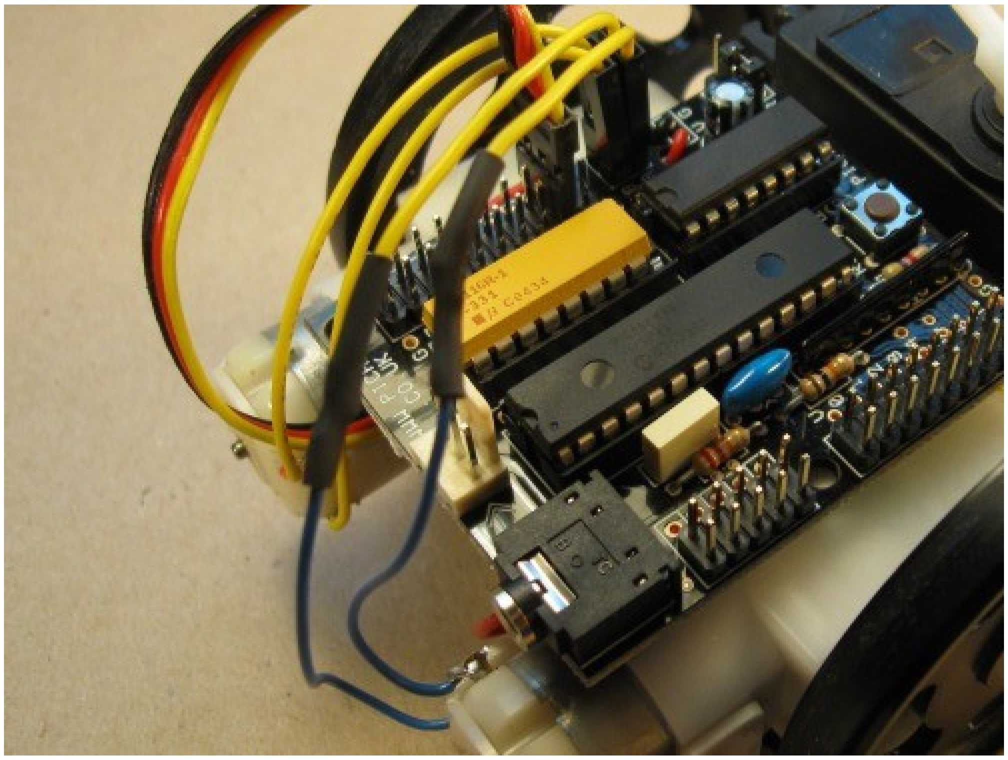
With some (more) foam tape, stick on the board to the rest of the robot.

Make sure the mini-jack (the metal ring in one end of the board) is placed so it is easy to plug in a cable, because you will be doing that many times! And be careful to make sure that the bottom of the board does not touch any metal ;) That would cause short circuits.

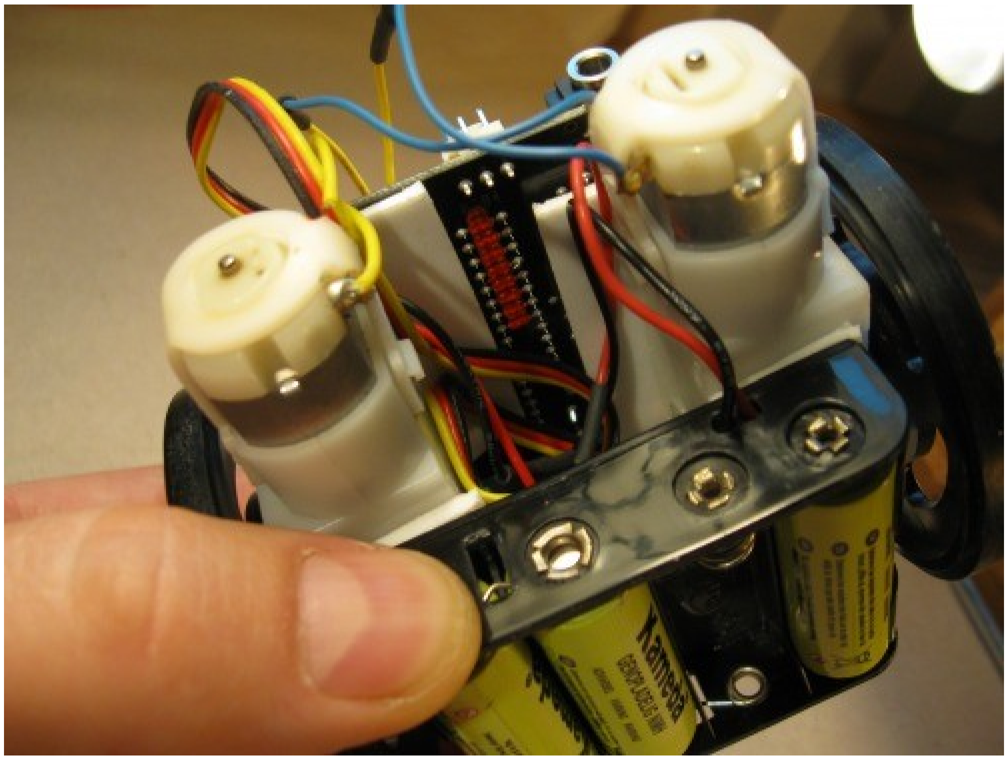
Hook up the motor wires, into the A&B-pins that you soldered on earlier:

One motor's 2 wires goes to A on the board, the other two wires go to B. It does not matter which motor connects to which output, we will manage the rest in the programming.

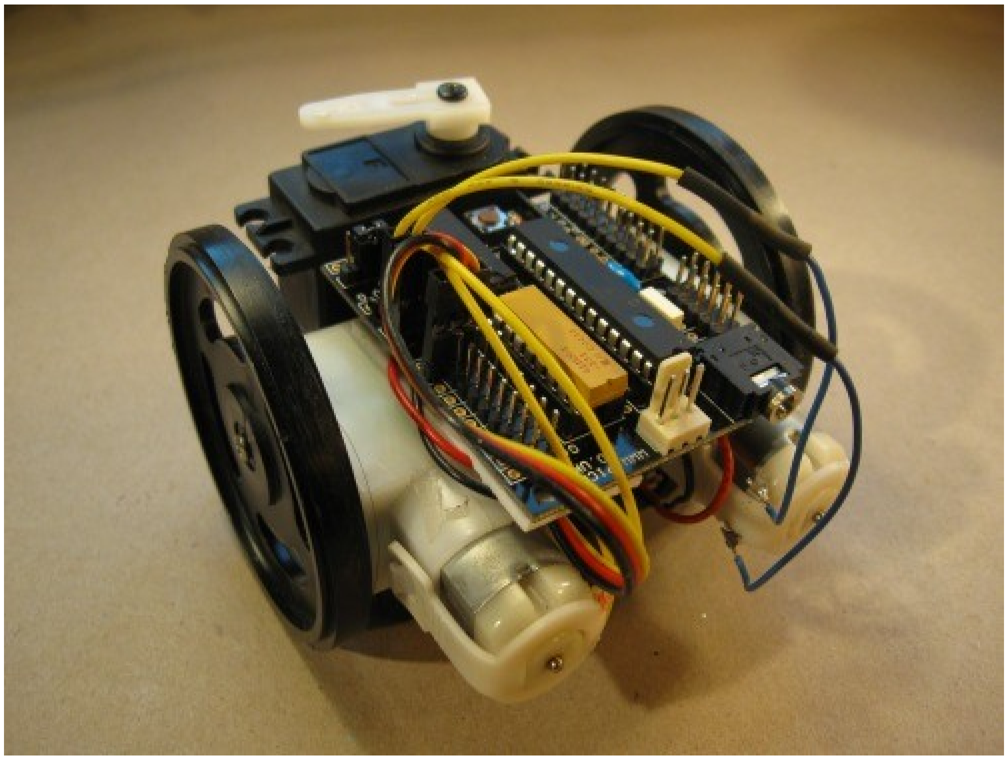
Oops, one of my motors wires was too short, so I added little blue extensions from a scrap piece of wire that I found.



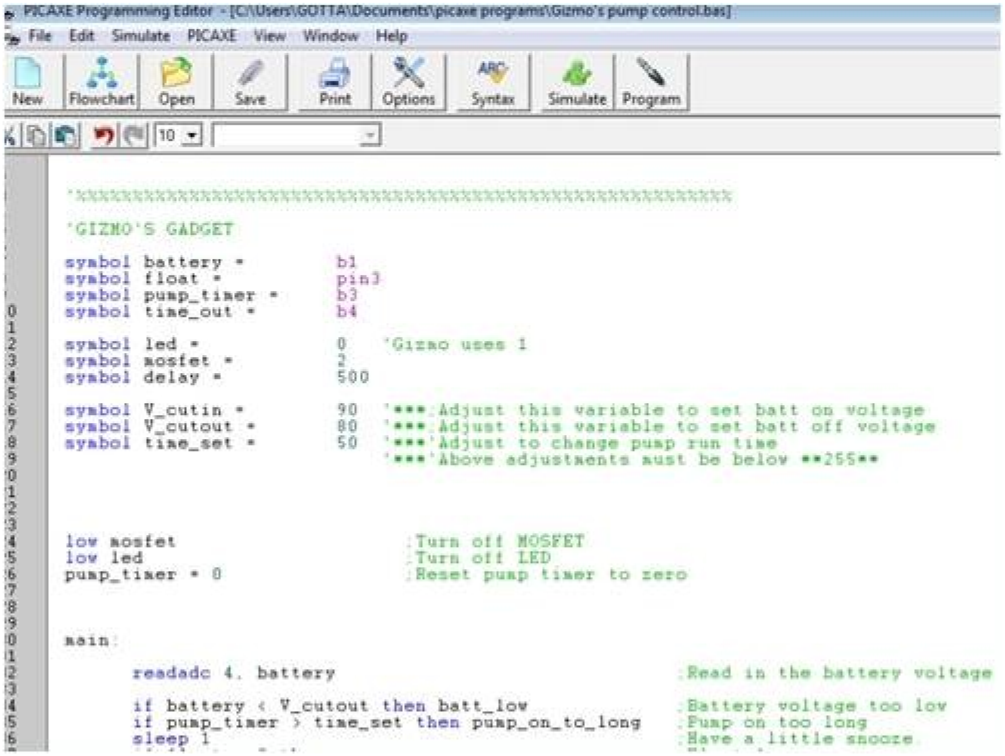
There is a nice little room to stuff in excess lengths of wires :)



And voilà!



Break out, software


PICAXE Programming Editor - [C:\Users\GOTTA\Documents\picaxe programs\Gizmo's pump control.bas]
File Edit Simulate PICAXE View Window Help
New
Flowchart
Open
Save
Print
Options
Syntax
Simulate
Program
10

```
'%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

'GIZMO'S GADGET

symbol battery =        b1
symbol float =          pin3
symbol pump_timer =     b3
symbol time_out =       b4

symbol led =            0    'Gizmo uses 1
symbol mosfet =         2
symbol delay =          500

symbol V_cutin =        90   '***;Adjust this variable to set batt on voltage
symbol V_cutout =       80   '***;Adjust this variable to set batt off voltage
symbol time_set =       50   '***'Adjust to change pump run time
                             '***'Above adjustments must be below **255**



low mosfet                      ;Turn off MOSFET
low led                         ;Turn off LED
pump_timer = 0                  ;Reset pump timer to zero


main:

      readadc 4, battery                                ;Read in the battery voltage

      if battery < V_cutout then batt_low               ;Battery voltage too low
      if pump_timer > time_set then pump_on_to_long     ;Pump on too long
      sleep 1                                           ;Have a little snooze
```

(Yes, I know your robot still has no face :)

We need to turn the servo to centre.

Of course you could try and do it by hand, and estimate it to be in centre, but the smarter way is to let the microcontroller put the servo to centre. Because then you can mount "the face", facing forward, right where the microcontroller thinks it should be, when facing forward.

You are going to take a "Time-Out" from this tutorial, because you will need to set up your computer to know there is a programming cable attached, and a piece of programming software must be installed.

I cannot help you much with this, as it depends on the type of computer you have got, and what the folks at the Picaxe website have updated after I wrote this.

However, go to

http://www.rev-ed.co.uk/picaxe (or the easier to remember: picaxe.com - that redirects you)

Depending on your OS, you want either the

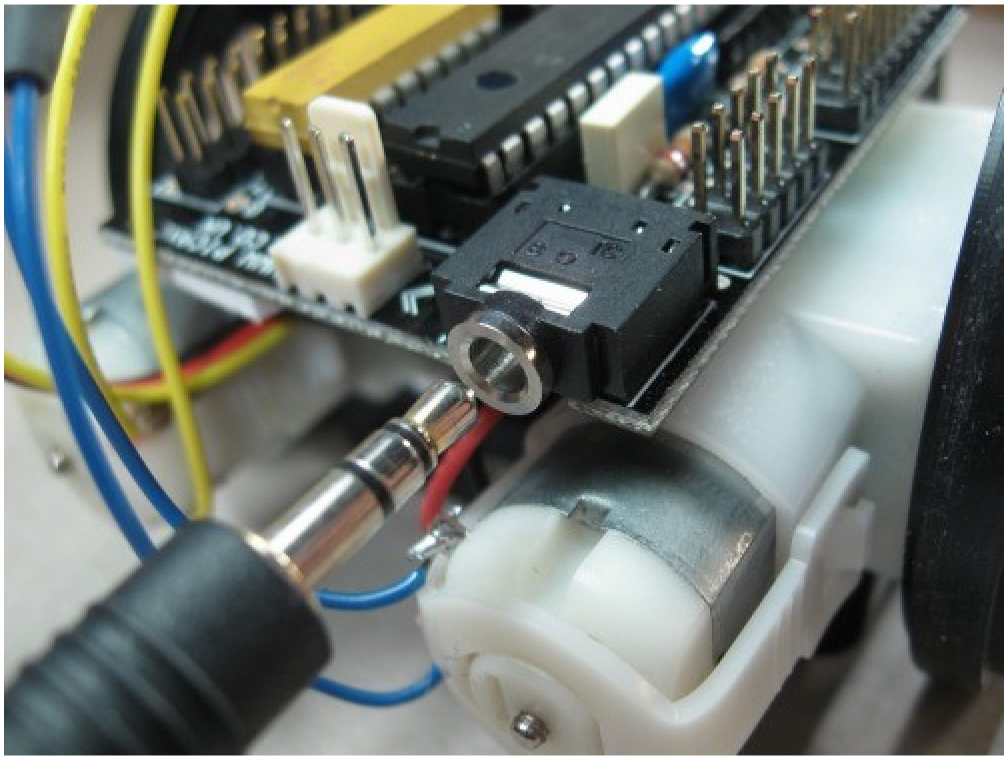
Free PICAXE Programming Editor or the **AXEpad** (which is also free, it is just not in the name ;)

Download and install which ever they claim will make you able to program the Picaxe chips!

Then you want to find the part that says

AXE027 PICAXE USB Download Cable

Install the driver and cable as described, and plug the jack into the board.



Insert all batteries in your robot.. wait.. wait.. sniff.. anything smells funny? No sparks, no smoke? No? OK, go on then.

Most versions of the Picaxe programming software have some form of menu item called "Options". Enter that, and look for a page or tab that says "Mode". Some editors even open this very page for you when you first start up the program.

Here you should find a button that says "Firmware" or "Check firmware version". Click that.

Now what should happen is that the editor talks to the cable, that talks to the microcontroller, and asks it what kind of a Picaxe chip it is. It should return something like "28X1/40X1", and this should be displayed on the screen for you.

If yes, then good; You have contact. Now somewhere in the same screens, you should be able to set the mode of the editor, set it to 28X1/40X1.

(It is a big mystery to me why this has to happen, by the way; Apparently the editor can see what kind of chip is there, so why on earth can it just not set it by itself?. hmm.. let me know if you find this reason one day ;)

OK, if you get any errors, cannot find the microcontroller or something is reported wrong with the cable, I will have to ask you to lean on Picaxe's help and manuals. It's usually very simple, so try again if something is wrong. Or try from another computer, just to see how it should work, and then try the first one again, and bug track that way.

Now, enter the main programming editor; It is like a big text editor. If no project is open, go to "File"; and open "New Basic" / "New".

In here write this:

```
servo 0, 150

wait 2
```

This is your first program, and it is very simple. The first line tells the microcontroller that there is a servo on pin 0, and that it should be put in the centre (center) position, which is **150**.

The next line tells it to think about life for 2 seconds (which gives the servo time to turn).

And after this, the microcontroller will stop doing anything at all, it will go zombie!

Write it in, and transfer the code to the microcontroller. That is done on some systems by pressing F5. No, wait, I think it is so on all systems. On all that I can test from here anyway :) You could also click "Program".

You should be told that the program is transferring, and then magic should happen; The servo should turn to the centre position.

Perhaps it did not do much, but that may be because it was already in centre.

Try to hold down the "Reset" switch that is placed on the board, while turning the servo to one side. Then let go of the reset, and it should turn back in place.

Perhaps you do not think it is centre, but it is. But maybe your servo "horn" is just mounted awkward. In the middle of it, there is a screw. Unscrew and take of the horn, make sure the microcontroller did put the servo in centre, and then screw on the "horn" (disc, or what ever) again, so it is the way you think it should look like when centred.

Now, let's try to turn the servo to the sides, let's extend the program:

```
servo 0, 75

wait 2

servo 0, 225

wait 2

servo 0, 150

wait 2
```

the servo command should only be using values from 75 to 225, which is the most a standard servo can go to either side. Values out of this range are not recommended, may produce funny results.

Every time you run this program (you can unplug the cable, take out the batteries, and turn it on again without the cable), it will start from the top. And every time you press reset, it will.

If you'd like it to go in a loop, you can add a label in the top, and in the bottom make it go back to that label. We make up any name for a label, just a single word, followed by a colon, watch:

```
servofun:
servo 0, 75
wait 2
servo 0, 225
wait 2
servo 0, 150
wait 2
goto servofun
```

Now it just goes on and on.. Try to play around some with it, change values etcetera, if you like :)

...

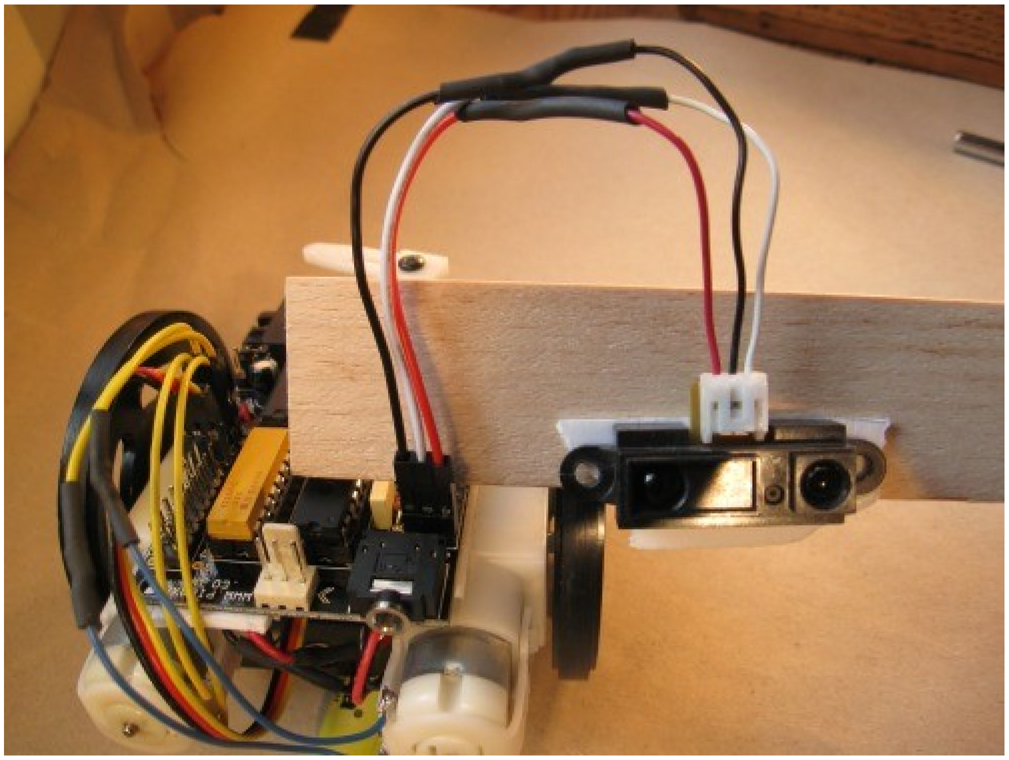
OK, back to building the robot ;)

Plug in the wire to the Sharp, if it was not in from the shop. in other words; Make sure there are 3 wires coming from the Sharp. Your colours may be different, but I have red, black and white, which is pretty meaningful for V, G and Signal.

You may need to add female headers to the wires, like I did below. These can also be in any colour, of course. However, I have soldered 3 of same colours on, so you can trust the ones in my picture to be leading you to which cable goes to where.

Be careful to check that you are plugging this right in, because the Sharp can fry pretty easily.

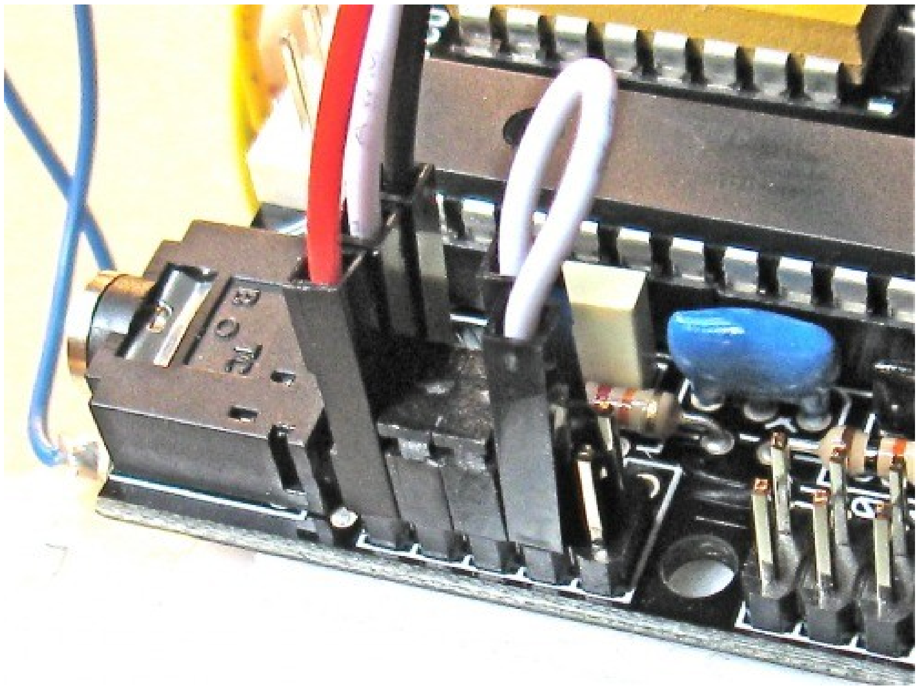
In the picture below, you can see what goes to where. The stick and strange set-up is just to make sure you can see the wires and their colours :)



You should have 3 little black things called Shortening Blocks. What they do is simply connect 2 pins next to each other.

If you don't have any, you can always just use a female-to-female jumper cable instead, like I did on the last one, with a little white cable. The blocks are smart because they don't take up any space, a wire is smart, because it can reach from one end to the other of a board.

As you can see I did on the next picture, connect analogue input 1, 2 and 3 to V, using shortening blocks or female-to-female.



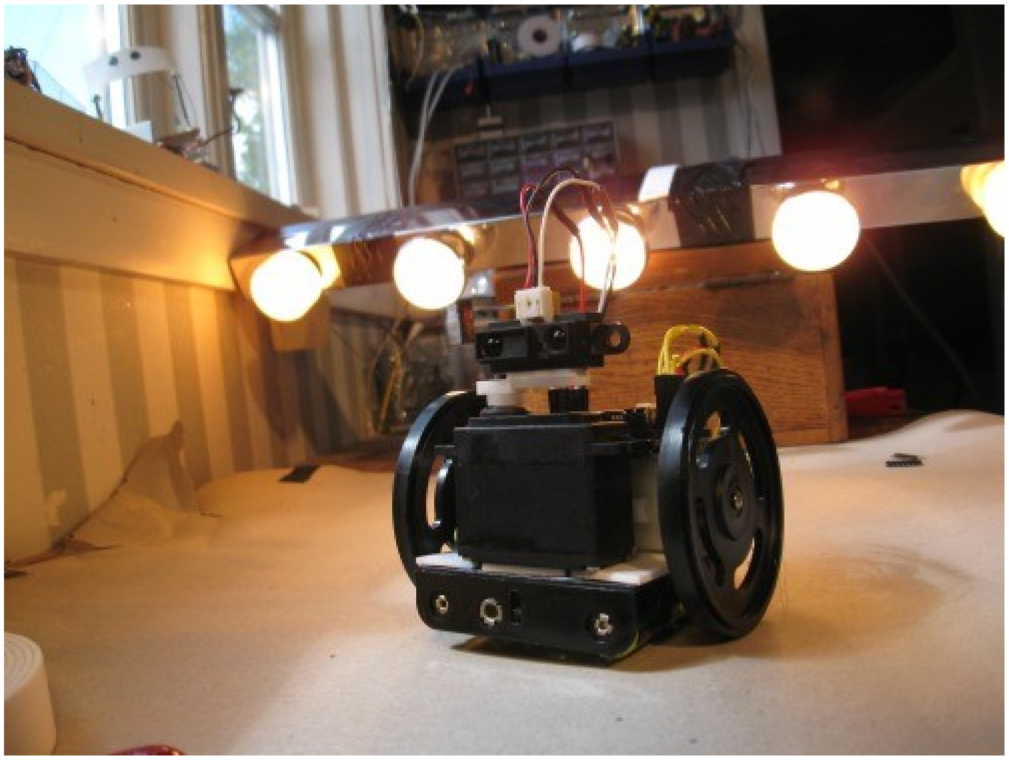
Why this? A brief and not very scientific explanation is; these 4 inputs (0, 1, 2 and 3) are analogue. Which means they measure "how much pressure is on the line". However, they are connected, if they like it or not. And so, a little pressure on one of them actually does something to the next. They are "left floating". By tying the 3 that we do not use to V, they are just returning "full value", and they are not left floating. So the last one, number 0, that we use, is way more accurate.

I have not read documentation that tells you to do this, however, I have at several occasions experienced strange readings, until I tied all unused analogue pins to either ground or V. Oh... and in fact I am writing documentation here (sort of :) So now it is written in the documentation to do this! :)

Make sure the servo faces middle, 150!

With a new piece of tape, mount the Sharp IR on the servo horn, facing forward.

Tadaa! :)



You're done building the basics!

The design may vary, you may have used other parts etc... But if you have connected as described, here are some tips to get started programming your robot:

Programming

Enter this code into your editor, and press F5 while the robot is connected:

```
main:
readadc 0, b0
debug
goto main
```

Now take your hand in front of the robot´s head and notice how the variable b0 changes value. You can use the knowledge gained to decide what should happen and when (how close things should get before..)

You may notice how things start to go "wrong" if stuff is too close to the "eyes"; The Sharp is made to work with objects 10-80 cm away. Things that are closer than 10 cm (4 inches) appear to be further way, which can be quite a challenge to program.

You can get many other distance sensors that do not have this problem. However the Sharp is the cheapest, and easiest to program, so that's why I made such a "bad" choice for you, sorry ;) Look around and see what everyone else is using, before you decide on an upgrade.

Now I advise you to put your robot up on a matchbox or similar, as the wheels will start turning.

Enter this code into your editor, and press F5 while the robot is connected:

```
high 4
low 5
```

One of the wheels should turn in one direction. Does your wheels turn forward? If so, this is the instruction for that wheel to turn forward.

If the wheel is turning backwards, you can try this:

```
low 4
high 5
```

To turn the other wheel, you need to enter

```
high 6
low 7
```

(or the other way around for opposite direction.)

What happens here is that by using only the options available to the microcontroller; power on or off (High / Low) on the pins, it is commanding the motor controller to set motor A or B in forward or reverse mode.

```
low 4
low 5
low 6
low 7
```

stops all motors

The servo you have already tried.

All the way to one side is:

```
servo 0, 75 wait 2
```

- the other side is:

```
servo 0, 225 wait 2
```

- and centre:

```
servo 0, 150 wait 2
```

Here is a small program that will (should, if all is well, and if you inserted the right parameters for high/low to suit your wiring to the motors) make the robot drive around, stop in front of things, look to each side to decide which is the best, turn that way, and drive towards new adventures. In the code I have made so called *remarks*: explaining you what is going on.

You can write such comments or remarks yourself in the code, it is a good idea to keep track.

They are written with an apostrophe (or single quote) sign. However, copying this text from here might alter that to something else, and you will have to fix that manually, sorry. Your programming editor has colour codes, that will help showing you what it recognizes as comments and what as code.

```
Symbol dangerlevel = 70 ' how far away should thing be, before we react?
symbol turn = 300 ' this sets how much should be turned
symbol servo_turn = 700 ' This sets for how long time we should wait for the servo to turn (depending on it´s speed) before we measure distance

main: ' the main loop
readadc 1, b1 ' read how much distance ahead
if b1 < dangerlevel then
gosub nodanger ' if nothing ahead, drive forward
else
gosub whichway ' if obstacle ahead then decide which way is better
end if
goto main ' this ends the loop, the rest are only sub-routines


nodanger:' this should be your combination to make the robot drive forward, these you most likely need to adjust to fit the way you have wired your robots motors
high 5 : high 6 : low 4 : low 7
return


whichway:
gosub totalhalt ' first stop!

'Look one way:
gosub lturn ' look to one side
pause servo_turn ' wait for the servo to be finished turning
readadc 1, b1
gosub totalhalt

'Look the other way:
gosub rturn ' look to another side
pause servo_turn ' wait for the servo to be finished turning
readadc 1, b2
gosub totalhalt
```

```
' Decide which is the better way:
if b1<b2 then
gosub body_lturn
else
gosub body_rturn
end if
return

body_lturn:
high 6 : low 5 : low 7 : high 4 ' this should be your combination that turns the robot one way
pause turn : gosub totalhalt
return

body_rturn:
high 5 : low 6 : low 4 : high 7 ' this should be your combination that turns the robot the other way
pause turn : gosub totalhalt
return

rturn:
servo 0, 100 ' look to one side
return

lturn:
servo 0, 200 ' look to the other side
return

totalhalt:
low 4 : low 5 : low 6 : low 7 ' low on all 4 halts the robot!
Servo 0,150 ' face forward
wait 1 ' freeze all for one second
return
```

With some clever programming and tweaking, you can make the robot drive, turn its head, make decisions, make small adjustments, turn towards "interesting holes" such as doorways, all working at the same time, while driving. It looks pretty cool if you make the robot spin while the head is turning ;)

Fun time

You could also attach a lamp or LED to pin 2 & ground, and write (remember LEDs need to turn the right way around)

```
High 2
```

to turn on the lamp, and

```
Low 2
```

to turn it off ;)

- How about a Laser-pen, mounted on an extra servo? Then you could make the robot turn the laser around, and turn it on and off, pointing out places.. you can do anything now :)

Pressing "Help" in the programming editors brings out all sorts of interesting tutorials and info!

Perhaps try this:

Pull out the servo, and take up the yellow chip. Insert the Darlington that you took out earlier. Hook up the speaker to the 2 pins above where the servo was, tat is output 1. And throw in the LED, or whatever it was that you found on output 2. Then program it something like this:

```
sound 1, (100, 30)
high 2
wait 1
low 2
sound 1, (105, 60)
```

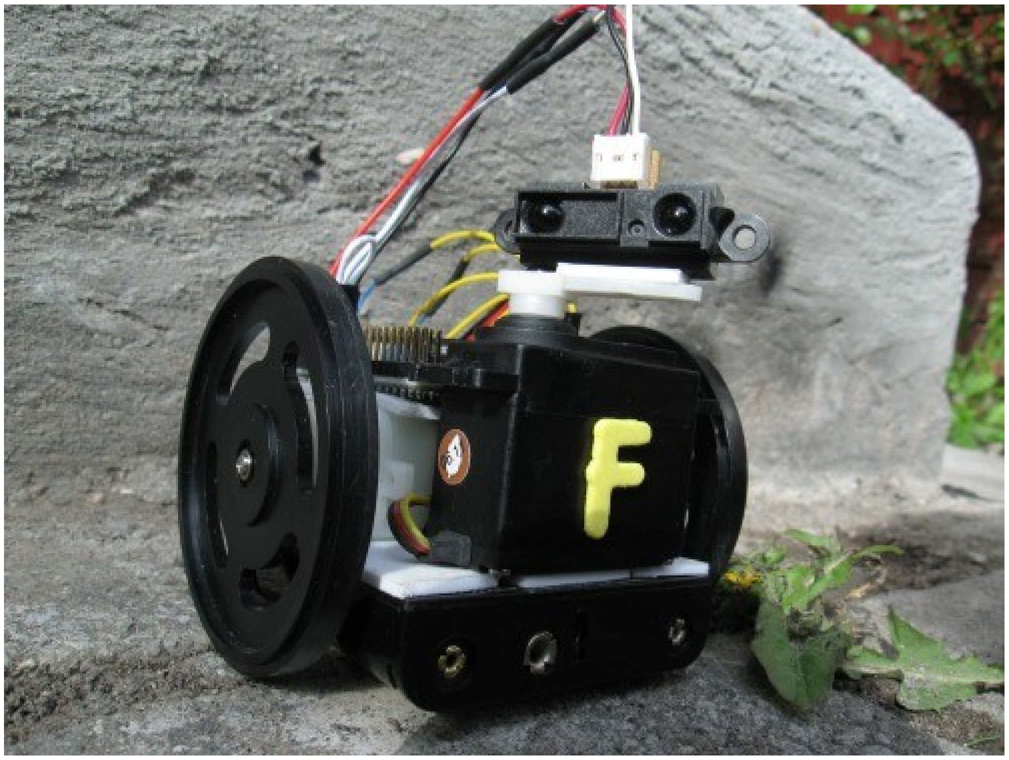
That should make a sound and turn something on, make a new sound and turn it off again.

Or the more interesting, make sure the Sharp is still in, hook a speaker up to pin 1, and then program this:

```
noise:
readadc 0, b0
sound 1, (b0, 2)
goto noise
```

Go to http://letsmakerobots.com/start to download a program for the robot

Ce ya!



PDF created by servello

(feel free to send comments or corrections)

Let's Make Robots

ALL LMR ARE BELONG TO US!