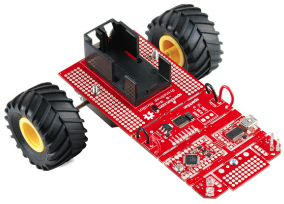
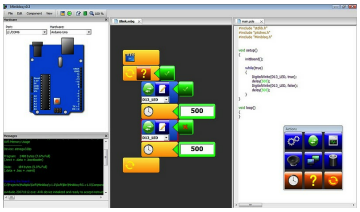
SparkFun's ProtoSnap MiniBot with Minibloq drag and drop programming

The Robot

Today you will be programming SparkFun Electronic's ProtoSnap MiniBot to perform very simple autonomous navigation. Below are the MiniBot and the Minibloq programming environment. There are four major parts of the MiniBot that you will be using today, all of these are described below.

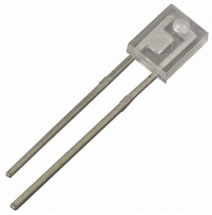


The ProtoSnap MiniBot

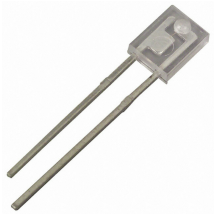


The Minibloq environment

The first two parts of the MiniBot you will be using are the infrared lights and infrared detectors. There are two pairs of these on either side of the front of the MiniBot. The infrared lights (or emitters) shine out infrared light, when this light bounces off of an object the infrared detectors can sense exactly how much of the light is shining off an object. This way the detectors can actually sense how close an object is to the MiniBot. The closer the object, the more light the detector detects. One catch though, the more light the detector senses, the lower the reading the MiniBot gets off the infrared detector.

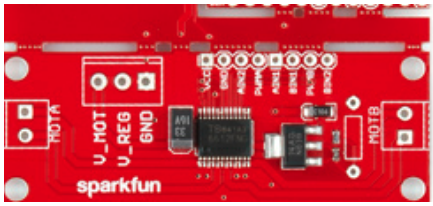


Infrared Emitter (yellow dot)

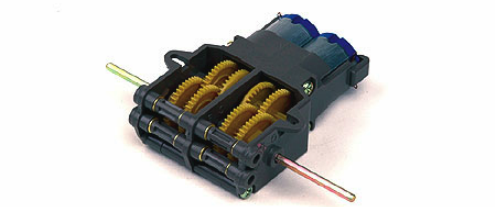


Infrared Detector (red dot)

The third part of the MiniBot you will be using is something called a “motor driver”. The motor driver is a way to control two different things on each of the two wheels. These are the direction the wheel turns, and the speed the wheel turns. Without the motor driver you would only be able to turn each wheel in one direction. With a motor driver you can go forward, backward, brake, or even spin in circles by making the two wheels turn in opposite directions.



ProtoSnap MiniBot Motor Driver

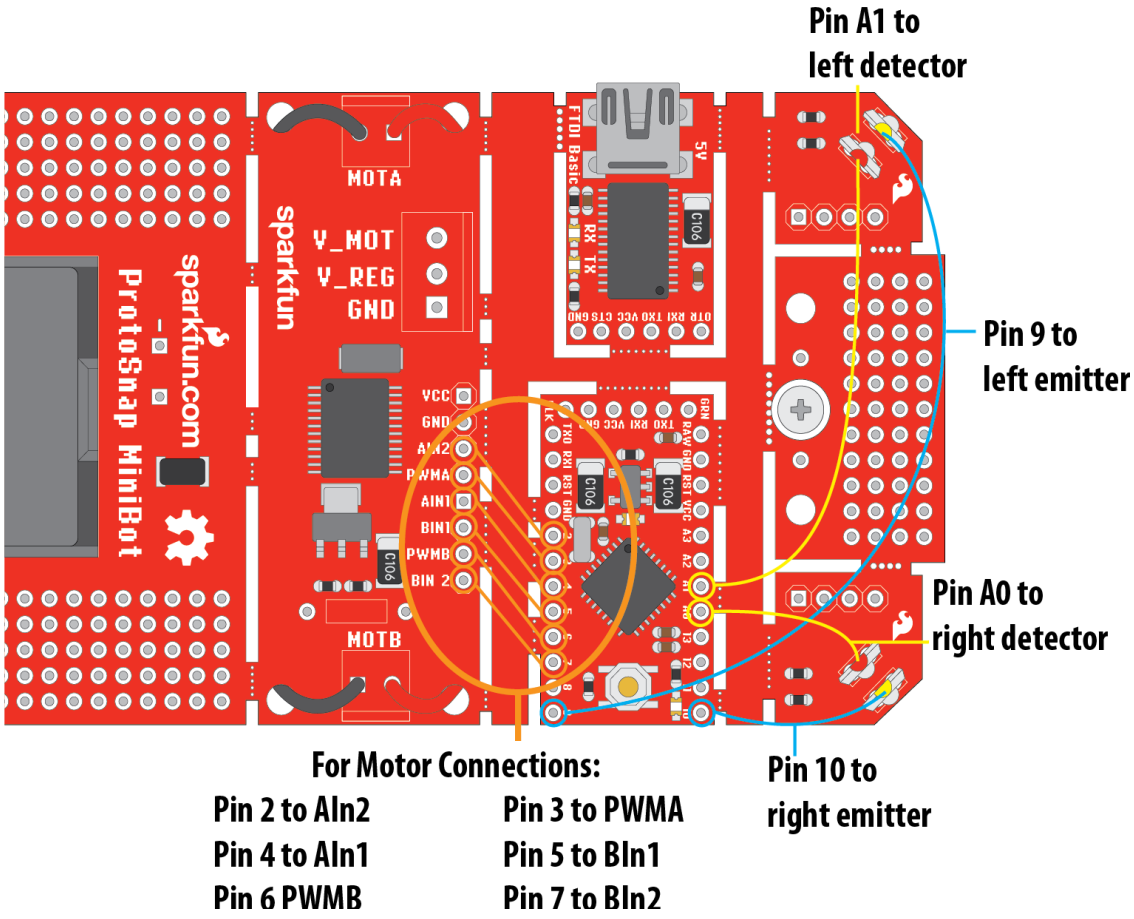


Dual Motor Gearbox

The final piece of the MiniBot you will be using is the actual brain of the robot! The ATMega chip that runs all Arduino MicroControllers. You will be loading code onto this chip to turn the “pins” that control the sensors and the motor driver. There are many different ways to put code onto the MiniBot, you are going to be creating code in a program called Minibloq. Below are descriptions of important parts of the code. Once you understand how the pieces of the code work you can put them together into your own order, creating a robot that does exactly what **you** want it to do.

SparkFun's ProtoSnap MiniBot's Pin connections

Below is a diagram indicating which pins on the Arduino Mini are connected to which sensors and motor driver pins. (If you are unfamiliar with the term “pins” these are the little holes in the electronics that are used to connect the pieces of the MiniBot together.)



The Motor Connections:
AIn1 and AIn2 are the two connections that control the direction of motor A, or the left motor.
BIn1 and Bin2 are the two connections that control the direction of motor B, or the right motor.
PWMA is the connection that controls the speed of motor A, or the left motor.
PWMB is the connection that controls the speed of motor B, or the right motor.

This leaves pins A3, A2, 13, 12, 11, and 8 open to attach to sensors not included with the MiniBot.

ProtoSnap MiniBot with Minibloq drag and drop programming

The Programming Environment

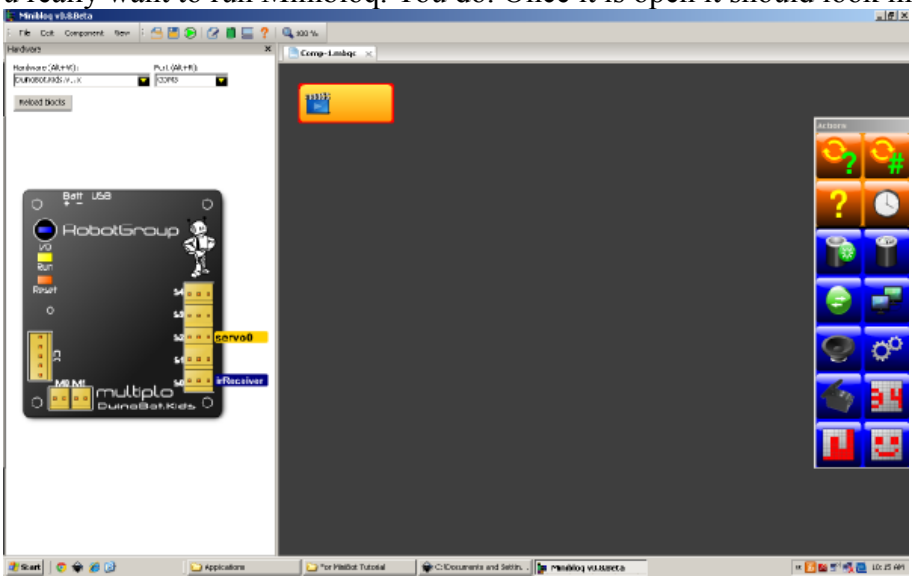
A programming environment is something that a person uses to create code and sometimes to upload it onto a MicroController. The programming environment you will be using today was created in Buenos Aires, Argentina by a group called Multiplo.

The first thing we have to do is make sure that we have the latest version of Minibloq installed on the computer. Check the name of the folder Minibloq is in, is it called Minibloq.v0.81? It is? Great, skip to the next page and get started programming. If your version of Minibloq is not version 0.81 then follow the instructions below to get the most recent version installed.

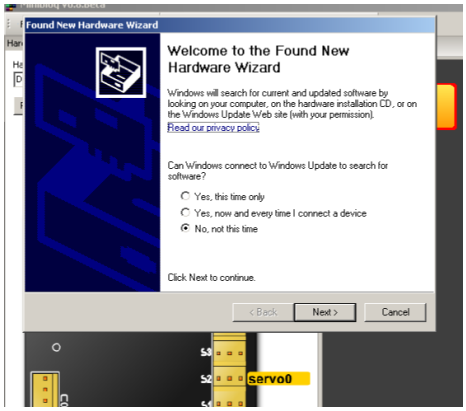
1. Go to blog.minibloq.org and find the most recent version of Minibloq to download.

2. Open the file you downloaded and put it in your program folder.

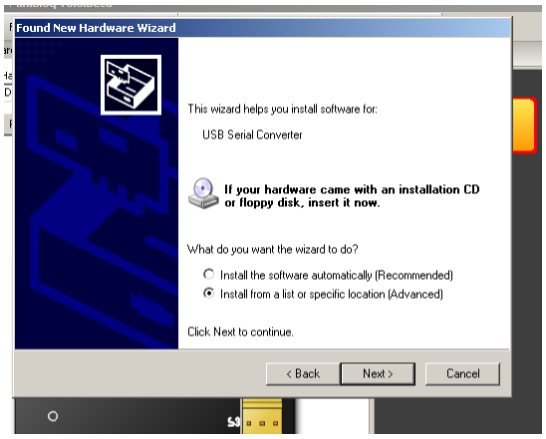
3. Double click on the file called MinibloqRun.exe to open Minibloq. You will get a pop up asking you if you really want to run Minibloq. You do. Once it is open it should look like this:



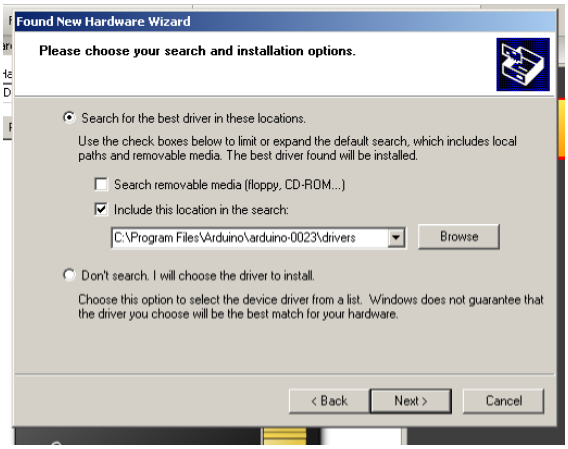
4. First let's plug one end of the USB cable into the MiniBot and the other end into the USB port on the computer. You should get a Hardware Wizard pop up menu that looks like this:



5. Continue through the Hardware Wizard installation process until the computer asks you if you would like it to find the USB serial converter software or install your own. The pop up window should look like the one below. Choose “Install from a specific location (advanced)”

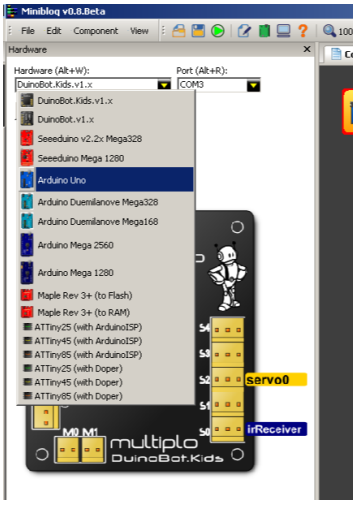


6. In the next pop up window find the folder that holds your Arduino driver files, it should be inside the Arduino folder labeled Drivers.



7. You may have to repeat this process of installing the drivers a second time.

8. Now we need to make sure you have selected the correct board in Minibloq, select the Arduino Uno from the drop down menu in the upper left corner.

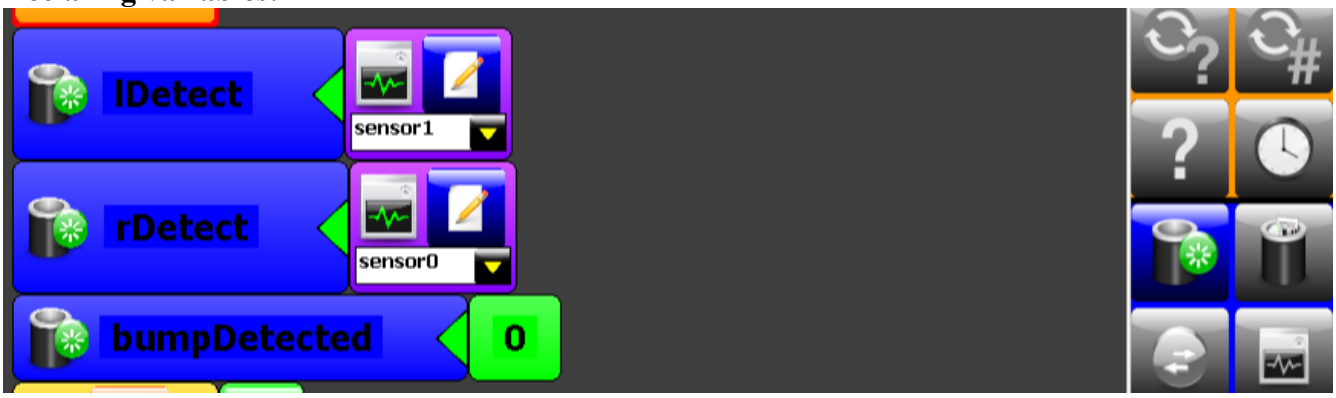


ProtoSnap MiniBot with Minibloq drag and drop programming

Programming the Robot!

Below are a couple different groups of code with each block explained. The blocks of code shown and described are the basic groups used in this first programming of the robot.

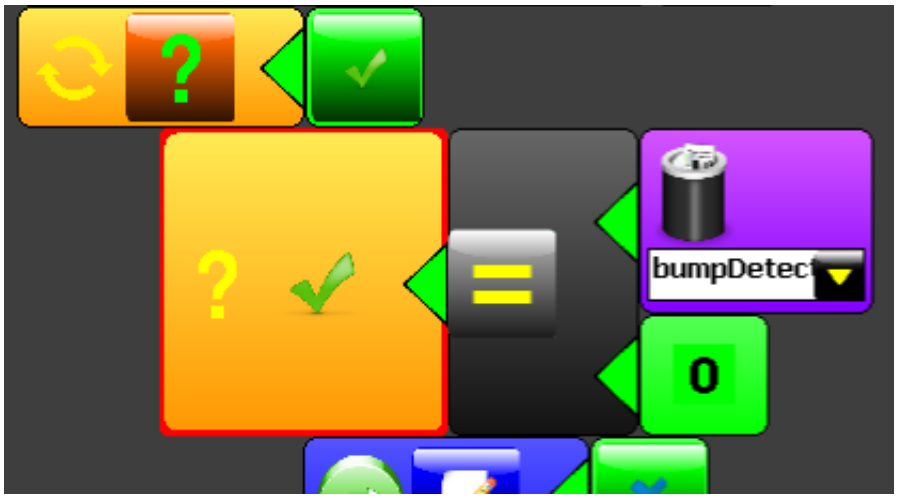
Declaring variables:



First you need to "declare" variables before you can use them. This is like telling the computer that there is something that can hold your information. Think of declaring a variable as getting a box to hold stuff, or getting a locker that you can later put your books or other information into. The difference is that most variables can only hold one single piece of information at a time.

In this case we declared three variables: lDetect to read the infrared sensor on the left side of the robot, rDetect to read the infrared sensor on the right side of the robot and bumpDetected so we can keep track of when either of those two sensors detects an obstacle.

A "While loop" and an "If statement":



The first block of code with the two arrows in a circle simply says, while there is a green check box to the right of this question mark keep doing all the code inside of this "while loop". See if you can find the block at the bottom of the code that closes this "while loop". (Hint, it will be orange.) Check with a teacher to see if you are right.

The larger block below the “while loop” is an “if statement”. That's a fancy way to say that the computer is asking a question. In this example the question is: Does the variable bumpDetected equal zero? In human speak this means, is there anything in the way of the robot's sensors? If this is true, and there is nothing in front of the sensors the robot will do the code inside this “if statement”. Like the while loop you can figure out what code is inside the “if statement” by finding the orange block that closes this “if statement”. (Hint, it's right after the question mark with a blue X next to it. If you're curious the question mark with the blue X is an “else statement”, the buddy of the “if statement”.)

Motor Driver code:

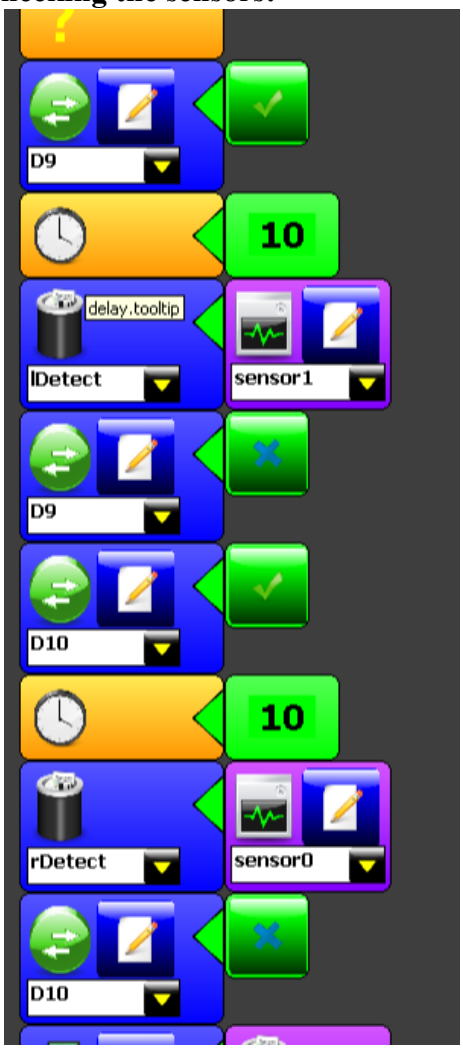
Ok, in the code above we figured out there is nothing in the way of the robot so it can move forward. Here is the code that makes that happen. The first two blocks tell one of the wheel which way to go. If you switch the blue X and the green check next to these two blocks of code that talk to D4 and D2 this wheel will spin the other direction. They two blocks of code below those two... labeled D5 and D7, you guessed it! They control the direction of the other wheel.

Below that the two blocks of code labeled PWM3 and PWM6 control the speed of the two wheels. Both wheels are set to 40% speed, but they could be set to any number between 0 and 100. What do you think would happen if you set them to different numbers?

Last of all, we need to tell the robot how long to move the wheels at the speeds we set! That's what the clock does, don't worry that ten miliseconds seems like a short amount of time to move forward, the point of the “while loop” is that this code will continue to happen over and over and over again.

Next up: Checking the sensors....

Checking the sensors:



After the robot moves forward it needs to check it's sensors and here's that code. It's a little more complicated than just checking the sensors though, remember that our infrared sensors are actually two part sensors. The other thing to remember is that because they both use infrared we need to check them one at time so the infrared from one doesn't show up on the other.

First what the robot does is turn on (green check mark) the infrared light connected to D9.

Then the robot needs to wait a little so there is infrared to measure. The block of code with the clock on it tells the robot to wait 10 miliseconds. If the robot didn't wait at least a couple miliseconds the infrared wouldn't have time to exit the sensor before the robot took a measurement.

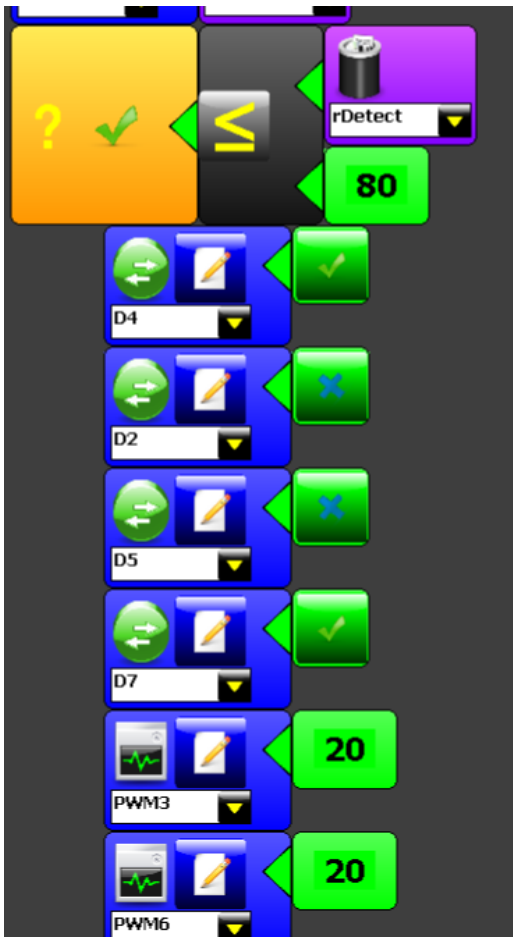
After that we can set the variable “lDetect” equal to the reading we get off of the analog sensor number one. This how we get the actual value off of the sensor.

Finally we have to turn the infrared light attached to D9 off so when we do it all over again with the other sensor there isn't extra infrared floating around.

With the block of code labeled D10 we do the whole thing over again, except with the other sensor!

And to finish: Turning away from an obstacle....

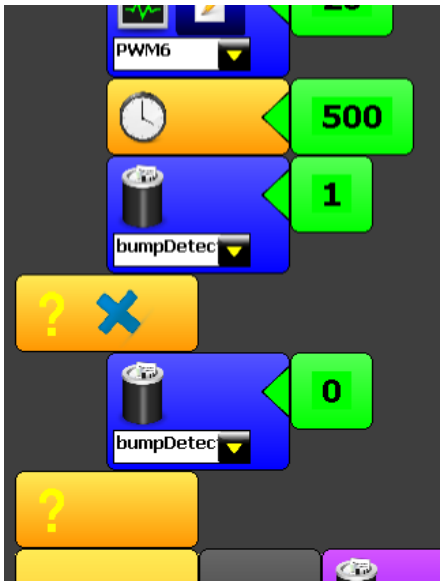
Turning away from an obstacle:



Finally we have a block of code that is similar to two of the blocks you dealt with before. These blocks of code use an "if statement" to see if something has come into the path of the right infrared sensor. The robot knows this because the value for the variable rDetect goes below 80.

The code below that "if statement" should look very familiar. It's the same blocks of code the robot uses to go forward, except both wheels are not turning in the same direction. They are turning in different directions so the robot turns away from the obstacle.

In order to turn this code into the same section that deals with the left sensor and turns the other way you would need to change a total of five blocks of code. What would they be?



And to wrap it all up we need to tell the robot one or two last things.

How long should the robot turn around for? In this case the code tells the robot to turn for one half a second, or 500 miliseconds. After that the robot sets the variable "bumpDetected" to 1 or true so it knows it shouldn't move forward like usual.

And then this last little block of code happens when the "if statement" above doesn't happen. Basically if the "if statement" does not see something in front of the robot the variable "bumpDetect" is set to zero. When "bumpDetect" is set to zero the robot knows it can move forward without turning before checking the sensors again.

The code explained above is not all of the code in the example, but it is most of it. This is a very simple piece of code to get a robot moving, but you can easily make it better! What would be your first move to make this code better? Try talking about it and writing out your code in English (or pictures) before trying to create it in Minibloq. If you have questions, as always the help menu is your best first move, that way you don't have to rely on a teacher to be there to answer your questions, you can try to answer them yourself. Good luck and have fun with your robot!