Inexpensive Ardunio-based robot platform

http://www.cla.purdue.edu/vpa/etb/
Fabian Winkler

Required parts/supplies for this workshop:

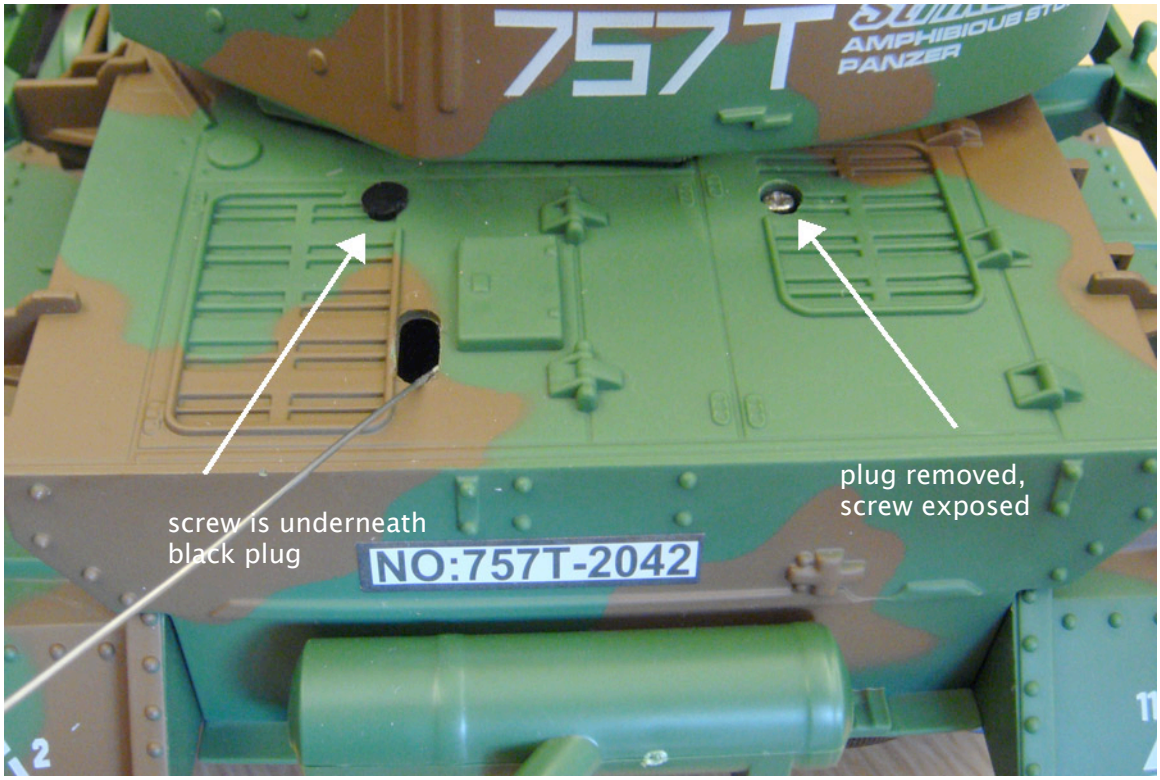
Description	**Get it from**
RC toy tank	hobbytron.com ZX-757T-2042
NiMH Battery Pack: 7.2V 2200 mAh (6xAA, 2Rx3) Not necessary but better than supplied batteries in toy tank.	batteryspace.com HAA6R1WR20
Multi-Current Smart Charger (1-2 A) For any 4.8V - 10.8V NiMH / NiCd Battery Packs - UL Listed Not necessary but better than supplied charger of toy tank.	batteryspace.com CH-UN0409ARC
wires (solid core)	Jameco.com PN 126360
SN754410 motor driver IC	Jameco.com PN1054684
Arduino board	Sparkfun.com DEV-00666
Microswitch, straight lever	Jameco.com PN2117333

Build an Arduino-based robot for less than $75,- including all parts and Arduino board by turning Hobbytron's Amphibious RC Panzer tank (or any other inexpensive R/C tank) into an autonomous caterpillar-track robot platform.

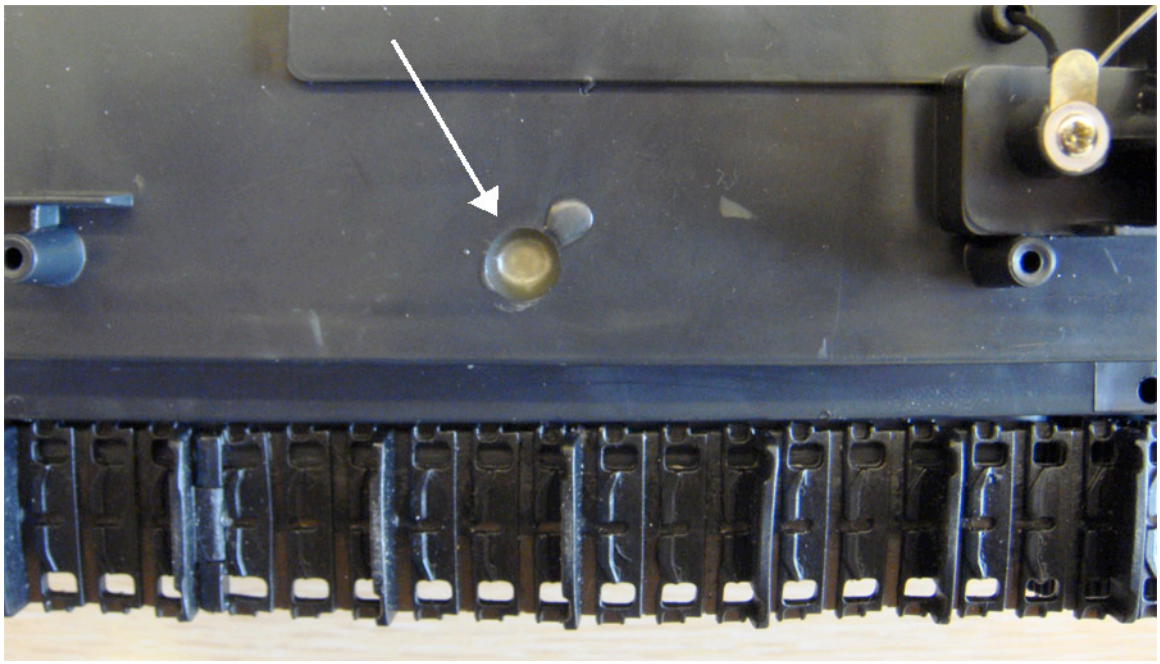
Purchase the Amphibious RC Panzer tank (USD 29.97 with free shipping at hobbytron.com as of March 5, 2010)

This set includes a 700mAh 6V rechargeable NiMH battery pack and charger (even though a better and much faster charger replacement is: Multi-Current Smart Charger (1-2 A) For any 4.8V - 10.8V NiMH / NiCd Battery Packs - UL Listed from http://www.batteryspace.com for USD 26.95)

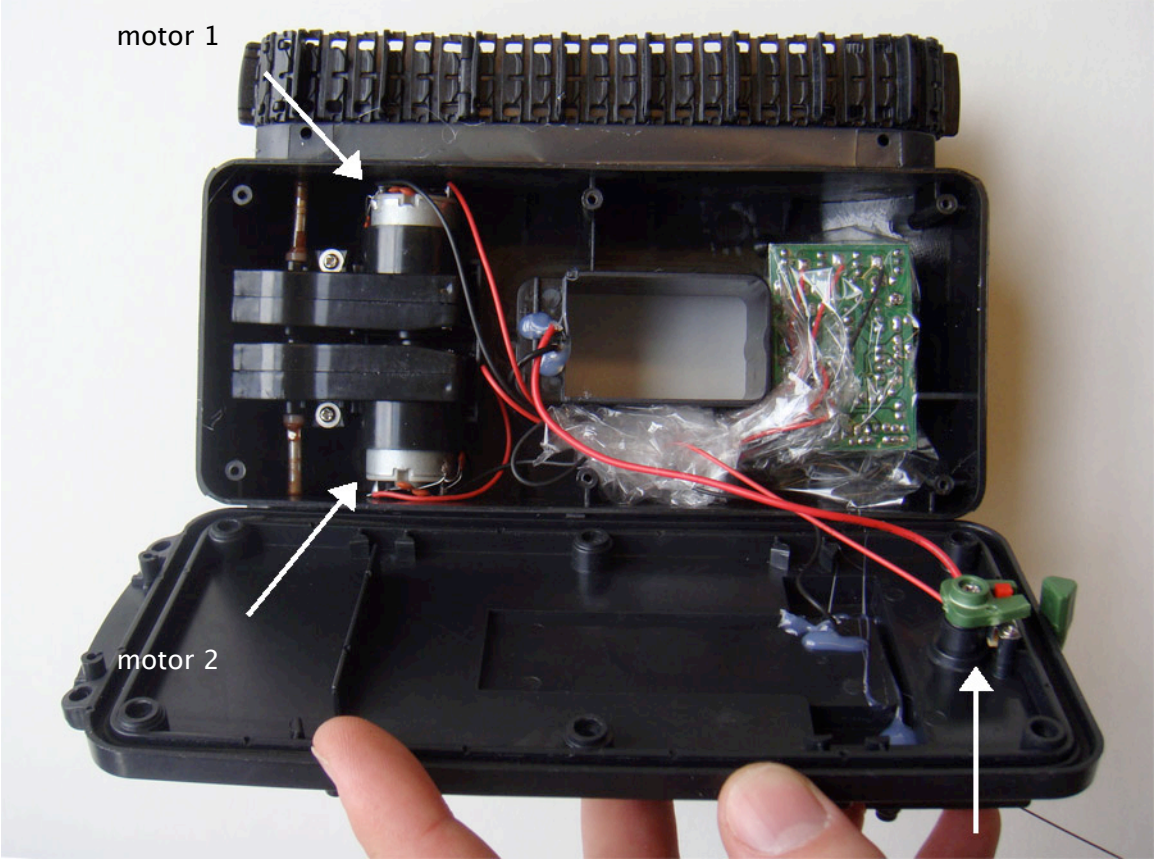
First, take off the top of the tank, there are two screws on the bottom side of the tank and two on top, hidden under waterproof plastic plugs.



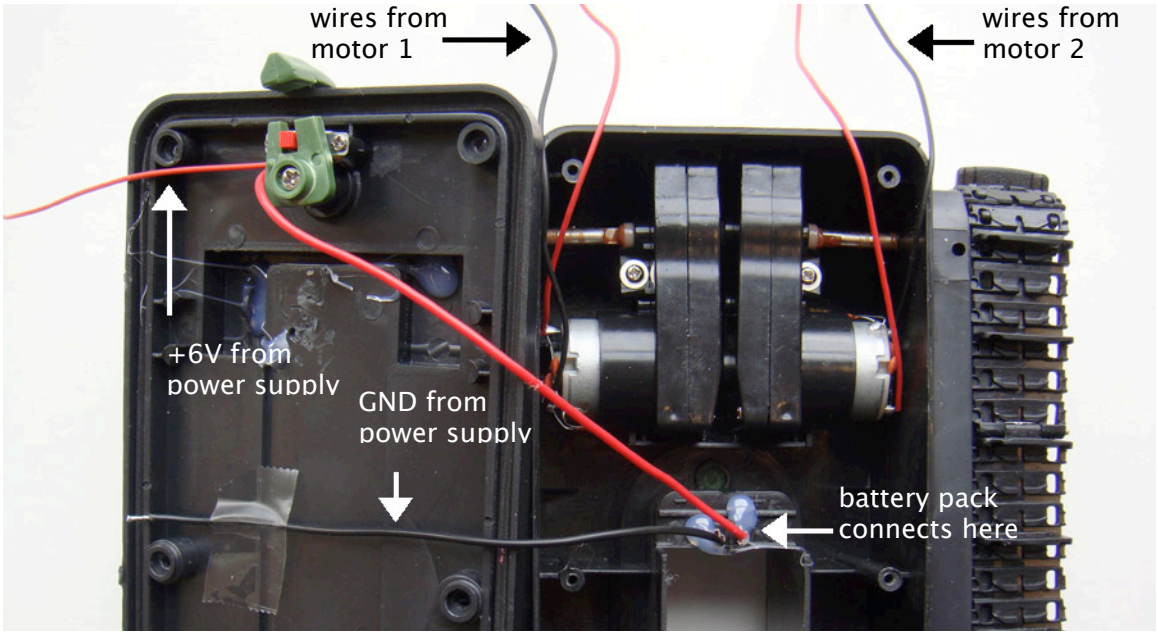
After the top is off, unscrew a total of six screws on top of the tank sealed with wax to get to the internal circuitry:



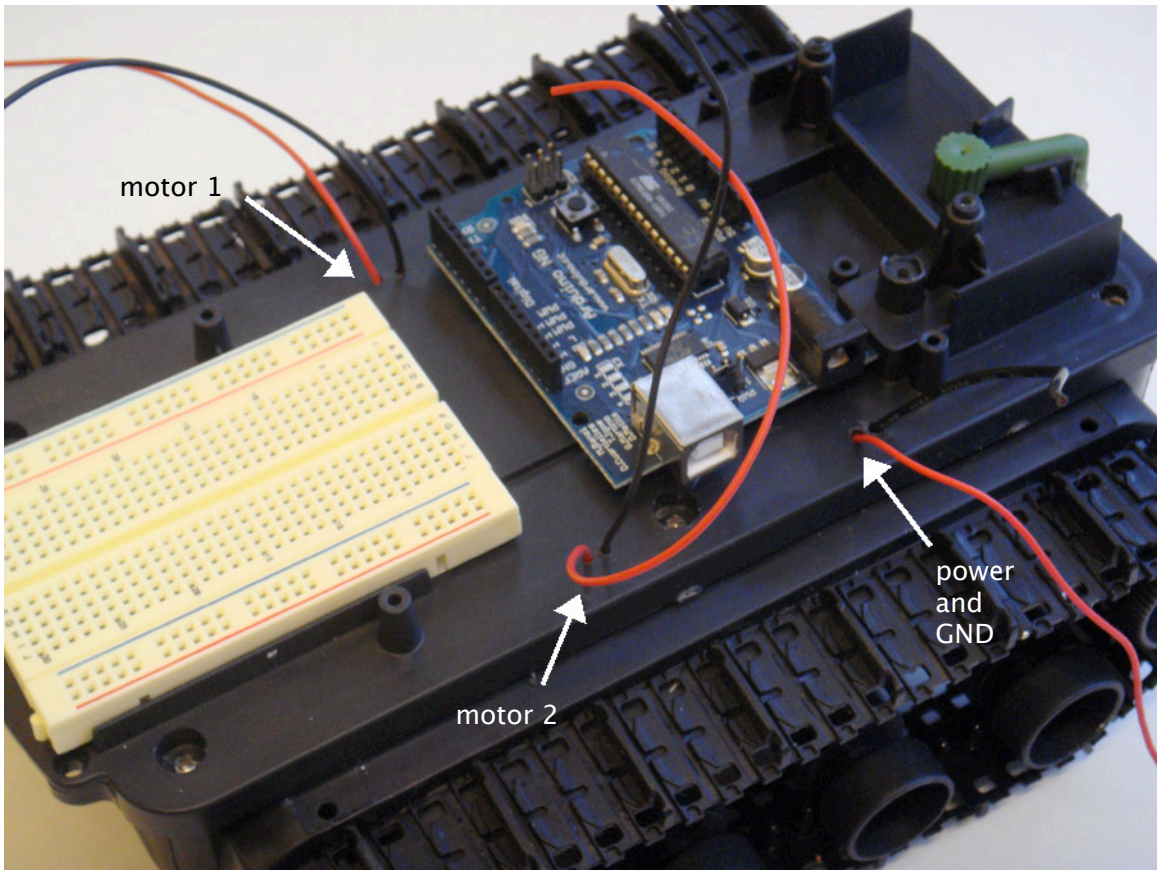
After you unscrew the screws, the black top part comes off easily and gives access to the motor control and RC circuitry on the inside of the tank.



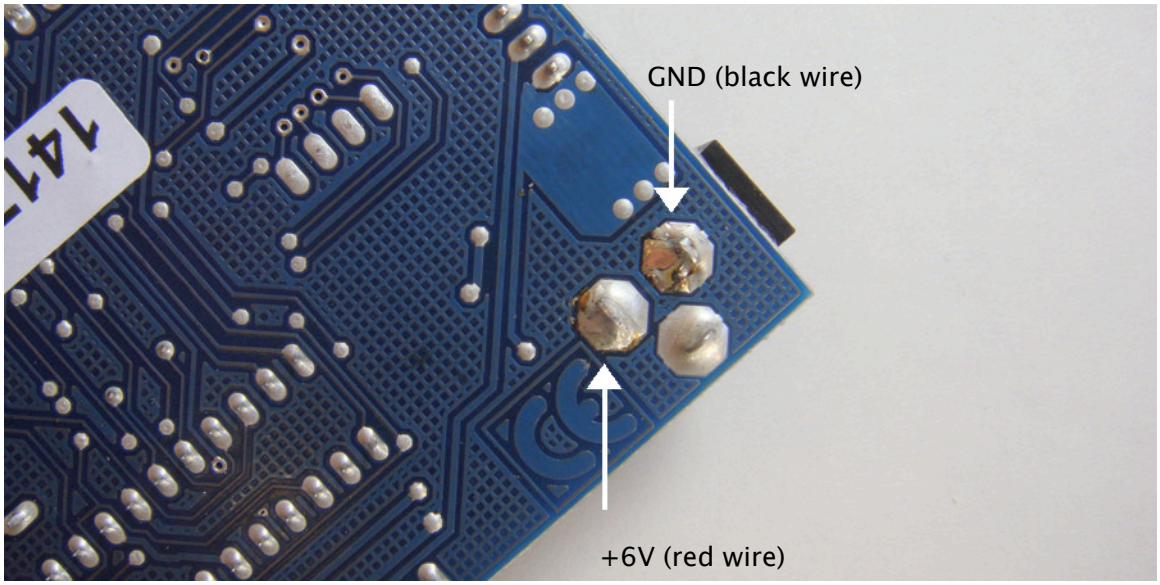
Take the circuit out of the plastic bag and separate the wires for the motors and the power supply, like this:



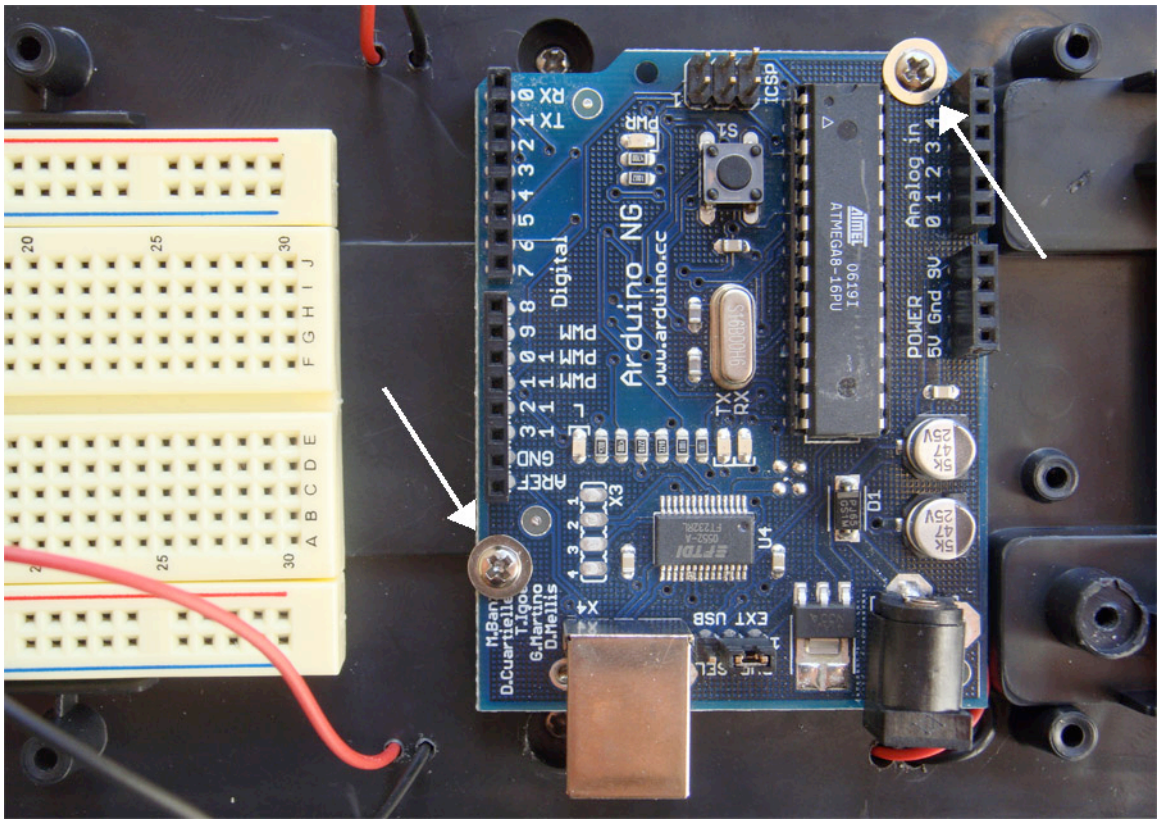
Then drill holes into the top cover – two for the GND and +6V wires, two for motor 1 and two for motor 2 – lay out your Arduino board and breadboard like this to determine the position of the holes:



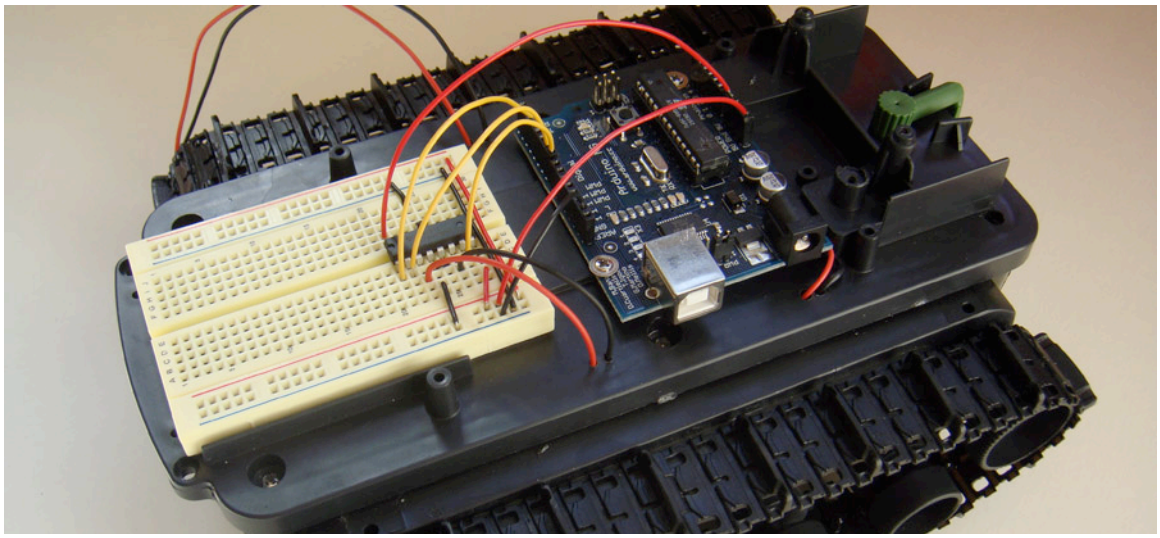
Now solder the power (red) and ground (GND, black) wire from the robot's tank's power supply (battery pack) to the bottom of the Arduino board's power plug, like this:



If you have an older version of the Arduino board (pre-Duemillanove), don't forget to set the jumper for the power supply on top of the board to "EXT" (instead of "USB"). Now use some of the extra screws from unscrewing the tank's top and secure the Arduino board with them. Use the sticky tape on the bottom of the breadboard to secure it to the front of the tank platform.



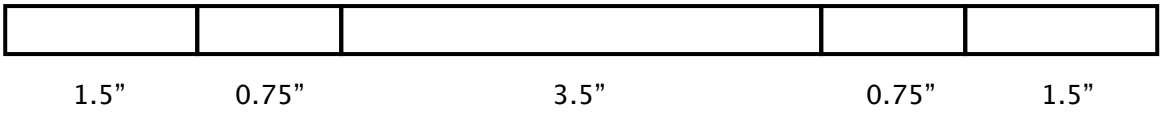
You are now ready to build the motor driver circuit based on the SN754410 on the breadboard and connect one of the tank's motors to it as outlined in workshop 01 (DC motor control). Your robot will look something like this:



Test out the Arduino code from workshop 01 and make sure to test out different motor speeds as well as reversion the direction of the motor using code.

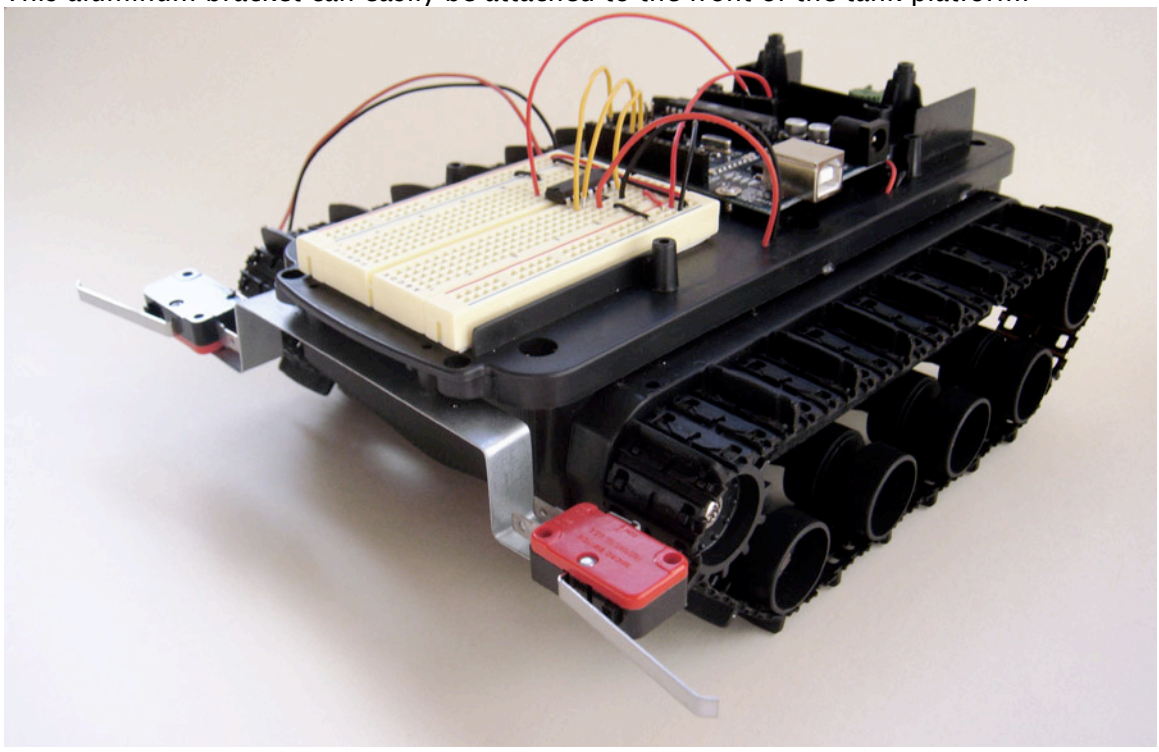
Can you hook up the second motor by looking at the SN754410 pinout diagram from workshop 01 and rewriting the Arduino code? The goal is to make the tank move forward on a straight line.

We can now work on the attachment of the bump sensors which are simple on/off micro switches. I used a piece of aluminum, roughly 8 inches long and ½ inch wide. I bend it as follows:



1.5” 0.75” 3.5” 0.75” 1.5”

This aluminum bracket can easily be attached to the front of the tank platform:

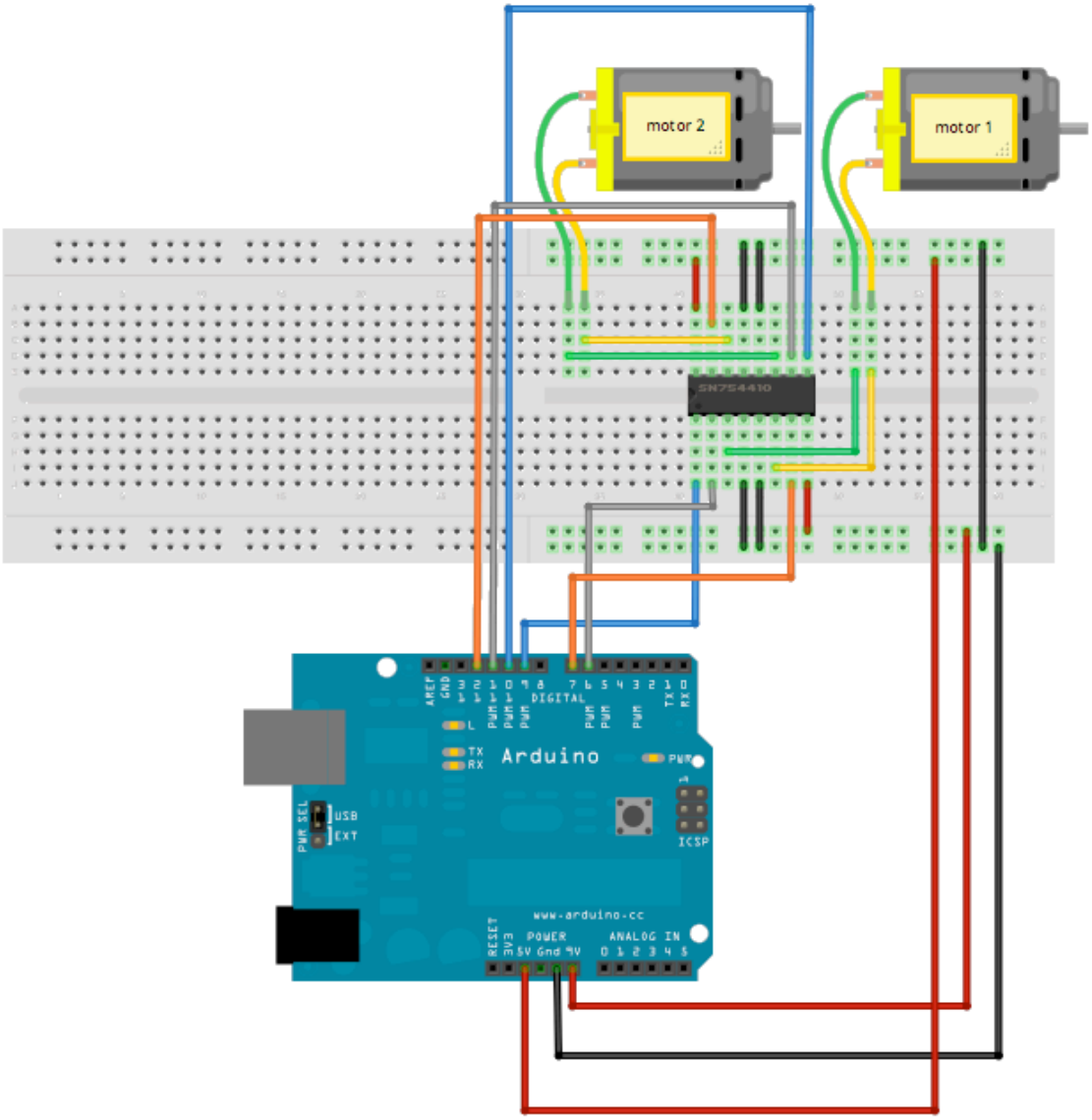


Just use the two small pre-drilled holes in the black plastic top and two screws left over from unscrewing the plastic decorative top of the tank earlier…

You might want to experiment with the placement of the micro switches in the front of the tank platform, the closer they are mounted to the middle of the platform the less the probability of not being triggered by objects (the picture on the previous page was my starting point and I had to mount them closer to the middle because often they were not triggered when running into an obstacle). Also experiment with the mounting height.

Before we continue, make sure that both motors work as expected by wiring them to the SN754410 motor driver IC and by writing the appropriate code. Please note that my circuit drawing and code are backwards compatible with older Arduinos (I am using the Arduino NG with the ATmega8), which have fewer dedicated PWM pins.

Here is a Fritzing circuit drawing that shows how both motors can be connected:

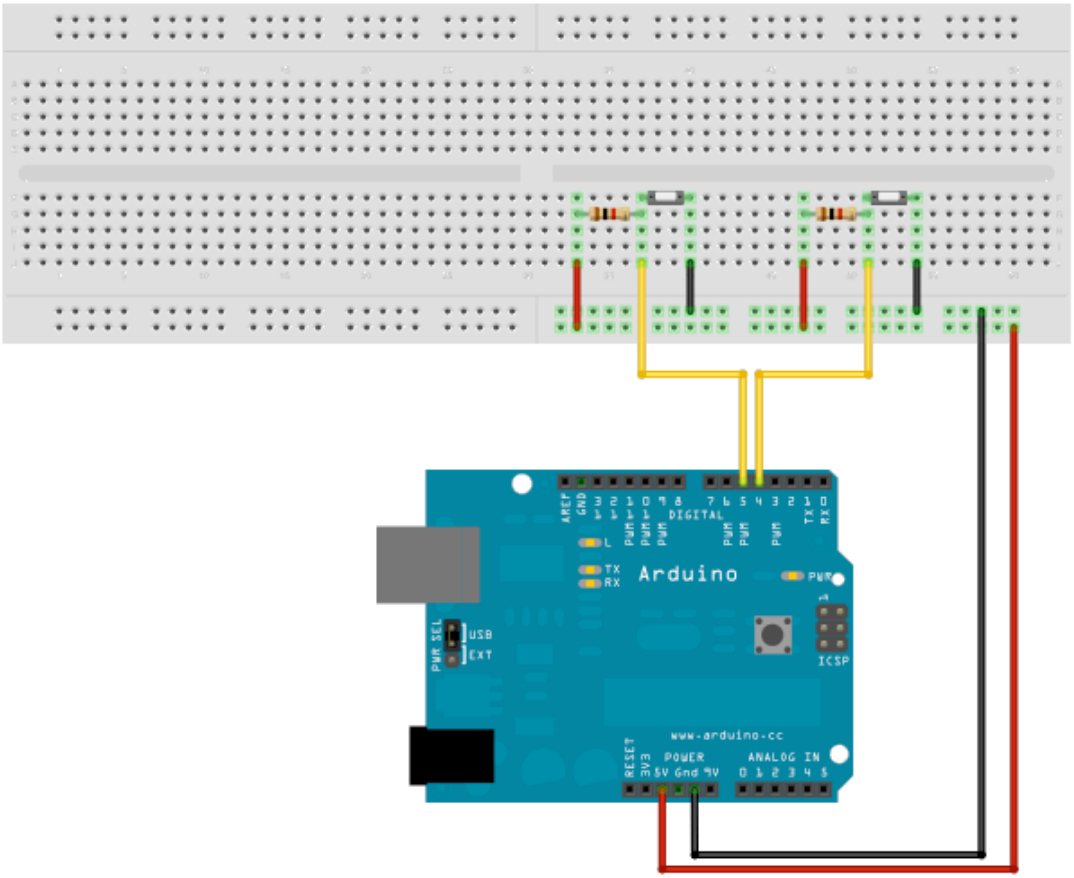


And here is the Arduinio program that goes with it – just to make the robot go forward in a straight line. Try it out, you might have to reverse the turning of the motors in the code to make it work properly:

```
/*
 * Arduino code for SN754410 H-bridge
 * motor driver control.
 * copyleft Feb. 2010, Fabian Winkler
 *
 */

int speedPin1 = 9;      // H-bridge enable pin for speed control motor1
int motor1APin = 6;     // H-bridge leg 1
int motor2APin = 7;     // H-bridge leg 2
int speedPin2 = 10;     // H-bridge enable pin for speed control motor2
int motor3APin = 11;    // H-bridge leg 3
int motor4APin = 12;    // H-bridge leg 2

int ledPin = 13;        // status LED
int speed_value_motor1; // value for motor1 speed
int speed_value_motor2; // value for motor2 speed


void setup() {

  // set digital i/o pins as outputs:
  pinMode(speedPin1, OUTPUT);
  pinMode(speedPin2, OUTPUT);
  pinMode(motor1APin, OUTPUT);
  pinMode(motor2APin, OUTPUT);
  pinMode(motor3APin, OUTPUT);
  pinMode(motor4APin, OUTPUT);
  pinMode(ledPin, OUTPUT);
}
void loop() {
  digitalWrite(ledPin, HIGH);  // status LED is always on

     // put motor1 in forward motion
    digitalWrite(motor1APin, LOW);   // set leg 1 of the H-bridge low
    digitalWrite(motor2APin, HIGH);  // set leg 2 of the H-bridge high

    // just invert the above values for reverse motion,
    // i.e. motor1APin = HIGH and motor2APin = LOW

    // put motor2 in forward motion
    digitalWrite(motor3APin, LOW);   // set leg 3 of the H-bridge low
    digitalWrite(motor4APin, HIGH);  // set leg 4 of the H-bridge high


    // control the speed 0- 255
    speed_value_motor1 = 200;
    speed_value_motor2 = 200;
    analogWrite(speedPin1, speed_value_motor1);  // output speed as
                                                 // PWM value
    analogWrite(speedPin2, speed_value_motor2);  // output speed as
                                                 // PWM value
}
```

Connecting the bumper sensors

After both motors can be controlled through the Arduino, let's continue with the bumper switches. They need to be connected to the Arduino in a circuit using a "pull-up" resistor, pulling the input pin high (i.e. to +5V) whenever the switch is not triggered and pulling it low (i.e. to ground) when the switch is triggered – this avoid the pin to "float" (i.e. being susceptible to weak electronic charges in its surrounding environment resulting in instable sensor output whenever the switch is not triggered. Here is a circuit drawing just of the switches, the resistor values are 1kΩ:



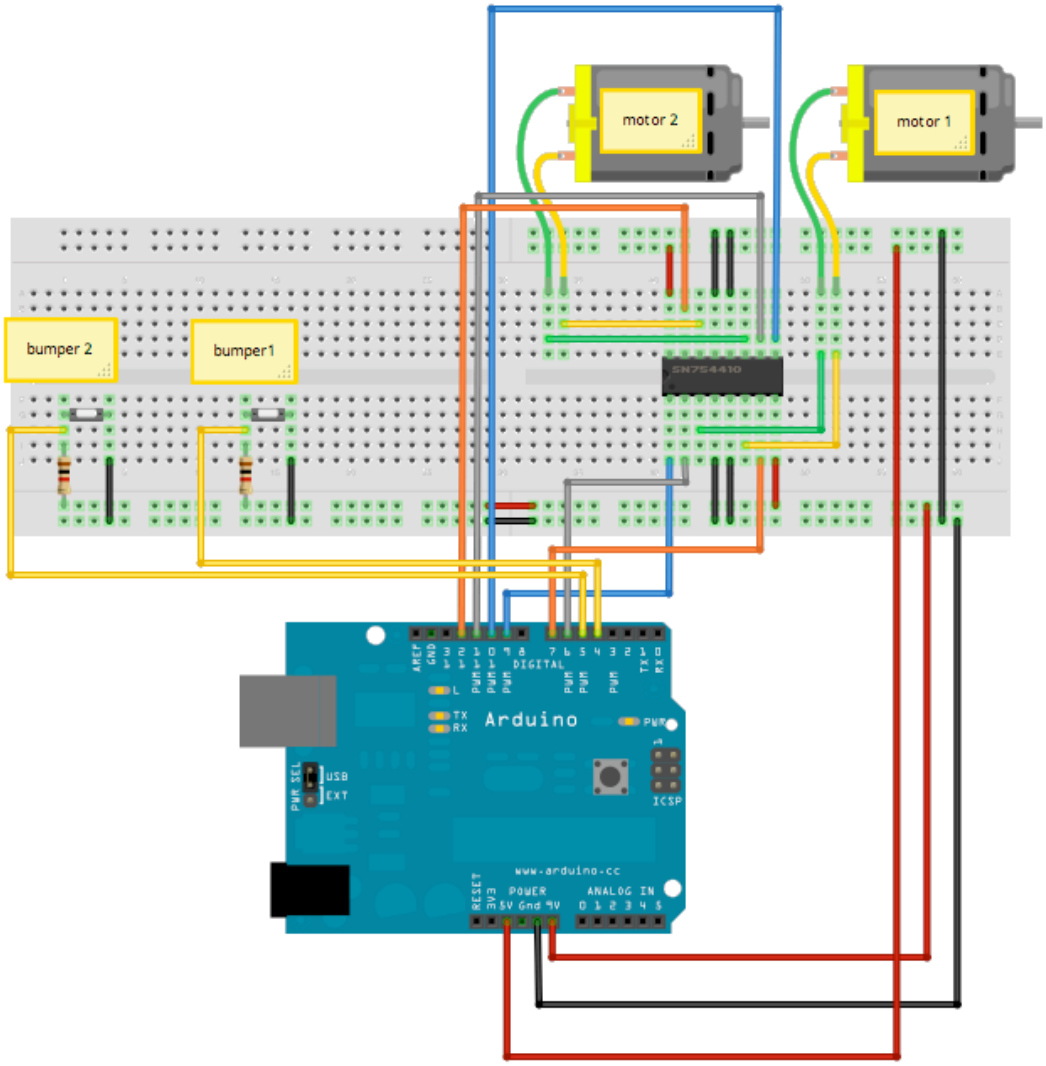
This is a code example that goes along with this circuit. It is only for testing the switches to make sure that they work. It uses the LED on pin 13 of the Arduino board to show when one of the switches has been triggered (it will light up).

```
int bumper_left = 4;          // switch 1
int bumper_right = 5;         // switch 2
int ledPin =  13;             // the number of the LED pin

// variables for reading the switches:
int bumper_left_state = 0;    // variable for reading the status of
                              // switch 1
int bumper_right_state = 0;   // variable for reading the status of
                              // switch 2

void setup() {
  // initialize the LED pin as an output:
  pinMode(ledPin, OUTPUT);
```

```
  // initialize the switch pins as an inputs:
  pinMode(bumper_left, INPUT);
  pinMode(bumper_right, INPUT);

}

void loop(){
  // read the state of switch 1 value:
  bumper_left_state  = digitalRead(bumper_left);
  bumper_right_state  = digitalRead(bumper_right);


// check if switch 1 is pressed.
// if it is, the bumper_left_state is LOW since we use pull-up
// resistors:
  if (bumper_left_state == LOW) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }

// check if switch 2 is pressed.
// if it is, the bumper_right_state is LOW since we use pull-up
// resistors:
  if (bumper_right_state == LOW) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }

}
```

Putting everything together

We are ready to connect the circuits together and also combine the Arduino code examples. On the next page is a circuit drawing and the complete code. The goal of the robot is to wander around in a room autonomously. When it bumps into something with its left bumper, it will go backward and turn left then forward again until it bumps into the next obstacle. It will behave accordingly for bumping into objects with its right sensor. The program also includes a little subroutine for exiting situations in which it gets stuck in tight corners. If the left and right bumpers are triggered consecutively (back and forth) for more than three times in a row, the robot makes a longer turn (approx. 180 degrees) to free itself.



Here is the code (it uses functions to make the individual movement stages clearer):

```
int ledPin = 13;                  // LED connected to digital pin 13
int left_bumper = 4;
int right_bumper = 5;
int left_val = 0;
int right_val = 0;
int right_motor_1a = 6; // corresponds to pin name on SN754410
int right_motor_2a = 7; // corresponds to pin name on SN754410
int right_motor_speed = 9;
int left_motor_3a = 11; // corresponds to pin name on SN754410
int left_motor_4a = 12; // corresponds to pin name on SN754410
int left_motor_speed = 10;

int stuck_left = 0; // if this value in combination with stuck right
exceeds a threshold the robot knows it's stuck and it will respond
correspondingly
int stuck_right = 0;
```

```
void setup()                    // run once, when the sketch starts
{
  pinMode(ledPin, OUTPUT);      // sets the digital pin as output
  pinMode(left_bumper, INPUT);
  pinMode(right_bumper, INPUT);

  pinMode(right_motor_1a, OUTPUT);
  pinMode(right_motor_2a, OUTPUT);
  pinMode(right_motor_speed, OUTPUT);

  pinMode(left_motor_3a, OUTPUT);
  pinMode(left_motor_4a, OUTPUT);
  pinMode(left_motor_speed, OUTPUT);

  Serial.begin(9600);
}

void loop()
{


left_val = digitalRead(left_bumper);
right_val = digitalRead(right_bumper);

 if (right_val == LOW) {
   if (stuck_left > 1 && stuck_right > 1) { // robot is stuck
                                            //turn around 180 degrees:
   go_backward();
   delay(400);
   turn_back_right();
   delay(1800);
   stuck_left = 0;
   stuck_right = 0;
   }
   else {
  // bump your way out
  go_backward();
  delay(400);
  turn_back_right();
  delay(900);
  }
 }


 if (left_val == LOW) {
   if (stuck_left > 1 && stuck_right > 1) { // robot is stuck
                                            //turn around 180 degrees:
   go_backward();
   delay(400);
   turn_back_left();
   delay(1800);
   stuck_left = 0;
   stuck_right = 0;
   }
   else {
 go_backward();
 delay(400);
 turn_back_left();
```

```
 delay(900);
 }
 }

  go_forward();
}



void go_forward() {
  // left motor:
   digitalWrite(left_motor_3a, HIGH); // this is forward
   digitalWrite(left_motor_4a, LOW);  // this is forward
   analogWrite(left_motor_speed, 255); // 0-255

  // right motor:
    digitalWrite(right_motor_1a, HIGH); // this is forward
    digitalWrite(right_motor_2a, LOW);  // this is forward
    analogWrite(right_motor_speed, 255); // 0-255
}

void go_backward() {
    // left motor:
   digitalWrite(left_motor_3a, LOW);  // this is backward
   digitalWrite(left_motor_4a, HIGH); // this is backward
   analogWrite(left_motor_speed, 255); // 0-255

  // right motor:
    digitalWrite(right_motor_1a, LOW); // this is backward
    digitalWrite(right_motor_2a, HIGH); // this is backward
    analogWrite(right_motor_speed, 255); // 0-255
}

void turn_back_left() {
      // left motor:
   digitalWrite(left_motor_3a, LOW); // this is backward
   digitalWrite(left_motor_4a, HIGH); // this is backward
   analogWrite(left_motor_speed, 0); // 0-255

  // right motor:
    digitalWrite(right_motor_1a, LOW); // this is backward
    digitalWrite(right_motor_2a, HIGH); // this is backward
    analogWrite(right_motor_speed, 255); // 0-255

    stuck_left = stuck_left+1;
    if (stuck_left > stuck_right+1) {
      stuck_left = 0;
    }
}


void turn_back_right() {
    // left motor:
   digitalWrite(left_motor_3a, LOW); // this is backward
   digitalWrite(left_motor_4a, HIGH); // this is backward
   analogWrite(left_motor_speed, 255); // 0-255

  // right motor:
```

```
    digitalWrite(right_motor_1a, LOW); // this is backward
    digitalWrite(right_motor_2a, HIGH); // this is backward
    analogWrite(right_motor_speed, 0); // 0-255

    stuck_right = stuck_right+1;
    if (stuck_right > stuck_left+1) {
      stuck_right = 0;
    }
}

void stop() {
  analogWrite(left_motor_speed, 0); // 0-255
  analogWrite(right_motor_speed, 0); // 0-255
}
```

The next step would be to use this most basic functioning robot platform to experiment with Behavior-Based-Robotics (BBR), subsumption architecture, bottom-up versus top-down approaches and add different sensors to the robot to change its behavior. This is material for future workshops…

A good starting point to read about Behavior-Based-Robotics is Paul Reiners' online article "Robots, mazes, and subsumption architecture" at:
http://www.ibm.com/developerworks/java/library/j-robots/