How to make your first Robot

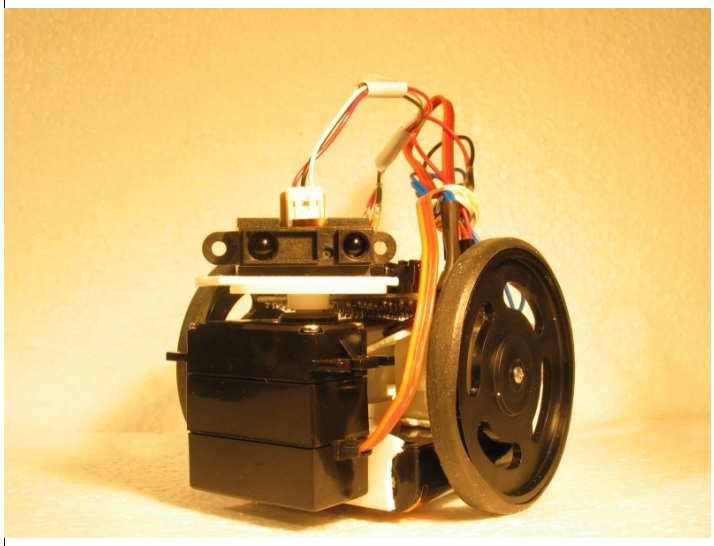
By Frits “fritsl” Lyneborg
http://letsmakerobots.com/user/4

PDF by Ajster

Find out more on:
http://letsmakerobots.com/

Contents Page

Introduction



Time to build: 2 hours
URL to more information:
http://video.google.com/googleplayer.swf?docI...

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

This project requires no knowledge of electronics, and it will not get much into that, it will only start you up with a robot really fast.

It is based on a system called Picaxe that is really simple yet powerful.

If you know that you want to get more into the electronics part of building, spend more time on learning, and use the system called Arduino, you should check out this First robot-project that uses Arduino and a breadboard.

Before you Start

Welcome. before we get started, here are some things you should know:

For your own sake, read this before you go any further

If you are looking for an even cheaper and faster project, this one might have your interest: http://letsmakerobots.com/node/87

If you have never build a robot before, the video here might inspire to how easy it is to construct, you do not need special materials to build robots: http://letsmakerobots.com/node/35

To inspire to where you can go from here, I have made a part II

OK - Now let´s get started :)

How to start

How to start a cool* robot for approx $85

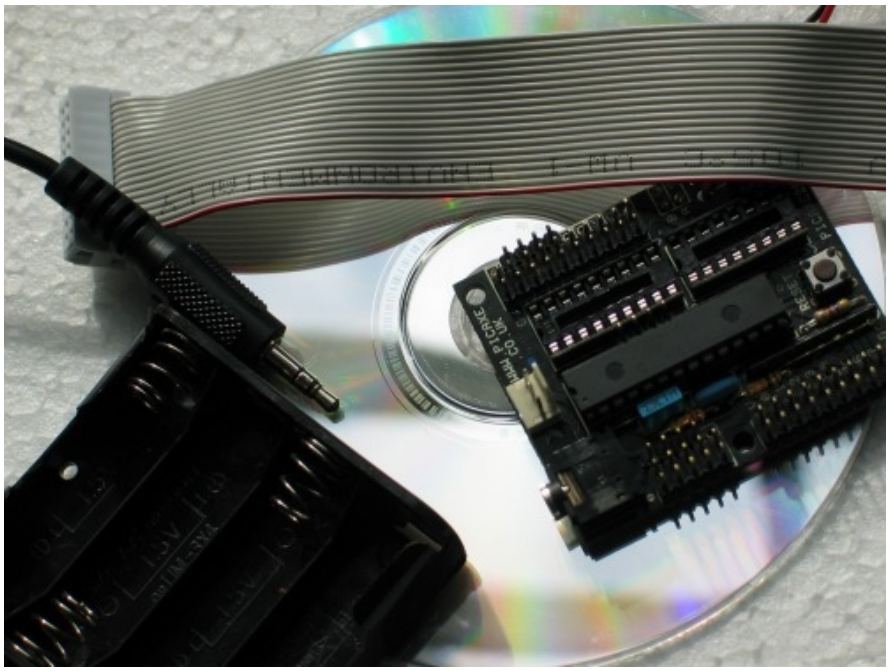
This is cool* because:

- The electronics used are ”real parts” (not little home-made things that wont really work unless you spend hours of tweaking, and not a kit that you just assemble and that´s it).
- It is EASY to do the basics, you have a robot within one hour!
- You can evolve from here, even with the same parts (if you can bare to take your robot apart).
- It is cheap.
- This is serious, but fun. This is the coolest Robot-beginners-project in any way, end of story! :)

Items Needed

Links are just where I happened to find the items from a world wide web perspective. You can use any (web) shop you'd like, of course.
Please see this page for "where to get Picaxe"

Prices are approx. As far as possible, try to get it all from the same shop, and from a shop located in your own country; to get the best deals and faster deliverance etc.



1x PICAXE-28X1 Starter Pack

The 28 pin project board in this package is like a game of Mario Bros; Fun and full of extras and hidden features, making you want to play over and again. This includes the main brain, the PICAXE-28X1.

This is a little expansive, but it is only the first time I recommend you to get this, it includes a lot of nice basic stuff, you get a CD-ROM with lots of manuals, cables, a board, the Microprocessor etc. Actually it is EXTREMELY cheap. Similar packages cost up to 10 times this price!

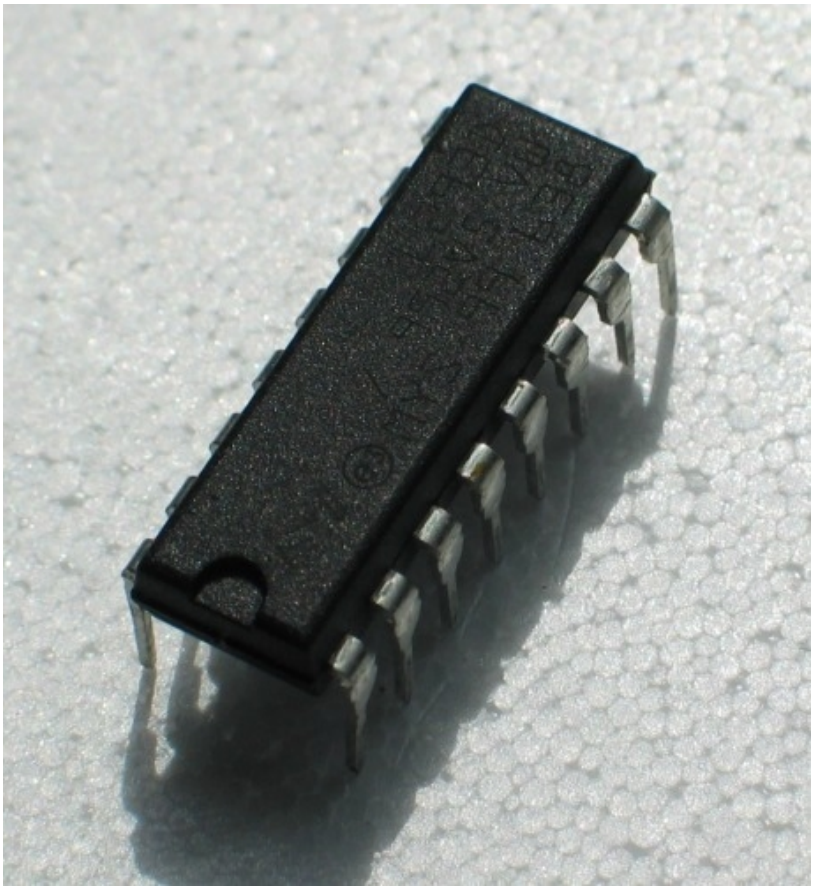
Be sure to get the USB-version, images in the shops may not match, and show a serial-cable when you are ordering a USB. When buying the USB-version, it is not necessary to get the USB-cable as an extra item, even though it is also sold separately.

You can get it here

Once you have bought this one time, just buy a new board and accomplishing Micro-controller for future projects, much cheaper, you are a Robot-builder with all the basics done.

Items Needed 2

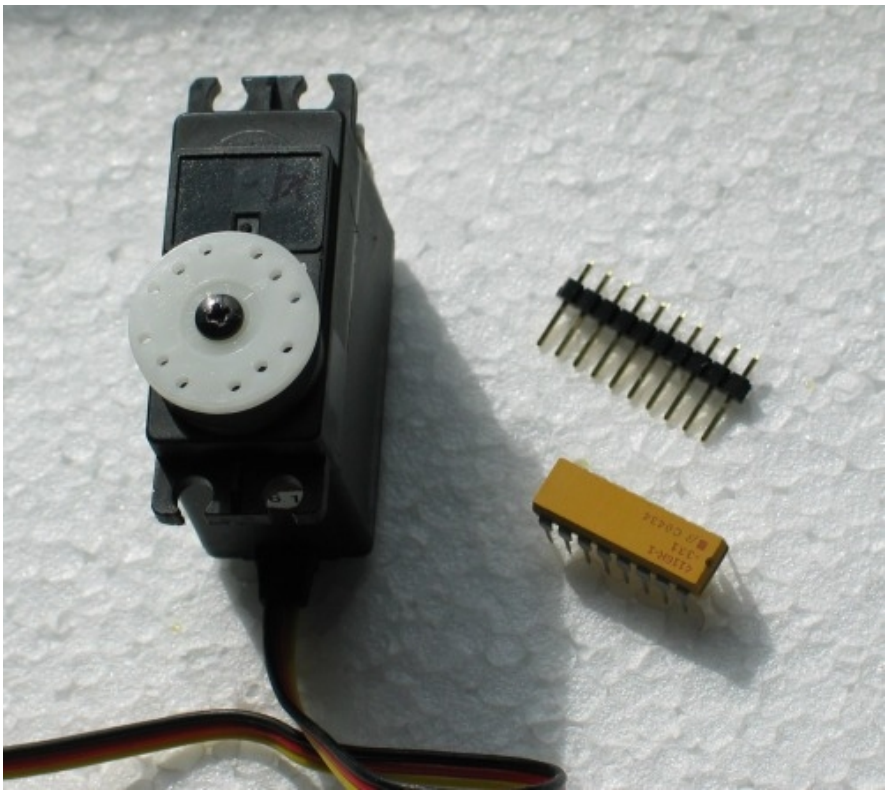
To connect things smooth, you may also want to invest in a lot of female to female jumpers like these. I recommend getting a lot, and I recommend it strongly. However it is not necessary to build this. But they are so nice to have.



1x L293D Motor Driver

The name says it all, more about this chip later :) You can get it here

Items Needed 3



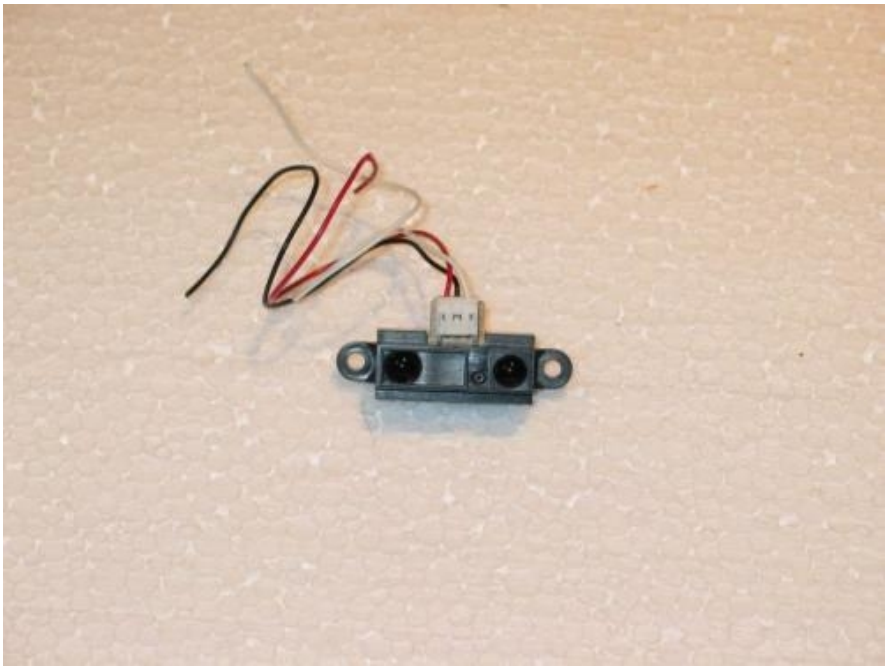
1x PICAXE Servo Upgrade Pack

An easy way to get a servo topped with some small parts needed for this project. You can also get any standard servo, the pins shown on the image, and a single 330 Ohm resistor instead of the yellow chip, if you should wish.
You can get the full package here

What is a Servo?

A Servo is a cornerstone in most robotic appliances. To put it short it is a little box with wires to it, and an axle that can turn some 200 degrees. on this axle you can mount a disc or some other peripheral that comes with the servo. The 3 wires are: 2 for power, and one for signal. The signal-wire goes to something that controls a servo, in this case that is the micro-controller Result is that the micro-controller can decide to where the axle should turn, and this is pretty handy; You can program something to physically move to a certain position.

Items Needed 4



1x Sharp GP2D120 IR Sensor - 11.5" / Analogue

11.5" or another range will do. Only do not buy the "Digital version" of the Sharp sensors for this kind of project, they do not measure distance as the analogue ones does.

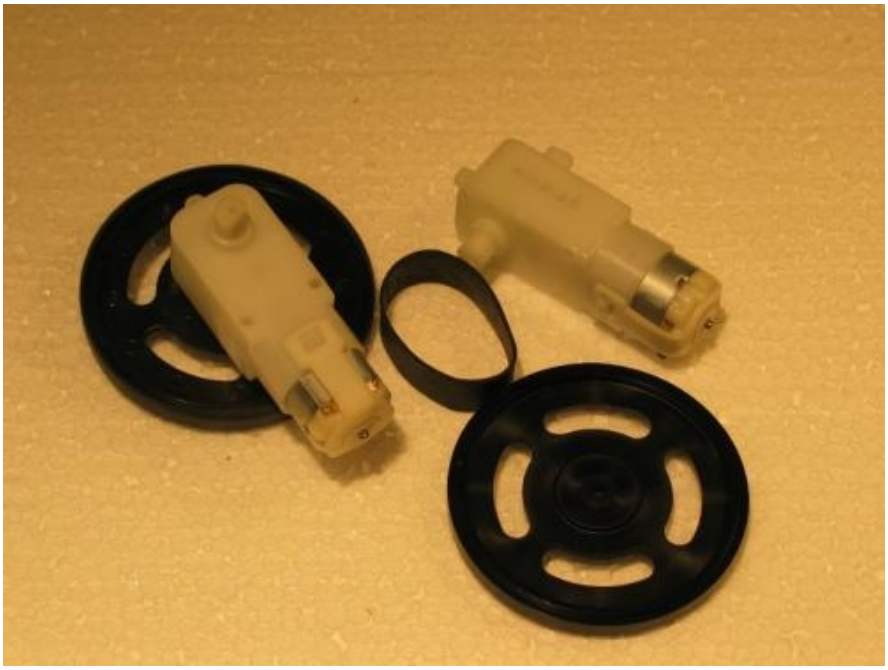
You can get it here

Be sure to get the red/black/white wires for it. This is not always included, and it is a non-standard socket!

This is actually not a favourite of mine, I usually use ultrasonic sensors, such as the SRF05 (they also sell it at the picaxe-store where they call it SRF005 and have a picture of the back of an SRF04 in the shop! But it is the right one, and I did tell them but..). Anyway; The SRF05 is much more reliable and precise. It is also faster, but costs a little more, is a little more complicated to write code to, and a little more complex to install - so it is not used here, but if you are fresh, buy one of these instead ;)

If you go for the SRF05,
I have made a small walkthrough to connecting the SRF05 here

Items Needed 5



2x Gear Motors with wheels

The higher the ratio, the stronger robot, the lower, the faster robot. I recommend ratio somewhere between 120:1 to 210:1 for this kind of project. The reason the robot on the video is so slow, is that is has a high ratio. Slower is easier or beginners, as it it easier to understand and follow what happens.

Price, total for 2+2: 15 USD You can get some here

Items Needed 6

You will also need:

- Double sided adhesive tape (for mounting, the foamy sort is best)
- Some wire
- Ordinary adhesive tape (to isolate a cable perhaps)
- Simple soldering equipment (Any cheap kit will do fine)
- An ordinary small nipper or scissor to cut things
- A screwdriver

You could also get, while you're at it:

- Some LED's if you want your robot to be able to signal to the world or make cool flashing-effects.
- More servos to make your robot move more..erh..arms? Or servos with servos on etc.
- A tiny speaker if you would like your robot to produce sound-effects and communicate to you.
- Some sort of belt-track system. Robots with belt tracks are way cool as well, and the controller and the rest will be the same.
  Here is an example to what you could take it to with belt tracks TAMYIA makes cool belt-track-systems, and this one is also a favorite of mine
  Here is more info on belt-tracks.
- Any kind of line-sensor-kit, to turn your robot into a Sumo, a Line-follower, stop it from driving off tables, and everything else that needs "a look down".

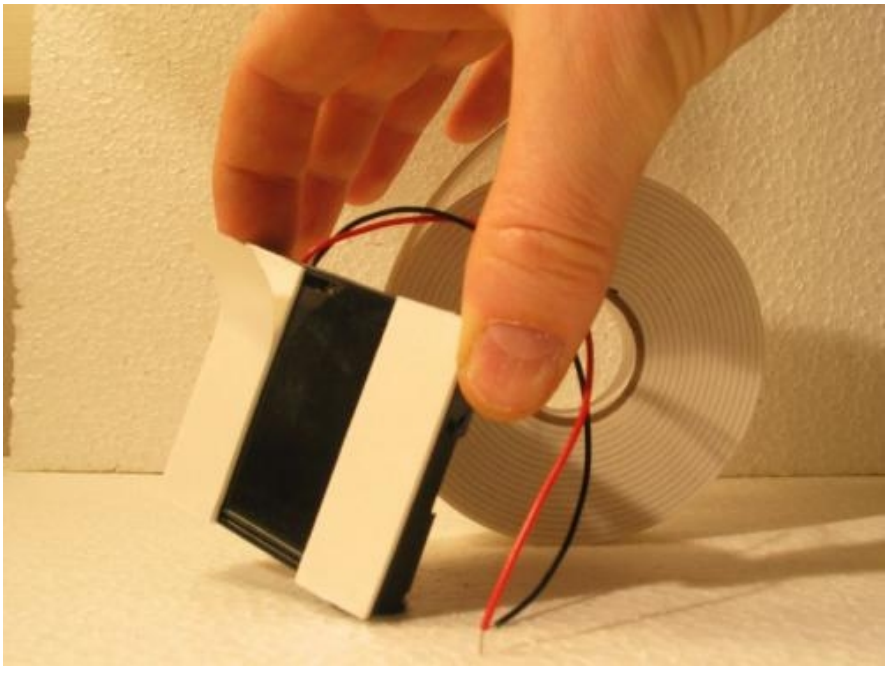
How to find Manuals for the Picaxe products

Getting Started

OK! You have ordered the stuff, received your package(s), you want to build :) well.. Let´s get started!

First mount the wheels to your geared motors. And add tires (rubber bands in this case).

An easy way to mount stuff for fast (and amazingly solid and lasting) robots is double adhesive tape.

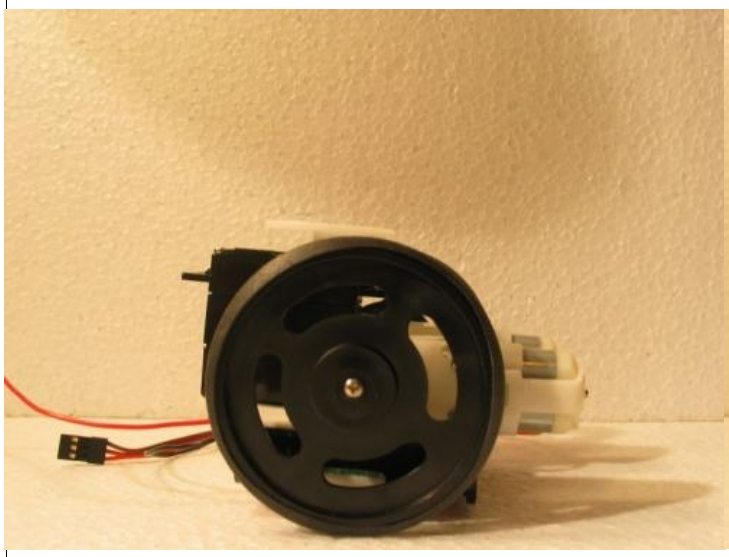


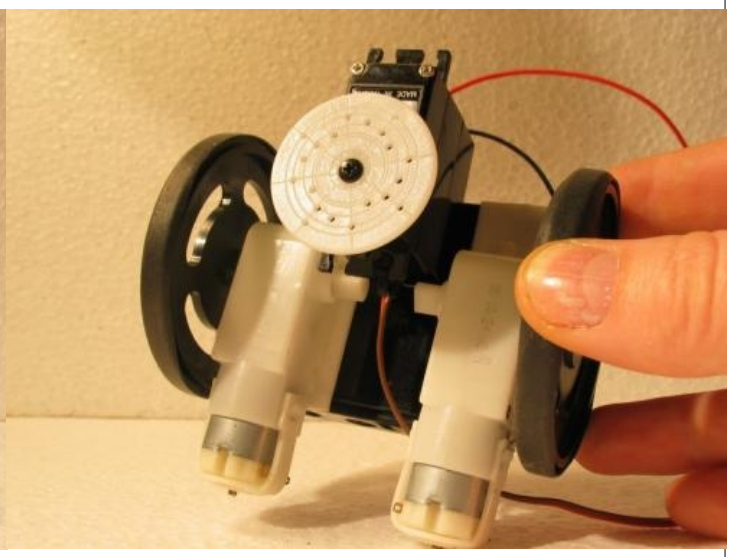
Sticking Together

Insert the batteries, so you have a realistic idea of weight and balance. Add some double adhesive tape to the button of the server as well..

Chose your own design, you can also add extra materials if my “design” is too simple.

Main thing is that we have it all glued together: Batteries, Servo and wheels. And wheels and servo can turn freely, and it can stand on it´s wheels somehow, balancing or not.





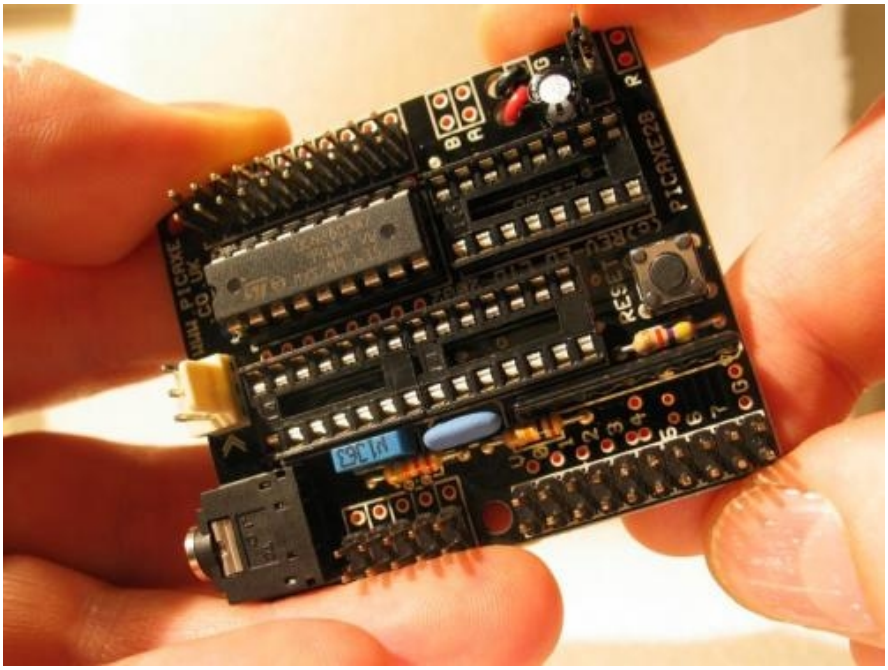
The Brains

Take out the batteries, to avoid burning something unintended!

And now for the brains. You should have a project board similar to this.

(and so this may be of interest to you: http://letsmakerobots.com/node/75)

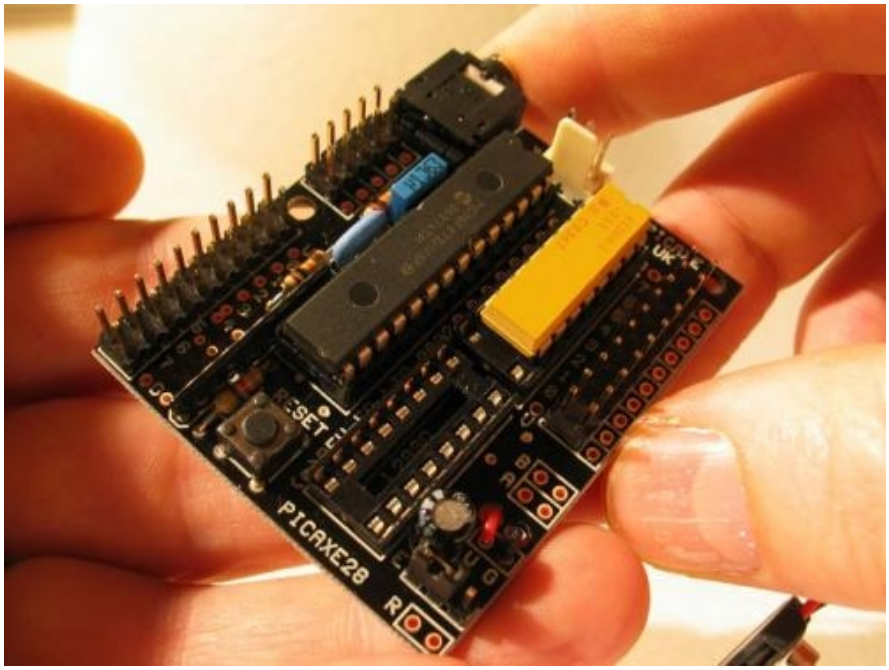
Notice that it has a chip in it. Take it out. The chip is a Darlington-driver that is quite handy placed there on the board, but we will not need it for this project, and we need it´s space, so away with that chip! It is easiest to get chips out of the socket by inserting a normal flat screwdriver just below it, move it in, and tip up the chip carefully.



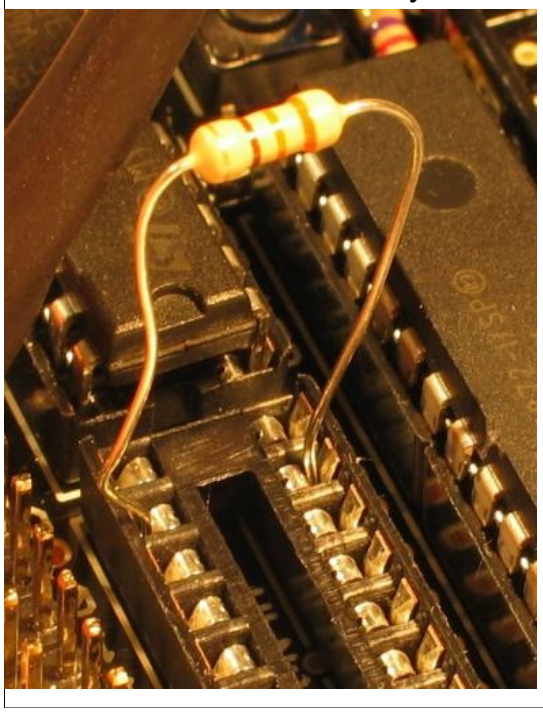
Project Board

A chip fresh, brand new chip usually do not fit into a socket right away. You will have to press it sideways down on a table, to bend all the legs in an angle so it will fit. (Legs go down, into the sockets).

Make sure all the legs are in the sockets. If you bought the Servo upgrade from Picaxe, you have a yellow chip. Put it in place of the Darlington. Note that not all holes in the project board are filled out with the yellow chip. We only need the eight to the right in the picture, as this is just simple resistors, we do not need to feed them extra.



This yellow chip is actually just 8 * 330 Ohm's resistors in a neat package. And so, if you should have a resistor, you can just insert it instead in slot numbered “0”, as this is the only one we will use, when we only use one servo.



Also insert the large chip, the brains, the micro-controller, the Picaxe 28(version number) into the project board.

Important to turn this the right way. Note that there is a little mark in one end, and so on the board. These must go together.

This chip will get power from the board via 2 of its legs.

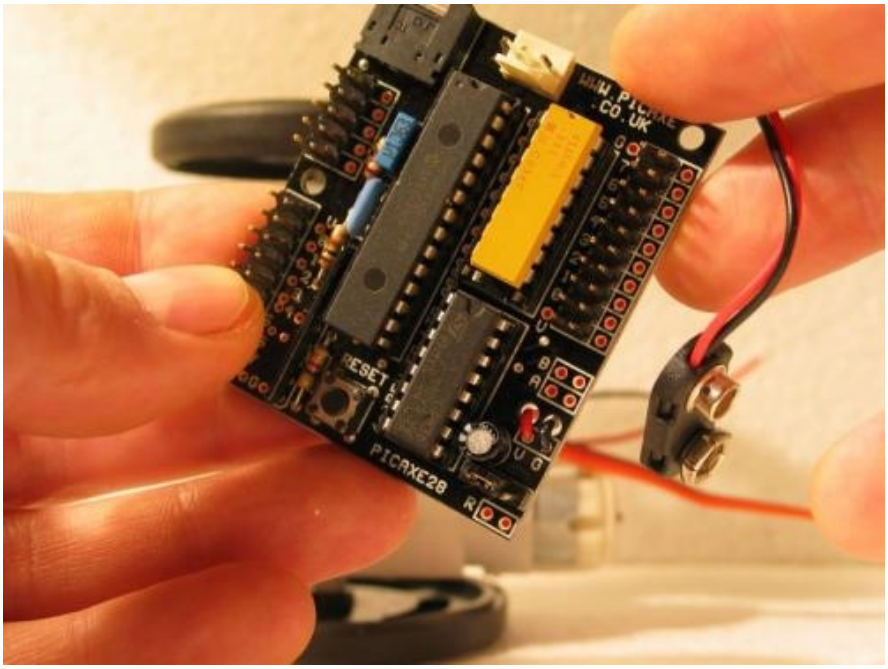
All the remaining 26 legs are connected around on the board, and they will be programmable for you, so you can send current in and out to detect things and control things with the programs you upload into this micro-controller

Project Board 2

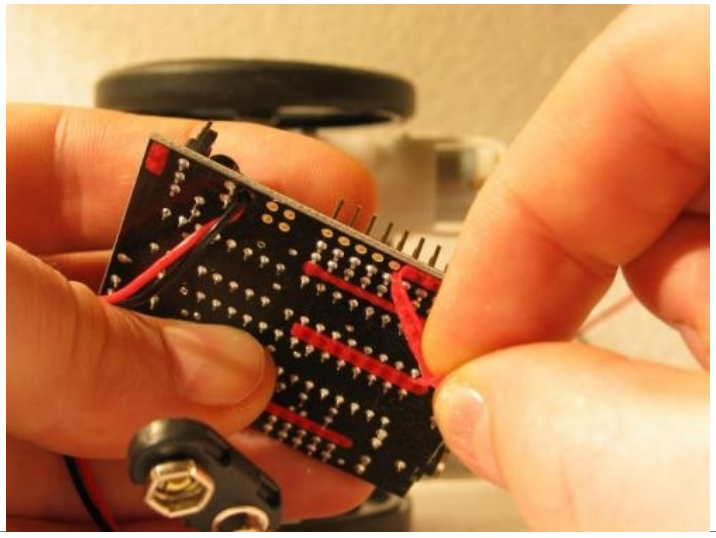
Now insert the L293D motor-controller.

This will take 4 of the outputs from the micro-controller, and turn them into 2. Sounds silly? Well.. Any ordinary output from the micro-controller can only be “on” or “off”. So just using these would (example) only make your robot able to drive forward or stop. Not reverse! That may come in unhandy when facing a wall.

The board is made so smart that the 2 (now reversible) outputs get their own space, marked (A) and (B) just next to the motor-controller (Bottom right on the picture). More about this later.

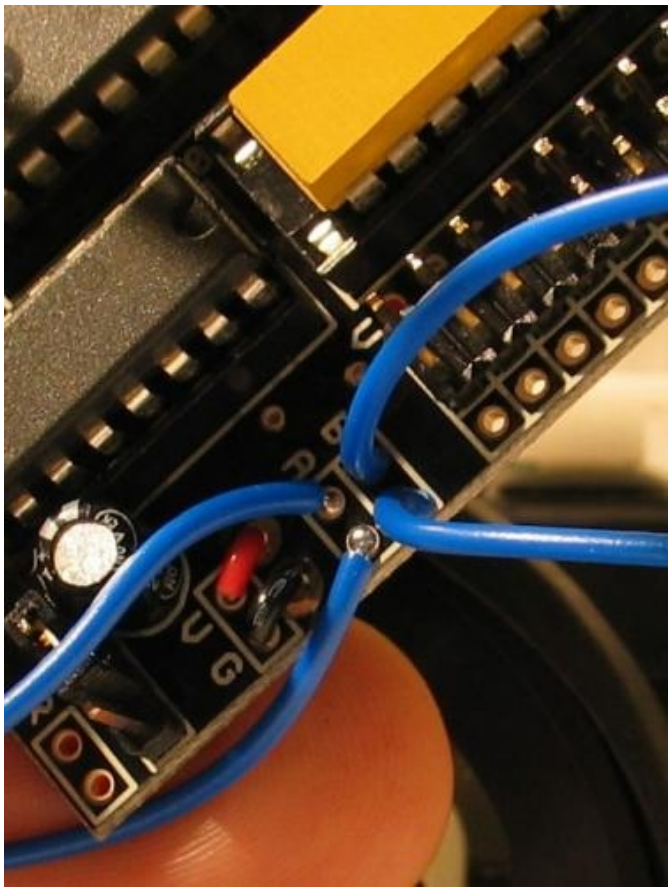


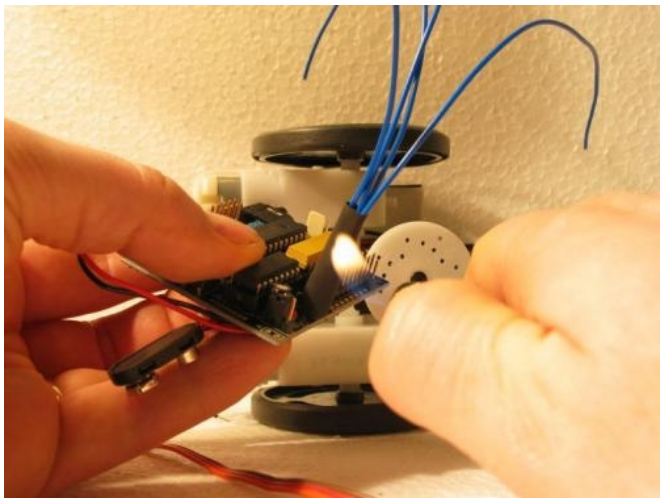
On the backside of the board you may find some strange plastic. This has no use, it is just a leftover from manufacturing. (They “dip” the board in warm tin, and parts they do not want so get tinned is sealed with this stuff) Just peal it off when you need the holes they seal.



Motors

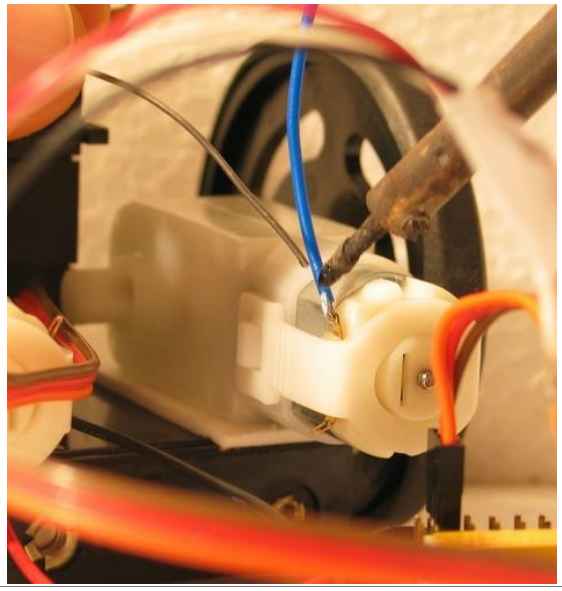
Take 4 pieces of wire, and solder them to the 4 “A & B” - holes. (or use some other means of connecting 4 cables to the standard sized holes, one can buy all sorts of standard sockets and pins etc.)





If you have some of that heat-shrinking plastic or some tape, it may be a good idea to support the wires with this.

The 2 “A” goes to one motor, and the 2 “B” to the other. It does not matter which is which, as long as “A” is connected to one motor, and “B” to the two poles of the other.



Servo

Now let´s hook up the servo.

If you should read the Picaxe documentation, you will read that you should use 2 different power-sources if you add servos. To put it short; We don´t mind here, this is a simple robot, and to my experience this works just fine.

You will need so solder an extra pin to output “0”, if you want to use the standard servo connection. Such a pin comes with the Picaxe upgrade pack (a whole row, actually), but you only need one for one servo, and they can be bought in any electronics store.

If your servos cable is (Black, Red, White) or (Black, Red, Yellow), the Black should be to the edge of the board. Mine was (Brown, Red, Orange), and so the brown goes to the edge.

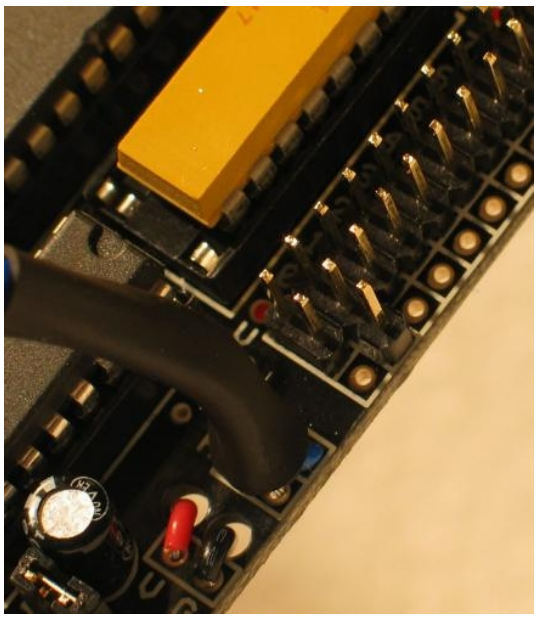
The hint is usually the Red; It is what is referred to as V, or any of these, used in random: (“V”, “V+”, “+”, “1”). This is where current comes from.

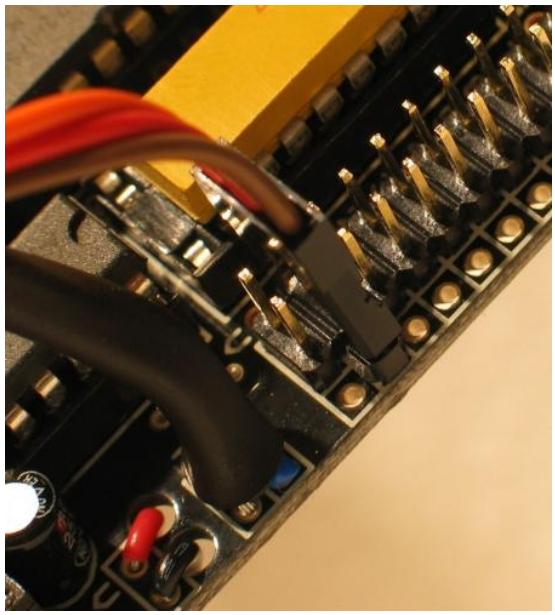
The black (or brown in my case) is G, or (“G”, “0” or “-”). This is also known as “Ground”, and is where current goes to. (the 2 poles, remember your physics-lessons?)

The last colour is then “the signal” (White, Yellow or Orange)

A servo needs both "+ & -" or "V & G", and a signal.

Some other devices may only need "Ground" and "Signal" (G & V), and some may both need V, G, Input and output. Can be confusing in the beginning, and everything is always named different (like I just did here), but after a while you will get the logic, and it is actually extremely simple - Even I get it now ;)





Sensor

Now let´s hook up “the head”, the Sharp IR-sensor.

(If you bought an SRF005 or similar instead, you should look here on how to hook this up, it is different from this!)

There are a million ways to do this, but here are clues:

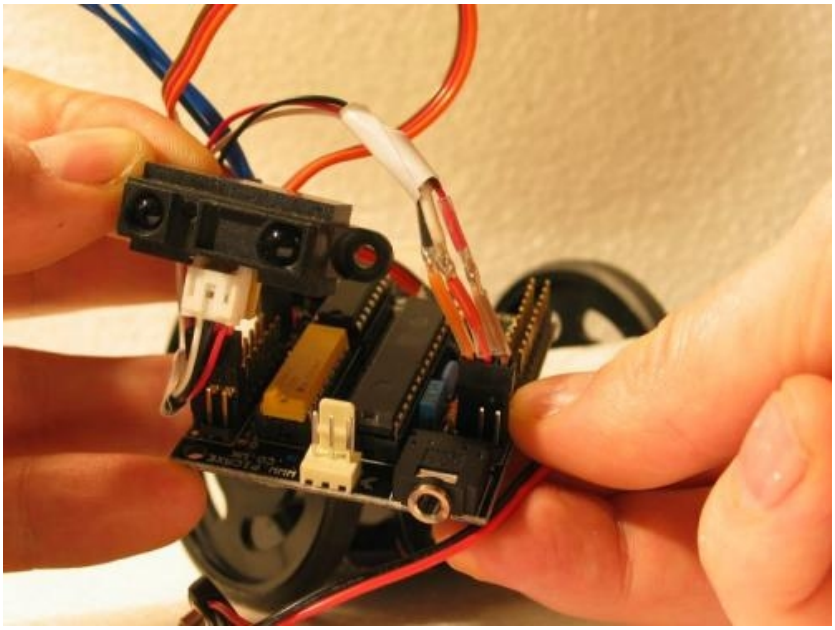
- Red needs to be connected to V1, that is (in this setup) anything marked “V”, or is connected to this.
- Black goes to G, anywhere on the board.
- White is to be connected to Analogue input 1.

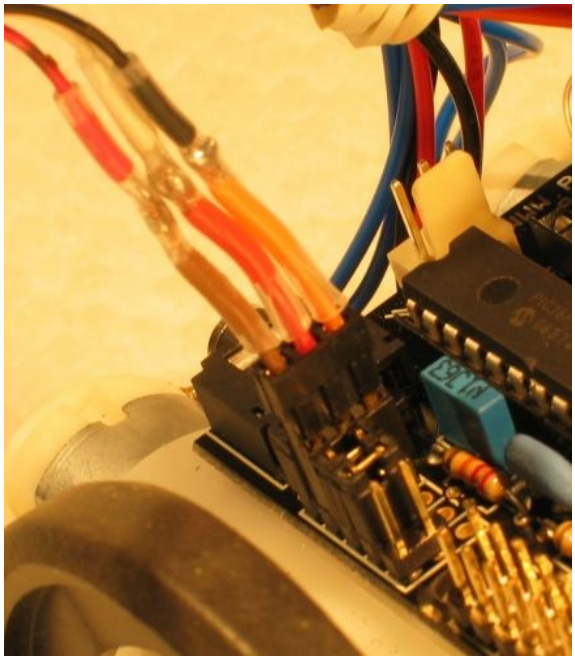
If you read the documentation that comes with the project-board, you can read how to attach the accompanying ribbon-cable, and use this.

What I have done on the picture, is to cut off a cable from an old burned out servo, soldered in a pin, and connected the whole thing just as a servo. You can use it to see which colours of the Sharp goes to which row on the board.

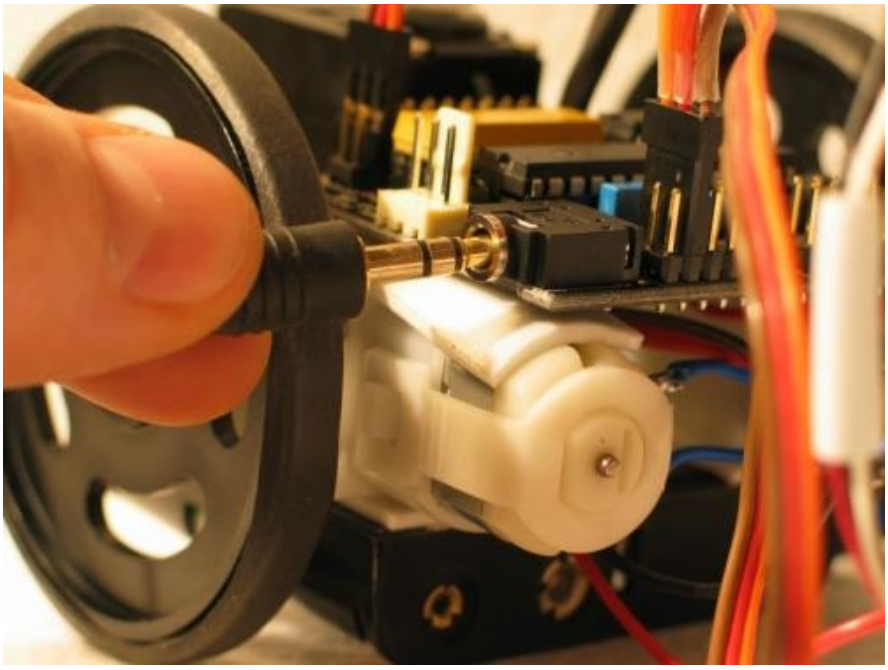
Weather you use the ribbons or “my method” of connecting the Sharp IR, you should also connect the 3 remaining analogue input to V. I had some jumpers laying, and you can see that all 3 connections left are short cut. (The last pair, not touched, are just two “Ground”, no need to short cut these). If you use the ribbon, you can just connect the inputs to V (or ground for that matter) by connecting the wires in pairs.

The reason it is important to shortcut the unused analogue inputs here is that the are “left floating”. This means that you will get all sorts of weird readings where you try to read if these are not connected. (to put it short, this is a fast paced walkthrough ;)





Let there be Life



Now for some fun! (Or "Let there be life")

Somehow you should get the Red wire from your batteries (+) hooked up to the red wire on the project board (V). And the black (-) to (G). How you do this depends on your equipment. If there is a battery-clip on both batteries and board you should still make sure that the "+" from the batteries ends up to the "V" on the board. Sometimes (though not often) the clips can be reversed to each other, and just putting two matching clips together is no guarantee that + gets to V and - gets to G! Make sure, or you will se melting things and smoke! Do not feed the board with more than 6V (no 9V batteries, even though the clip fits)

As a note; We are only working with one power-supply here. Later you will want to use same Ground, but both V1 and V2. That way your chips can get one source, and the motors etc. another (stronger) voltage.

Setting the Sensor

Install the Picaxe Programming Editor on a PC, follow the manuals to get your Jack / USB / Serial hooked up, Insert the batteries in your (still headless) robot, insert the jack stick in your robot.. enter the programming editor, and write **servo 0, 150 wait 2**

press F5, wait for the program to transfer, and your servo gives a little yank (or spins, depending on which way it was).

This code (**servo 0, 150 wait 2)** is actually "2 lines". Some people (and some editors) find it more correct to write it like

servo 0, 150 wait 2

because it is actually 2 commands. I, however, like to keep same "orders" in one line, and the Picaxe editor allows for this way of editing.

The "Wait 2" is not really needed, only it is :D Reason is that normally you let your program run around in loops, and the picaxe "stays alive". However here, in the testing phase, your program reaches the end, and stops.

This makes the Picaxe execute an "end" command, and this makes the servo-command stop before it really got to do much, as it happens just after the servo-command. And the servo needs continuously to be fed the pulse from the Picaxe, or else it will do nothing, it has no brains.

Therefore, just for now, you need to tell the Picaxe to wait in 2 seconds.. or something.. so it will send out signals to the servo BEFORE it shuts itself down :)

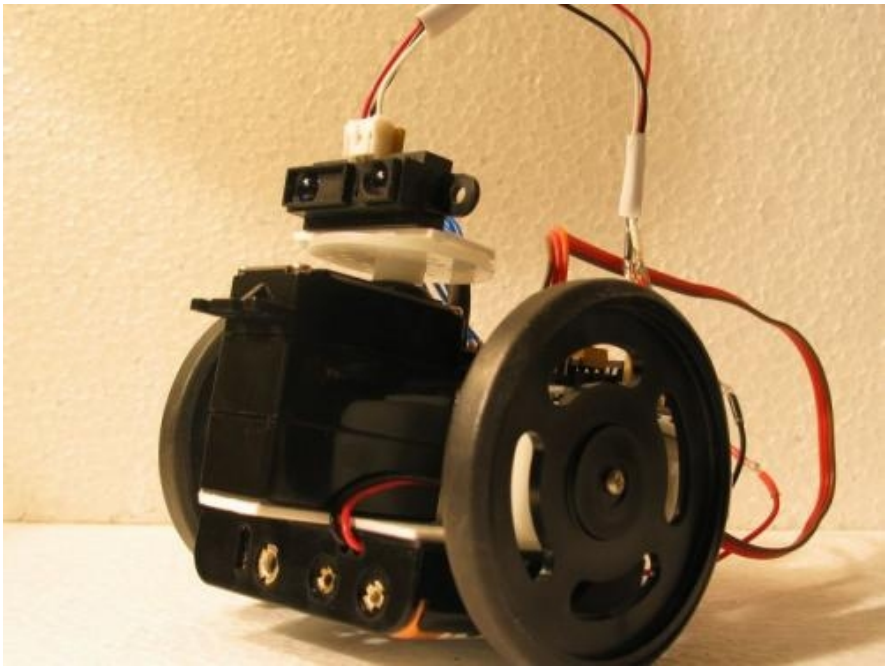
All this "wait 2" is actually added later to this post, as there has been some changes in the Picaxe's that made it necessary.

Something else that has happened since I originally wrote this tutorial is, that a new command has been introduced, and you should read the manual and look for the command "servopos" - it solves many problems that you might bump into.

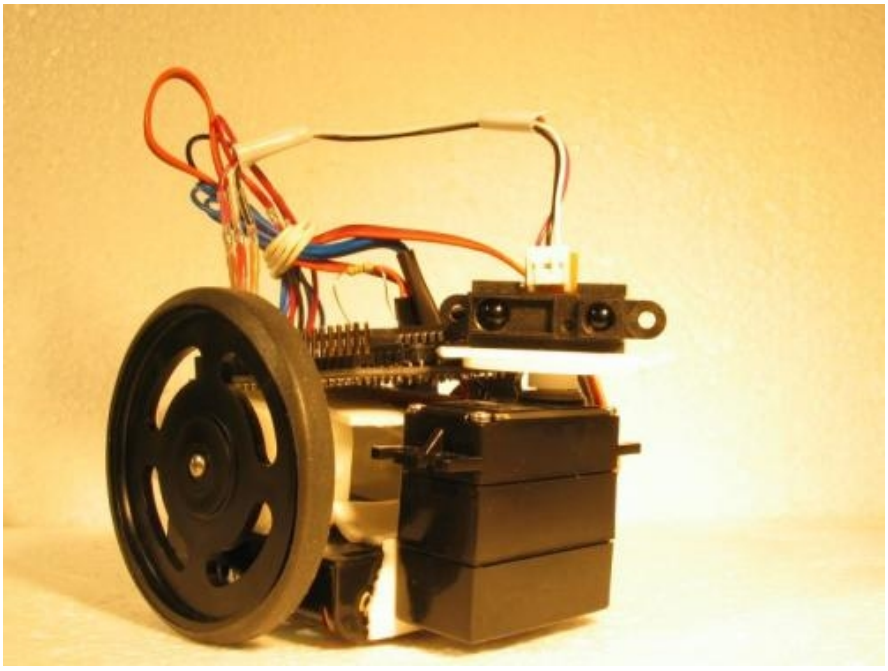
If something goes wrong here, you need to mess with the manuals and ports, power supply etc., until no errors are reported, and all seems to work, To test, try to write **servo 0, 200 wait 2**

and press F5 The servos disc should spin a little and stop. To get back, write: **servo 0, 150 wait 2** and press F5 Now your robot's “neck” is facing forward. Stick on the “head” - the Sharp IR

Fully Built



Hello world, I am a robot, ready to take your commands and explore the world :)
You're done building the basics!



The design may wary, you may have used other parts etc.. But if you have connected as described, on the next page are some tips to get started programming your robot.

Starting to Program

Enter (copy-paste) this code into your editor, and press F5 while the robot is connected:

+++

main: readadc 1, b1 ' takes the voltage returned to analogue pin 1, and puts it into variable b1 debug ' this draws out all variables to the editor.

goto main

+++

Now take your hand in front of the robot´s head and notice how the variable b1 changes value. You can use the knowledge gained to decide what should happen when (how close things should get before..)

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

Enter (copy-paste) 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

+++

Programming...

To turn the other wheel, you need to enter:

+++

high 6

low 7

+++

(or the other way around for opposite direction.)

The servo you have already tried.

All the way to one side is:

servo 0, 75 wait 2

the other side is:

servo 1, 225 wait 2

- and centre:

servo 1, 150 wait 2

On the next page is a small program that will (should, if all is well, and you insert 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.

```
+++
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

' … next page
```

' … continued

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 ;)

Look in part II for code on this.

Add some Extras

Sound:

You can also add a small speaker to example pin 1 & ground, and write

Sound 1, (100, 5)

- or within the example program above make it

Sound 1, (b1,5)

– to get funny sounds depending on the distance to objects ahead.

You could also attach a lamp or LED to pin 2 & ground, and write (remember LED´s 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 :)

Welcome to a very funny world of home-made robots, there are thousands of sensors and actuators just waiting for you to hook them up and make robots out of them :)

Remember to share with us what you have made, or perhaps is struggling to make, what you learn etc. :)

You can also take a look here: Part II if you have even read this far ;)

How to find Picaxe Manuals

I think Picaxe is a great product.

But somebody should kick their web department. Sometimes people buy a Picaxe product in another shop, and do not get the manual on print. "Find it on the web".. yes, but where? They have a really crap web at Picaxe, a mix of info, links and URL's not really well made at all

I OFTEN get questions about the manuals, so something is wrong!

This is the way I go about it, it could be done simpler with more direct links, but here is how I tend to do it, because I can remember this, I just have to remember step 1, and the rest comes naturally to me after many visits:

picaxe.co.uk -> Top menu, "DATASHEETS"

Main thing is; It is all very well documented, it is sometimes very hard to find, but it is there, from the page picaxe.co.uk :)

Anyhow - that is the manuals for the boards, chips etc. Something else that you want - this is very important:

Picaxe main manuals:

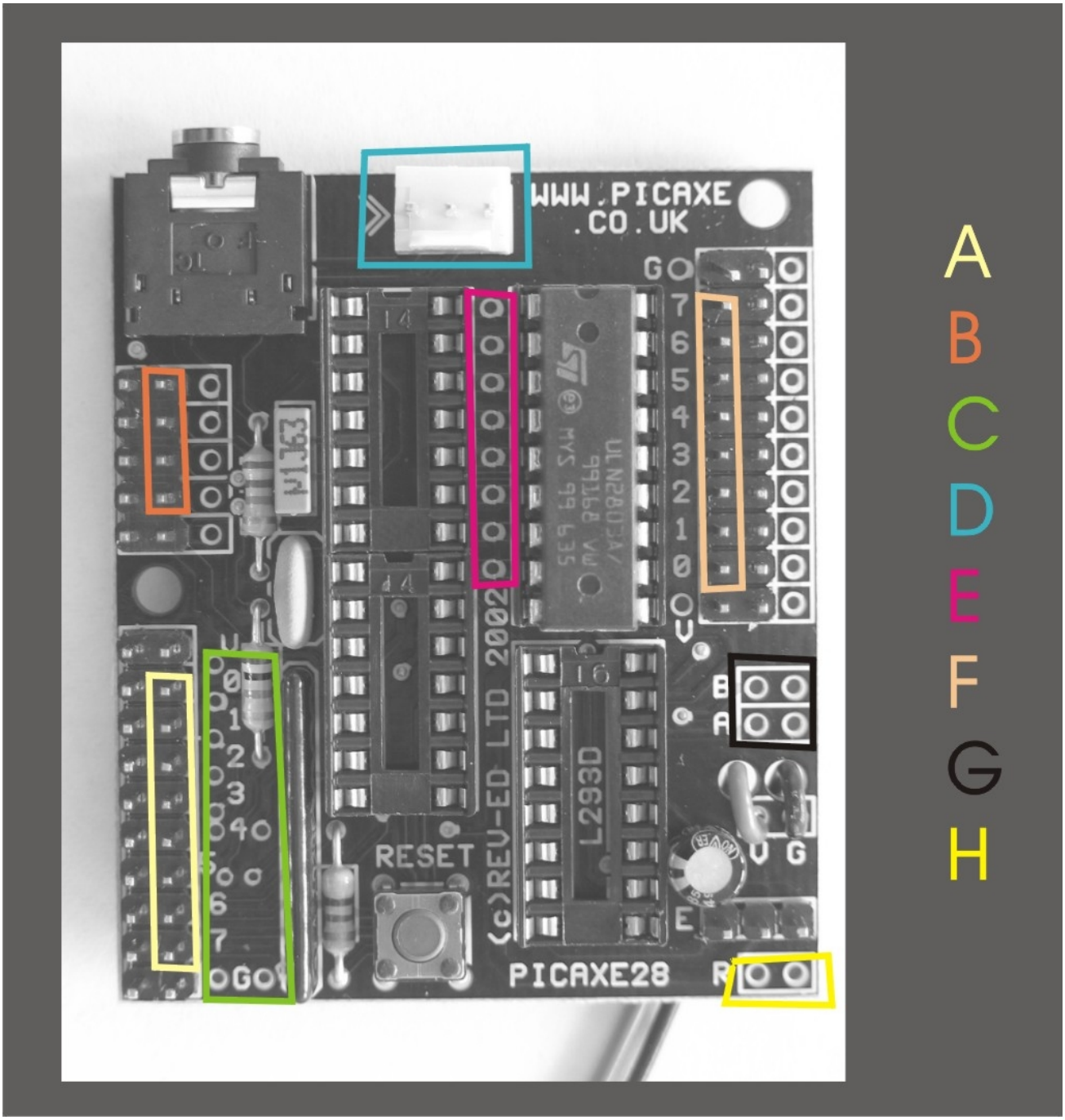
Section 1 - Getting Started

Section 2 - BASIC Commands

Section 3 - Interfacing circuits

Specially Section 2 - that is your friend in the everyday work with Picaxe.

28 pin Project Board (AXE020), Picaxe for dummies



Ground!

Often you get something that you want to connect, and the documentation says connect this to "Ground", or any other name often used in random. It has many names, that are the same thing:

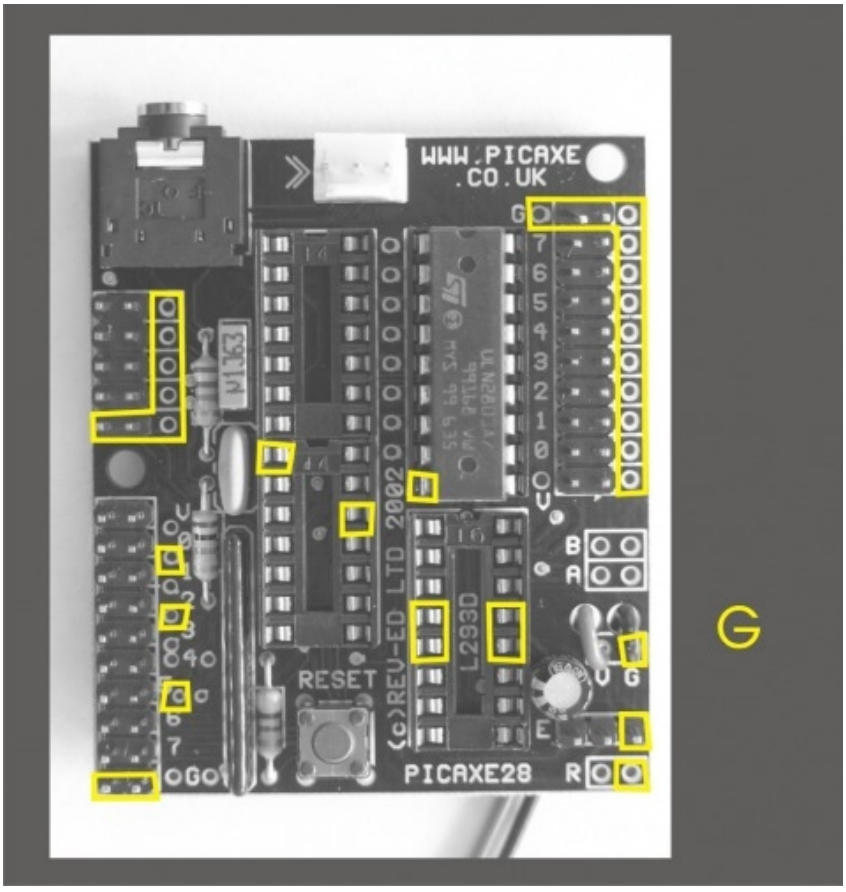
- Ground
- Usually "the black wire" if there is a red and a black
- G
- GND
- -
- 0
- Vee
- Vss
- Negative Supply

- All the same!

This is simply where current goes to when it comes from the other pole on the battery (the positive, +).

When something can be connected to your board using "just a single wire", it always also has to have a second wire, connected to this ground.

If you want to hook up to ground, you can use any of these connections on the board:



V

Also referred to in random as :

- Positive supply
- "the red wire"
- +
- Current
- Power
- VCC
- V+
- B+
- VDD

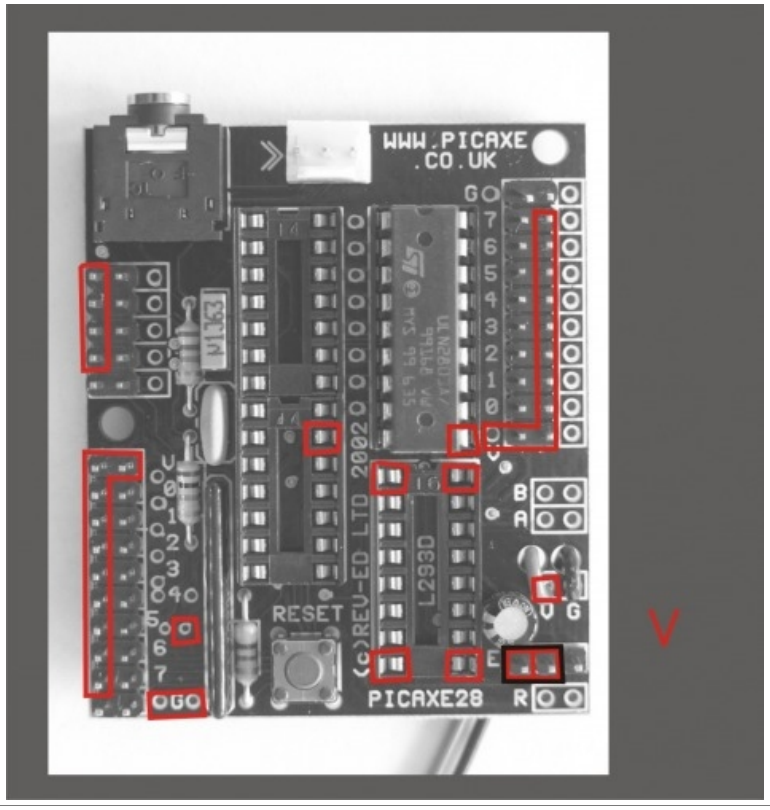
Actually the reason for having a Micro-controller on a board is, that you should let the micro-controller control when you send current to something (turn it on or off).

However, quite often, you will need to feed peripherals with current (and ground, this goes unsaid), on top of what you send to it via the micro-controller

Perhaps you are connecting a Sharp IR range finder, it has 3 wires: Black, White and Red. The black is Ground. The Red then will be V, to power the module, and the white will be a signal returning from the Sharp module to the micro-controller on your board.

Perhaps you want to control a Servo. That will also need a G and a V to power it´s circuit and its motor. And on top of this, it needs a pulse-signal from the micro-controller, the last wire - what colour it may be on the particular model.

The standard setup of the board looks like this, all these areas are V:



V1 & V2

So - what´s with the V1 / V2?

Well, everything returns to ground. But it can come from different sources. And they can have different voltage.

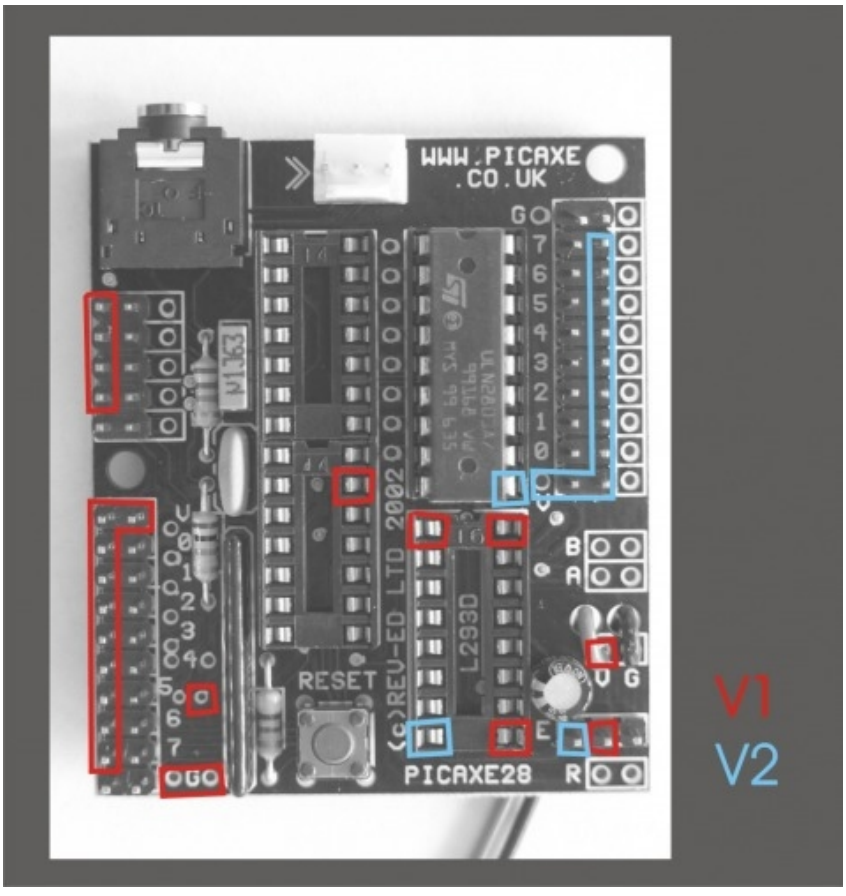
This way you can power a servo with V2, while the pulses that signals to the servo where it should turn to, is from the Micro-controller that runs on V1. All to the same G.

This 28 pin Project Board has 2 options:

Standard: See illustration above. V1 & V2 are connected, result: There is only one V. As you may have read from the documentation: It is the little jumper in the corner that connects V1 & V2, and when it is on, there is only V & G on the board.

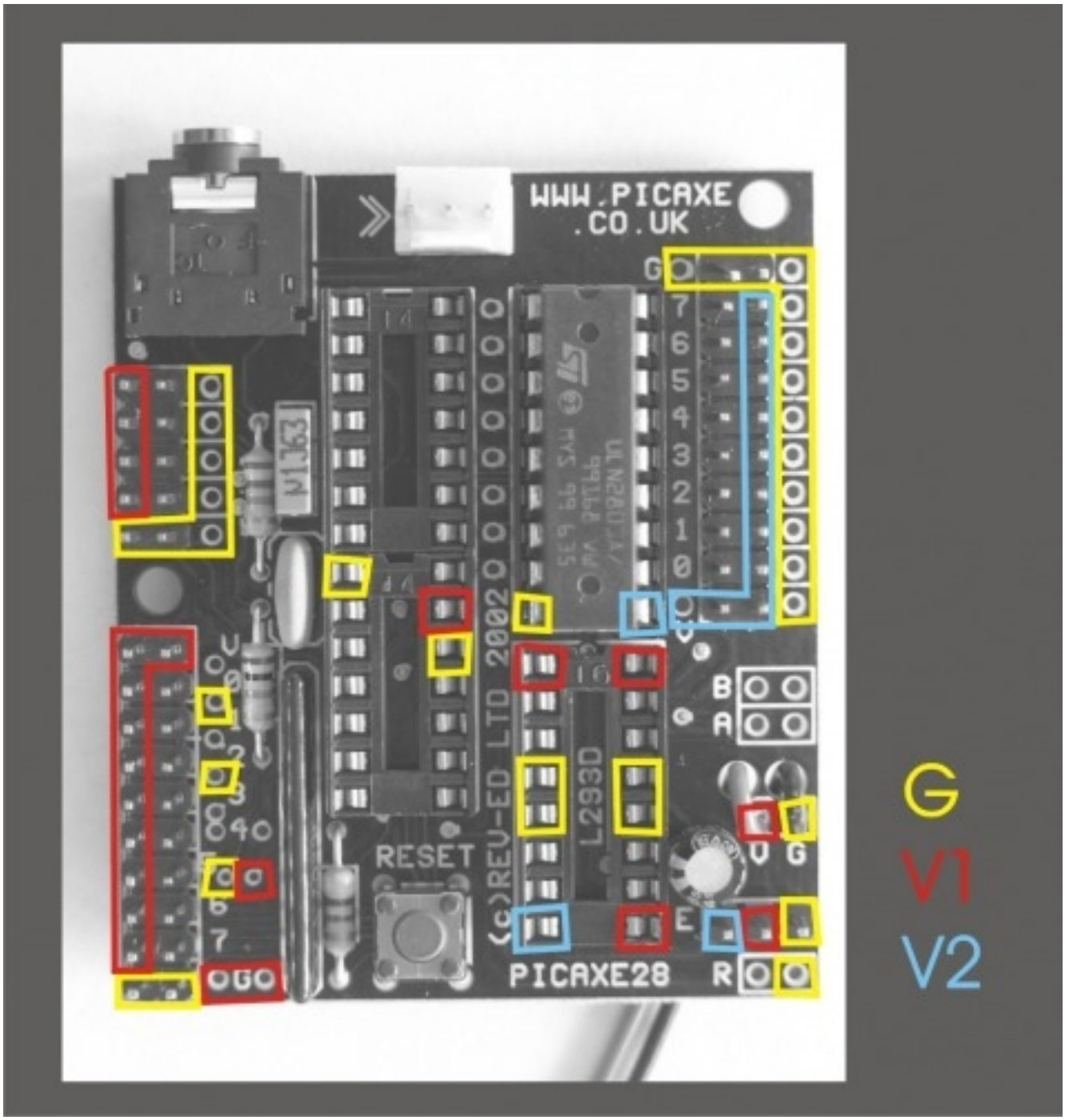
2 power supplies: See illustration below. Take off the jumper, and connect a separate power-supply to V2 (and that power supplies G to ground, goes unsaid from now ;) - and the board will have V1 to the circuits and inputs, V2 to power external stuff, and G for all of it.

That way you can feed the chips with 5V, and the motors with 12V that would otherwise kill the chips.



Power on the Board

So if you hook up a board with 2 power-supplies, this is where you have current and ground:



Peripherals

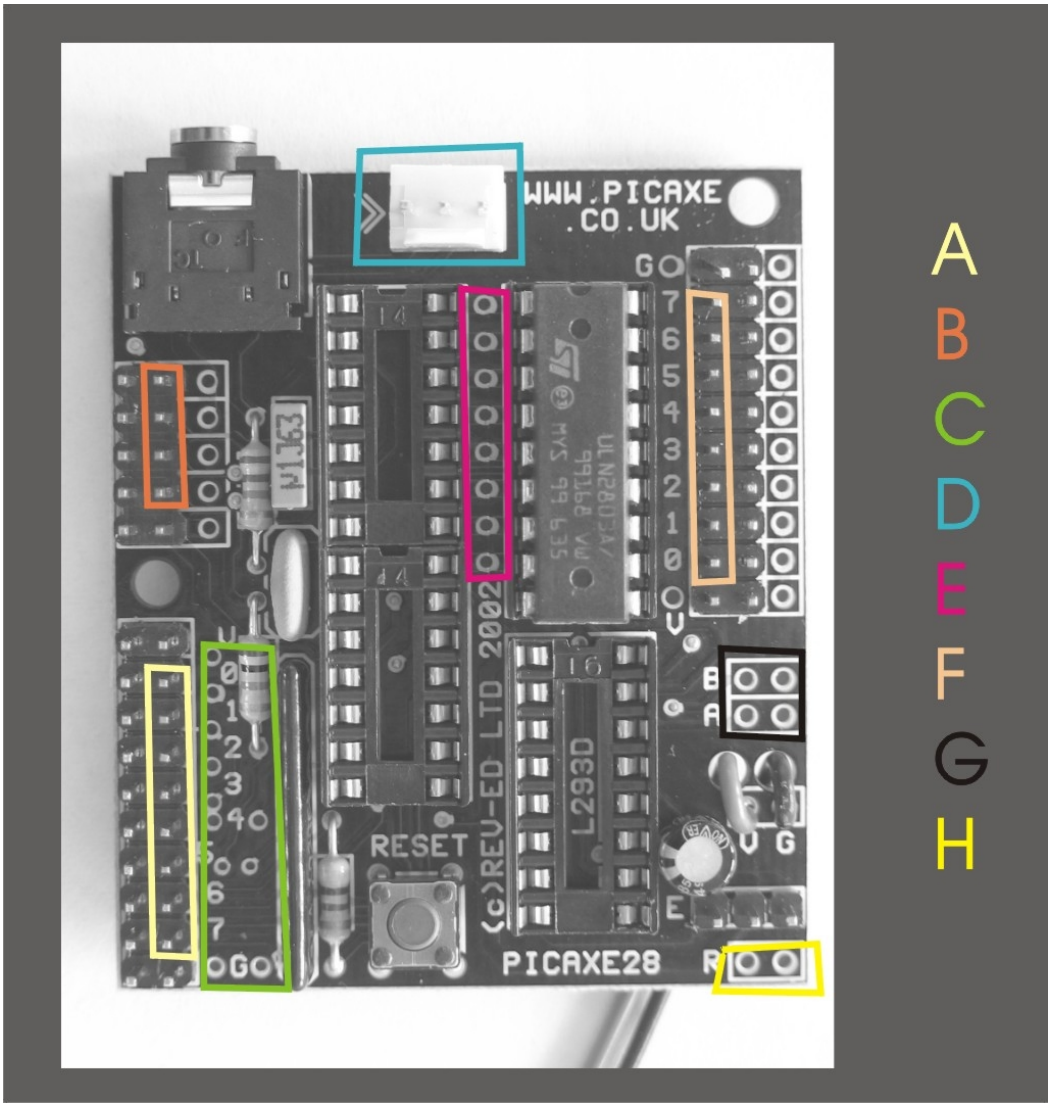
.. What is left?

Only the interesting stuff! :)

Power they need, the peripherals.. But it is the inputs and outputs that are interesting.

None of the above connections can be controlled by the Micro-controller, and unless you burn something, the Micro-controller does not sense weather you connect something to these or not.

This is where the input's and outputs come into the picture:



These connect to the Micro-controllers legs. Quite often, for advanced usage, things can be altered; An input can be switched in mode, so that it is an output etc. But you will figure that out, this is just the basics, as things are wired in default.

A, B, C

A: The digital inputs:

Connect a switch to V1 and one of these. In the program editor you can then write (let's say you have connected to leg 4, there are 8 in this area)

If pin4 = 1 then

...

B: The analogue inputs

How much current from V1 (or another source not more powerful then V1) comes to this pin?

Well - it is not a question i ask, but what the pin asks :)

The answer is then returned into a variable, example, pin 2 out of the 4 in this area:

readadc 2, b1

Now the variable b1 is a value between 0-255, depending on how much current from V1 (or another source not more powerful then V1) comes to this pin.

C: The IR-area

This is a quite undocumented feature on the board! It is used if you want to use standard TV-remote control-protocols to send to your micro-controller

You can hook up IR ins-and out to anything, to measure IR, for instance to see if something is near or far, or white or black.. but you will have to do with analogue levels of signal and similar, you will not be able to transmit "signals" that can be translated to numbers such as TV remotes does. Most IR remote controls use 38 KHz sub-carrier, and this area let´s you hook up to that quite simply. You should only use it for that!

D, E, F

D: The Serial communication

The square on the picture should perhaps include the jack socket :) This is the serial connection used to program and more.

You can also use output- and digital input pins for serial connection, but the ones on D are hard-wired

E: Direct outputs from the picaxe

These should (only) be used if you are trying to communicate "chip to chip". I think this is called TTL-level, where you let the chips talk to each other via pulses. This should not be used to drive anything, but it should always be used if you want your chip to talk to others, like in this example:

http://letsmakerobots.com/node/66

F: Output pins

These are what you should start with, IMHO: Connect a LED or something (Remember LED´s should turn the right way around) to one of these (other LED-leg to ground).. let´s say you connect to leg 3 out of the 8 in this area. Write:

high 3

to turn power on

low 3

to turn it off

Note that if you are using a motor controller, you should not use leg 4, 5, 6 and 7 for anything else!

These pins are also used for standard servo connection.

G, H

G: Motor connections A & B

The board is made for the typical applications, and one is where you want to control 2 motors, and being able to switch directions on them.

Of course if you only can turn things on or off, it can be hard to reverse a motor.

You can purchase a L293D motor controller - chip, and place it in here.

Then read the documentation for how to make output 4, 5, 6 and 7 control 2 way-drive of 2 motors connected to A & B outputs. (one motor to A and one to B, not 2 times A+B)

H: The reset switch

If you mount your board in a case or something, and would like to be able to remote-reset the Micro-controller, add a switch to this, press it and.. Hello? I remember nothing, let´s start all over!

Something (slightly) interesting... or not.

You will most likely only see names like VCC, VEE, VSS and VDD on chip pins. If you see VCC and VEE it means that the chip is made out of bipolar junction transistors (aka BJTs) and if you see VSS and VDD it means that the chip is made out of metal-oxide -semiconductor field effect transistors (aka MOSFETs).

VCC means "Voltage at the collector pin of a BJT"

VEE means "Voltage at the emitter pin of a BJT"

VSS means "Voltage at the source pin of a MOSFET"

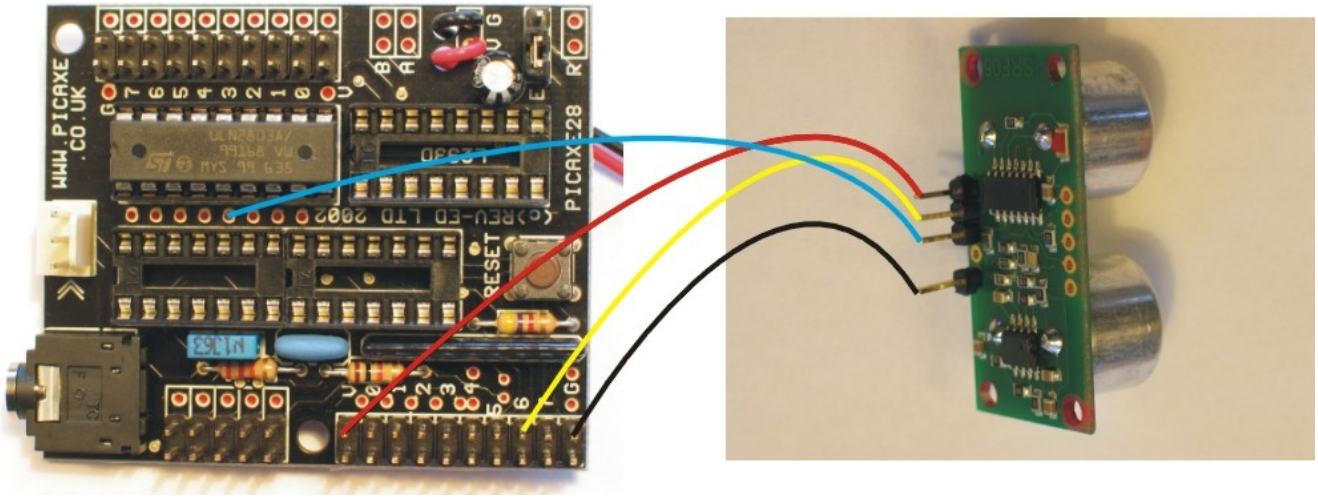
VDD means "Voltage at the drain pin of a MOSFET"

I know Frits won't really care about that information :-)

By jip

How to connect SRF05 to Picaxe 28 pin Project board

URL to more information:
http://www.rev-ed.co.uk/docs/srf005.pdf
Sensors / input devices: SRF05



Description:

If you are a newbie, you may only have learned that the Picaxe 28 pin Project Board has some inputs and outputs.. And you may have purchased an SRF05 (that Picaxe refers to as SRF005), and you are trying to connect the darn thing.. without result?

Reason is likely to be that you are connecting the output from the Picaxe to the SRF05 through the darlington on the board.

The SRF05 needs (like many other things) a Pulse-signal, not just a "power on / off". The signal therefore must come directly from the Picaxe chip.

Luckily the board has little "hidden" holes for that kind of connections. (Blue wire, the signal in to the SRF05 directly from the chip). These holes, however, are not described, simply omitted in the manual for the board. So I wonder how a newbie should ever find out, personally I burned a SRF05 or two in my frustrated attempts to get it right :)

If you follow the illustration above, you should be able to make the below code work just fine. You will know that the SRF05 is getting a pulse-signal when the little red LED on it´s back is flashing red.

SRF05 Code

****

```
symbol trig = 3 ' Define output pin for Trigger pulse
symbol echo = 6 ' Define input pin for Echo pulse
symbol range = w1 ' 16 bit word variable for range
main:
pulsout trig,2 ' produce 20uS trigger pulse (must be minimum of 10uS)
pulsin echo,1,range ' measures the range in 10uS steps
pause 10 ' recharge period after ranging completes
' now convert range to cm (divide by 5.8) or inches (divide by 14.8)
' as picaxe cannot use 5.8, multiply by 10 then divide by 58 instead
let range = range * 10 / 58 ' multiply by 10 then divide by 58
debug range ' display range via debug command
goto main ' and around forever
```

****

Notes:

I did not have a fresh SRF05, and so there are drilled extra holes and soldered pins on the one on the picture.

Be aware that if you also connect servo(s) to the same Picaxe chip, program execution may be bumpy with little irregular breaks when using things as the SFR05 that needs pulses. If it is important for you to have a steady program execution, or if a servo acts totally irrational, it sometimes helps to have a small pause or turn off a servos pulse for a short time in the code:

"low 3" or "pause 10". Apparently there is no system, it is just trial and error with your particular setup.

This pages might have your interest: See what Robot Sees