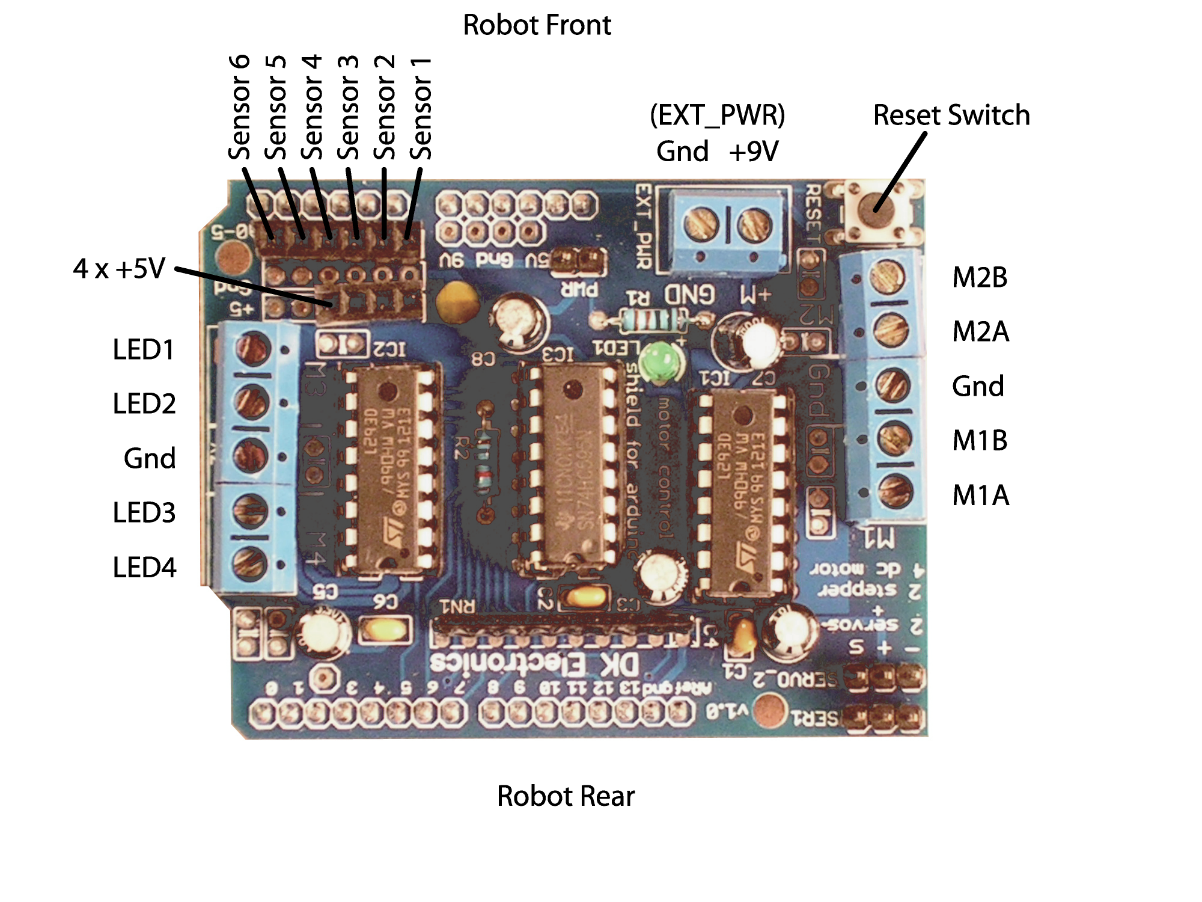
Motor Controller Overview



Breadboard Connections

|  |  |
| --- | --- |
|  | breadboard.png |
|  |  |

How to Install the Arduino program

On Your Computer

1. Go to the Arduino website: www.arduino.cc

2. Click on “Download”

3. Scroll down and under “Arduino IDE,” click on your operating system. The installer will download.

4. Open the file and click through to start the install.

How to Load a Program

Onto the Robot

1. Open the Arduino program

2. Go to Tools->Board and make sure “Arduino Uno” is selected.

3. Download the ChickTech Arduino zipfile from the USB drive onto your computer (Robotics Arduino Files.zip) and extract it.

4. In Arduino, go to File->Open and open the CTRobotSimple.ino file (in Robotics Arduino Files->CTRobotSimple). This will open a new window with the code in this file.

5. Go to Sketch->Import Library->Add Library. Navigate to Robotics Arduino Files->libraries and click on the TwoMotorGearbox folder. Then click open.

6. Click on the check mark “Verify” button in the top left of the Arduino window: verify.png

Check to make sure it says “Done compiling” at the bottom of the window.

7. Attach the programming cable to the Arduino board on the robot and to your computer’s USB input.

8. Click the arrow “Upload” button upload.png next to the Verify button. This will load the program onto the robot.

9. If there are errors, go to Tools->Serial Port and choose a different serial port.

Programming the Robot to Move

Open “CTRobotSimple” and look for the following section of code:

// time in milliseconds, direction M1, speed M1, direction M2, speed M2

CCTTwoMotorControl commandList[] = {

CCTTwoMotorControl(100),

CCTTwoMotorControl(500, 'F', 250, 'F', 250), // move forward .5 seconds

CCTTwoMotorControl(100),

CCTTwoMotorControl(500, 'R', 250, 'R', 250), // move backward .5 seconds

CCTTwoMotorControl(100),

CCTTwoMotorControl(500, 'F', 250, 'R', 250), // turn left .5 seconds

CCTTwoMotorControl(100),

CCTTwoMotorControl(500, 'R', 250, 'F', 250), // turn right .5 seconds

CCTTwoMotorControl(100),

CCTTwoMotorControl(2000, 'F', 250, 'F', 250),

CCTTwoMotorControl(100),

CCTTwoMotorControl(2000, 'R', 250, 'R', 250),

CCTTwoMotorControl(100),

CCTTwoMotorControl(4000, 'F', 250, 'F', 100),

CCTTwoMotorControl(100)

};

Each line that begins with “CCTTwoMotorControl” makes the robot move in a different way.

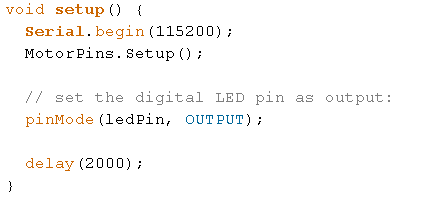
* The line “CCTTwoMotorControl(100)” tells the robot to do nothing for 100 milliseconds (0.1 seconds).
* The line “CCTTwoMotorControl(500, 'F', 250, 'F', 250)” tells the robot to move forward for 500 milliseconds (0.5 seconds).
* The first parameter “500” is the time in milliseconds that the robot will move. The second parameter “F” tells the left wheel to spin forward. The third parameter “250” is the amount of power to apply to the left wheel. 255 is maximum power. The fourth and fifth parameters is the direction and power to apply to the right wheel.
* The valid values for direction are “F” forward, “R” reverse, “S” stop, “B” brake. The effect of stop and brake are the same.
* By adding and changing these lines code you can make you robot in any manner you wish!

Code Breakdown

In Arduino Code, there are two functions that determine what your robot does. The first is the “setup()” function, and the second is the “loop()” function. All of the code in the setup function is executed once when the robot is powered on, or when it is reset.The function is used to configure the robot so that it is ready to run the commands that you specify in the command list.

The “loop()” function is run over and over again, starting right after the setup function completes until the end of time (as long as the robot has power). That’s why it’s called the loop function; it loops continuously. It’s in this function that all of the commands in the command list are read and executed.

So what is a function? Basically, a function is a command to do something. A function appears in two different forms: the function definition, and the function call. Below is the function definition for the “setup” function:



A function definition describes how a function works. In order to describe how it works, there are four things to consider:

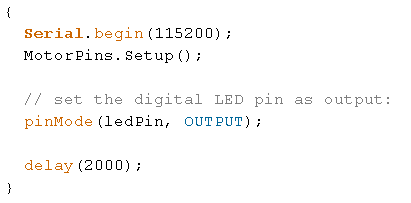
1. What is the name of the function (function name)?
2. What additional information do I need to tell the function so that it can run (input parameters)?
3. What does the function do when called (function body)?
4. After the function is done, what results does it provide (return value)?

In all function definitions, the function name, input parameters and return value (1,2 and 4) are all on the first line. Below is the first line of the setup function:



* “void” is the return value. A return value of “void” means that the function doesn’t provide any results because no results are needed. A result can take on many different forms, like a number, a message, etc.
* “setup” is the name of the function. The name is important so that it can be called later.
* Input parameters appear in between the parenthesis “()”. In this case, the setup function does not require any additional information to run, so it doesn’t take any information.

Finally, everything that appears between the starting curly brace “{“ and the ending curly brace “}” is the function body, or what the function does. Here is the function body of the setup function:



Every single line in the body of this function definition are calls of different functions. In other words, the setup function is defined as the usage of these four commands in that particular order. Functions are layered and combined in this way to make bigger programs, like those that control robots.

How to add LEDs (and blink)

If you’d like to add an LED, add this line to the **loop()** section of the code and connect an LED to the inner M3 pin:

varyLED.motor(3, 'F', 100);

Repeat, changing “3” to “4” for M4.

---------------------------------------------------------------------

For adding a blinking LED:

Connect an LED to the inner M3 pin.

Add this code before the “void setup()” line:

int ledState = 0;

unsigned long ledPreviousMillis = 0;

unsigned long ledInterval = 500;

The “ledInterval” tells it how fast to blink the LED, in milliseconds. So, decreasing ledInterval will make the LED blink faster.

Add this code inside “loop()”:

if(currentMillis - ledPreviousMillis > ledInterval)

{

// save the last time you blinked the LED

ledPreviousMillis = currentMillis;

if (ledState == 0)

ledState = 100;

else

ledState = 0;

varyLED.motor(3, 'F', ledState);

}