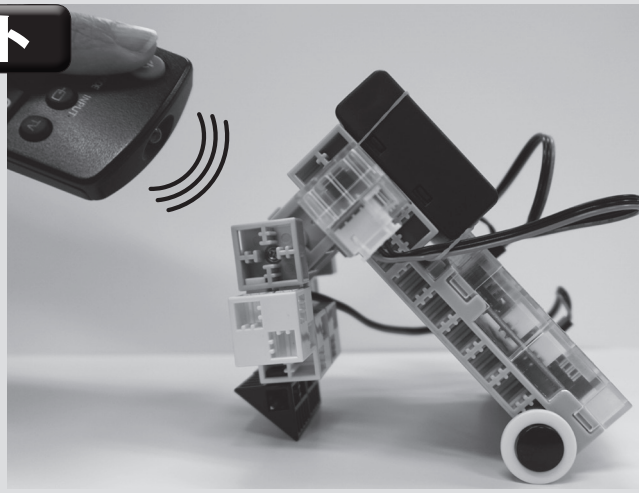
Scratchベースで動かそう!

Studuino（スタディーノで）ラクラク電子工作

第6回 シャクトリムシロボット



今回は、シャクトリムシのような動きをするロボットをつくってみよう。テレビやエアコンなどで使われている赤外線リモコンを使って、離れた場所からでもロボットを動かしたり止めたり、操作できるようにするぞ。

監修・原案／青山学院大学客員教授 阿部和広
協力／NPO法人 CANVAS　文／塩野祐樹

「コカねっと!」のスペシャルページで復習しよう

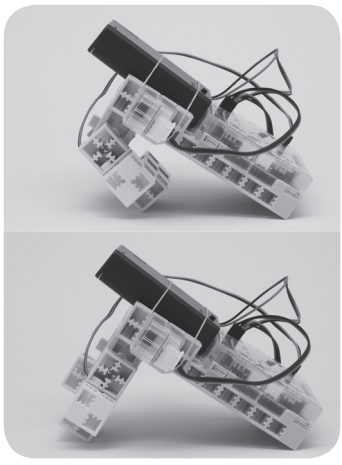
www.kodomonokagaku.com/magazine/studuino/

シャクトリムシロボットのしくみ

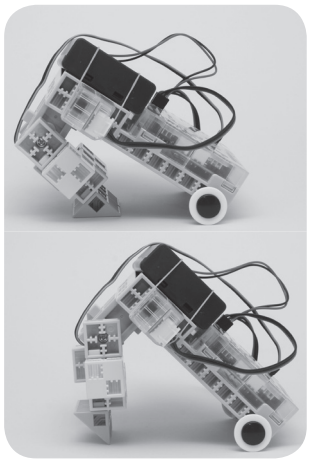
みんなはシャクトリムシを見たことがあるかな? シャクトリムシは、体の前後にある歩脚で交互に葉っぱなどを掴み、体をU字型に折り曲げたり伸ばしたりしながら進んでいく。

スタディーノでも、サーボモーターを使えば体を折り曲げたり伸ばしたりすることはできるけど、脚で地面を掴むことができない。だから、脚が滑ってしまい、前に進むことができないんだ。

そこで、ブロックの組み立て方や電池ボックスの位置を工夫して、体を折り曲げているときと伸ばしているときとで、体の重心と、前後の脚にかかる摩擦力が変わるようにした。これで、シャクトリムシのように前に進むことができるぞ!



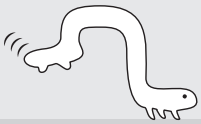
脚で地面を掴むことができないので、同じ場所で脚が動くだけで前に進めない。脚を動かしてるけど、電池ボックスの位置は変わっていないよね。



前脚となる丸い目のパーツは摩擦が小さく、後脚となる三角ブロックの角は摩擦が大きい。脚をグッと伸ばしたとき、電池ボックスが前に動いているのがわかる。重心や摩擦を工夫することで、前に進むようになったんだ。

シャクトリムシの歩き方

❶前脚を離して体を伸ばし、前方に掴まる。



❷後脚を離して体をU字型に曲げ、前脚の近くに掴まる。

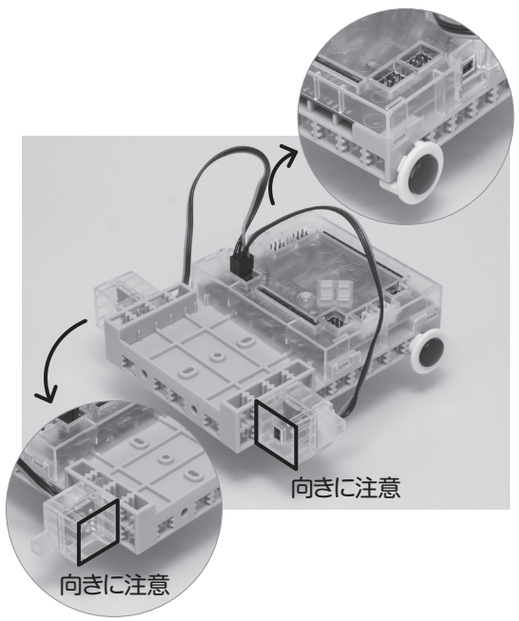
❸再び前脚を離して体を伸ばし、前に進んでいく。

ブロックの組み立てと配線

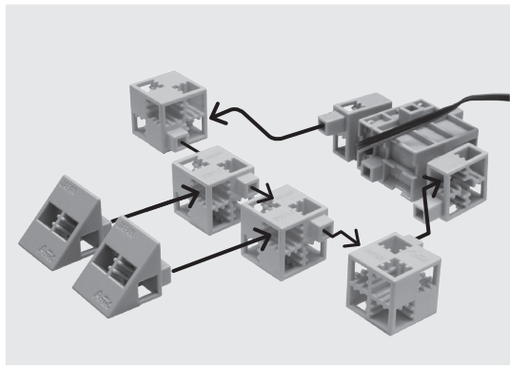
パーツ表

スタディーノ　ロボティスト用カバー台座付き…1個
LED…1個（色は自由）
赤外線フォトリフレクタ…1個
センサー接続ケーブル…2本
アーテックブロック　四角…4個、三角…2個、目…2個
サーボモーター…1個
電池ボックス…1個

まず、LED、赤外線フォトリフレクタ、目を、写真で示した台座の穴に取り付ける。LEDの色は何でもいいよ。LEDと赤外線フォトリフレクタの向きに注意しよう。センサーケーブルは、いつものように灰色の線が内側になるようにして、LEDはA4、赤外線フォトリフレクタはA5に差し込もう。

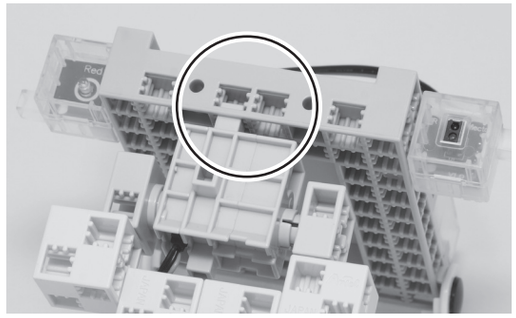


続いて、脚を組み立てるぞ。脚には、基本四角のブロックを4個、三角Aのブロックを2個、サーボモーターを使う。ポッチの位置に気をつけながら、写真にしたがって組み立てよう。



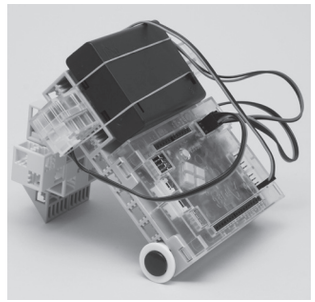
サーボモーターは、右がモーターの軸、左が自由に回転する軸だ。

出来上がったら、脚を台座の裏側に取り付ける。サーボモーターのケーブルはD9に、灰色の線が内側になるようにして差し込もう。



スイッチをOFFにした電池ボックスを台座に載せ、動かないように輪ゴムで止める。輪ゴムは、LEDと赤外線フォトリフレクタのブロックに引っかけるようにするといいぞ。電池ボックスのケーブルをPOWERコネクタに差し込めば、シャクトリムシロボットの完成だ!

輪ゴムが赤外線フォトリフレクタのレンズにかからないよう注意



赤外線リモコンのしくみ

テレビやエアコンのリモコンからは、赤外線が出ている（電波を使うものもある。赤外線を使うタイプには、赤外線を通すために窓や穴があるので探してみよう）。リモコンのボタンを押している間、ずっと赤外線が出ているのではなく、出たり出なかったりが高速で切り替わっている。このパターンをボタンごとに変えることで、どのボタンが押されたかわかるようになっているんだ。

今回は、ロボットを操作するためにリモコンを使うけど、このパターンを読み取るのではなく、単に赤外線が出ているかどうかを赤外線フォトリフレクタで調べることにする。だから、どのリモコンのどのボタンを押しても、同じように動くぞ。

ブロックプログラミング環境の準備と設定

スタディーノ基板とパソコンをUSBケーブルでつなぎ、パソコンでブロックプログラミング環境を起動しよう。［編集］メニューの［入出力設定…］を、配線した通りに設定しよう。

設定ができたら、電池ボックスのスイッチをONにして、［編集］メニューの［モーター校正…］で、ブロックでつくった脚が台座に対して垂直になるようにD9の角度を調整しよう。終わったら、電池ボックスのスイッチをOFFにしておこう。

赤外線リモコンのテスト

［実行］メニューから［テストモード開始］を選んで、テストモードに切り替えよう。すると、［テスト・ボード］の［A5］に赤外線フォトリフレクタの値が表示されるはずだ。まず、赤外線フォトリフレクタの前に手をかざして数値が変化することを確認しよう。

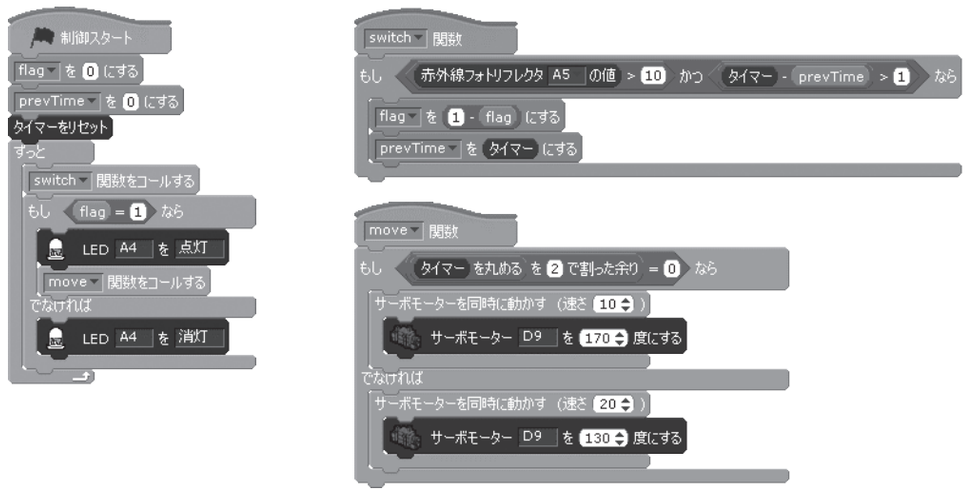
次に、テレビやエアコンなどの赤外線リモコンを赤外線フォトリフレクタに向けて、ボタンを押してみよう。このとき、テレビやエアコンに影響がないように、そっちにはリモコンを向けないよう注意しよう。

リモコンの種類や押すボタンによっても違うけど、値が大きくなったはずだ。この値は、リモコンと赤外線フォトリフレクタの距離や角度によっても変化する。ボタンを押していないときと、ボタンを押したときの値をメモして、スイッチを動作させる値をその中間くらいに決めよう。ここでは、その値を10として説明するよ。

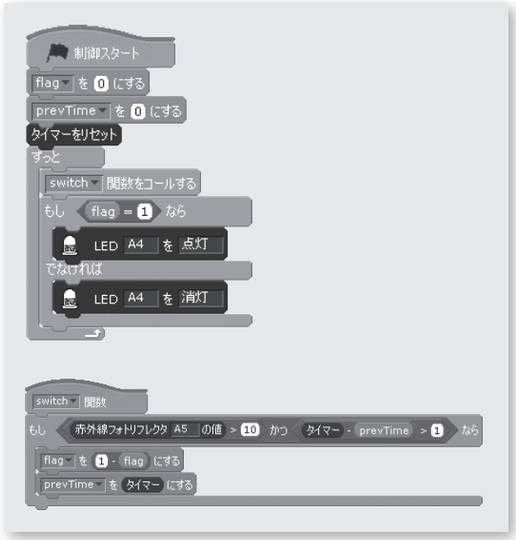
赤外線リモコンをスイッチにする

では手始めに、赤外線リモコンでLEDをオンオフするプログラムをつくってみよう。

今月のプログラム



●リモコンでLEDをオンオフするプログラム



まずはメインプログラムから。プログラムがスタートすると、まずは変数「flag」と変数「prevTime」の値を0にして、［タイマーをリセット］する。その後、［ずっと］の中の制御を、プログラムが終了するまで繰り返し実行するようになっているぞ。

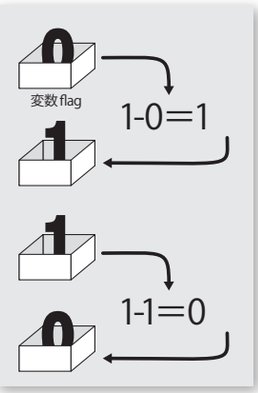
変数「flag」は、旗という意味で、LEDのスイッチの状態を表している。0は旗が倒れている状態でオフ、1は旗が立っている状態でオンを表す。最初は0を入れたので、スイッチはオフの状態からスタートする。

変数「prevTime」と［タイマーをリセット］の詳しい説明は後ですることにして、先に［ずっと］の中を見てみよう。

［ずっと］の中では、まず「switch」関数がコール（呼び出し）されている。この関数は、赤外線リモコンのボタンを押すたびに、変数「flag」の値を、0から1、1から0へと変えている。つまりスイッチのオン・オフを交互に切り替えているわけだ。このような動きを「トグル」と呼んでいる。

その次の［もし～なら～、でなければ～］では、変数「flag」の値に応じて、LEDを点けたり消したりしている。似たプログラムは前にもやったことがあるから大丈夫だよね。

続いて、「switch」関数の中身を見ていこう。このプログラムでいちばん重要なのは、「もし～なら」の中にある、変数「flag」の値を逆にする（トグルしている）しくみだ。



ここでは1から「flag」の値を引き、その結果を「flag」に入れなおしている。つまり、「flag」の値が0なら「1 - 0」で1になるので「flag」の値を1に、「flag」の値が1なら「1 - 1」で0になるので「flag」の値を0にする、というわけなのだ。

この入れ替えをするかどうかは、その直前にある「もし～」の条件に当てはまるかどうかで決まる。条件は次の2つだ。

①赤外線フォトリフレクタの値が10より大きい

②タイマー - prevTimeの値が1より大きい

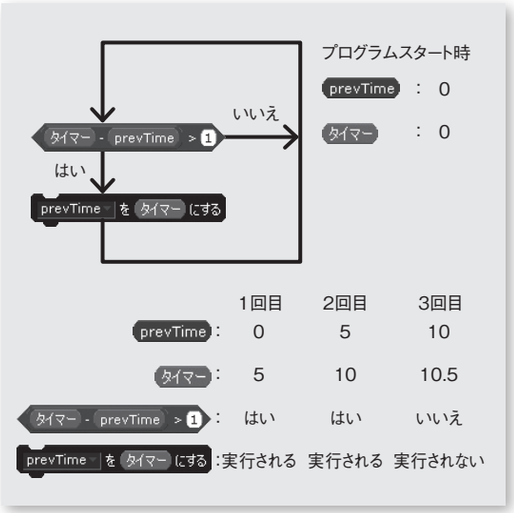
そして「もし～なら」の中の条件が「①かつ②」となっているので、この2つの条件が両方とも「はい」にならないと（満たされないと）、「もし～なら」の中は実行されない。

①は、赤外線フォトリフレクタの値が10より大きいか、つまり赤外線リモコンのボタンが押されているかどうかを調べている。押されていれば「はい」となり、条件は満たされる。

②は［タイマー］の値から変数「prevTime」の値を引き、その答えが1より大きいかどうかを調べている。［タイマー］はその言葉通り、時間を計るもので、［タイマーをリセット］してからの時間を計っている。これはメインプログラムの最初の方にあったね。そして変数「prevTime」の値も、メインプログラムの最初で0にしていた。

だから、最初に「switch」関数がコールされたときは、「プログラムがスタートしてからの秒数」-「0」が1以上になれば、条件は満たされて、「もし～なら」の中が実行される。

では2回目以降はどうだろう。「もし～なら」の中が実行されると、2行目で変数「prevTime」に「タイマー」の値が代入される。例えば［タイマーがリセット］されてから5秒後に「もし～なら」の中が実行されたとすると、変数「prevTime」の値は「5」となり、次に条件を満たすのは［タイマーがリセット］されてから6秒後以降になる。つまり、条件②は、前回「もし～なら」の中が実行されてから1秒以上経っているかどうかを見ているということなのだ。



変数「prevTime」と「タイマー」の値だけに注目すると、このような処理が行われている。プログラムがスタートすると、変数「prevTime」には「0」が入れられ、「タイマー」もリセットされて「0」になる。

1回目:1回目の条件判定が、プログラムスタートから5秒後に行われたとする。変数「prevTime」の値は「0」、「タイマー」の値は「5」なので、結果は「はい」となり、その後の処理が実行されて、変数「prevTime」に「タイマー」の値「5」が入る。

2回目:プログラムスタートから10秒後に2回目の条件判定が行われたとする。変数「prevTime」の値は「5」、「タイマー」の値は「10」なので、再び結果は「はい」となり、変数「prevTime」は「10」になる。

3回目:その直後の10.5秒後に3回目の判定が行われたとする。変数「prevTime」は「10」、「タイマー」は「10.5」なので、今回の結果は「いいえ」となる。だから、その後の処理は行われず、変数「prevTime」の値は「10」のままとなる。

こうして、前回の処理から1秒以上経たないと、次の処理が行われないようになっているんだ。

なんで、こんなややこしいことをしているかというと、リモコンのボタンを1回押しただけなのに、オンオフが何回も切り替わってしまうこと（チャタリングという）を防ぐためだ。このようなプログラムにすれば、前回ボタンを押してから1秒以内にボタンを押しても、オンオフの切り替えは行われない。

でも、それだけなら、「～秒待つ」でもいい気がするけど、ひとつ問題があるんだ。「～秒待つ」で待っている間は、プログラムが先に進まないということだ。Scratchと違って、ブロックプログラミング環境では、プログラムを1つしか書けない（複数のプログラムを同時に動かせない）。したがって、ここで動きが止まっている間は、他の処理もすべて止まってしまう。だから、このような工夫が必要になるんだ。これが、マイコンプログラミングの難しいところ。今は全部わからなくてもいいけど、心に留めておいてほしい。

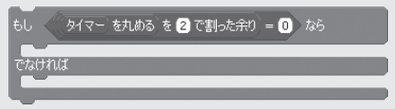
さて、このプログラムを入力し終わったら、緑の旗をクリックして実行してみよう。リモコンを赤外線フォトリフレクタに向けて、ボタンを押すとLEDが点き、もう一度押すと消えるだろうか。

シャクトリムシロボットを動かそう

リモコンによるスイッチができたところで、次にロボットを曲げたり伸ばしたりする動きについて考えてみよう。そのための関数を「move」として定義したよ（前ページ下の「今月のプログラム」を参照）。

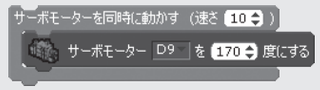
この関数では、ロボットの脚の角度を170°から130°の間で交互に動かしている。170°が最も縮んだ状態で、130°が最も伸びた状態だ。この角度は、実際に実験してみて、ちょうど良い動きになるように調べたものだ。

この曲げ伸ばしを制御しているのが、「もし～なら、でなければ～」だ。



ここでは、曲げ伸ばしを実行する条件として、［タイマー］の値を丸めた（四捨五入した）値を2で割った余りを使っている。余りが0なら脚を縮め、1なら伸ばしている。つまり、1秒おきに、曲げたり伸ばしたりを繰り返すようにしているんだ。ここでも、「～秒待つ」を使って関節が動く時間を確保しているのではないことに気をつけてほしい。

それから、サーボモーターの角度を変えるブロックは「サーボモーターを同時に動かす（速さ ～）」というブロックの中に入っているね。



このブロックは、本来、複数のサーボモーターを同時に動かすためのものだけど、動かすスピードを指定できるので、ここで使っている。

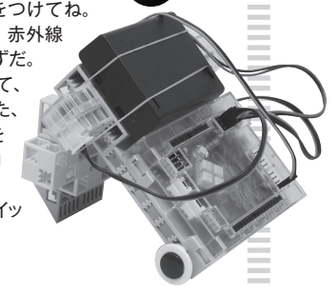
「速さ ～」で指定できるのは0から20までの値で、20が最も速い。ここでは、曲げるときはゆっくり、伸ばすときは素早く動かすことで、動きに非対称性を出しているよ。

出来上がった「move」関数はメインプログラムでLEDを点灯させた後にコールすればよいだろう。

これでプログラムは完成だ。

「実行」メニューから「プログラム作成・転送」を選んで、ロボットをパソコンから切り離せるようにしよう。転送が終わったら、USBケーブルを抜いて、電池ボックスのスイッチをONにしよう。

動画をコカねっと!でCHECK!

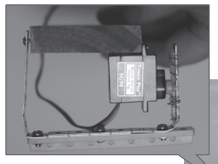


赤外線フォトリフレクタに向けて赤外線リモコンのボタンを押すと、LEDが点灯してシャクトリムシロボットが動き始めるぞ。だんだん遠ざかる向きに動くはずだ。意外と大きなアクションと音が出るぞ。うっかり指を挟まれないように気をつけてね。動きを止めたいときは、もう一度、赤外線リモコンのボタンを押せば止まるはずだ。

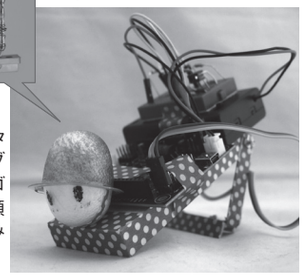
脚が動く角度や早さを変えてみて、動きがどう変わるか見てみよう。また、ブロックを組み替えて、進む向きを変えたり、他の方法で移動するロボットがつくれないか考えてみよう。遊び終わったら、電池ボックスのスイッチをOFFにしよう。

応用編 普通の電子部品でつくろう

前号の「ラーメンタイマー」応用編で使用した回路をそのまま使って、身近にあるものでシャクトリムシロボットをつくってみよう。



◀サーボモーターと金具を針金で固定してシャクトリムシの後脚をつくろう。



▶かまぼこ板の上にスタディーノ、電池ボックス、ブレッドボードを乗せて輪ゴムやテープで止めよう。顔をつくって載せたら虫みたいになった。

詳しくはコカねっと! のスタディーノページへGO

本連載の内容ができるキットがKoKa Shopで販売中!!

本連載で紹介するロボットや便利ツールをつくることができるキットが子供の科学のオンラインショップ「KoKaShop」で販売決定! 2016年1月号～8月号で紹介する連載内容が遊べるよ。

KoKaスタディーノプログラミングキット 1万800円（税込）

つくれるもの（予定）
- あっち向いてホイ（1～4月号）
- ラーメンタイマー（5月号）
- シャクトリムシロボット（6月号）
- 逃げる目覚まし時計（7、8月号）

このキット1つで、2016年1月号～8月号で紹介する連載内容が遊べるよ。

《キット内容》
- スタディーノ　ロボティスト用カバー台座付き　1個
- ロボット用LED 赤　1個
- ロボット用LED 緑　1個
- USBケーブル　miniB　1本
- センサー接続ケーブル　3本
- 電池ボックス　1個
- アーテックブロック　四角赤4、四角ペールオレンジ15、四角黒2、三角青2、三角白4、目2
- サーボモーター　1個
- ロボット用DCモーター　1個
- ロボット用赤外線フォトリフレクタ専用カバー付　1個
- タイヤ 2個
- タイヤゴム　2個

KoKaスタディーノ追加セット 3880円（税込）

「KoKaスタディーノ基本セット」を購入した人向けの追加キット。追加購入することで、さらに8月号の連載内容まで遊べるように。

※すでに「KoKaスタディーノ基本セット」をご購入された方のためのキットになります。初めての方は「プログラミングキット」をお買い求めください。

《キット内容》
- アーテックブロック　四角赤4、四角ペールオレンジ15、四角黒2、三角青2、三角白4、目2
- サーボモーター　1個
- ロボット用DCモーター　1個
- ロボット用赤外線フォトリフレクタ専用カバー付　1個
- センサー接続ケーブル　1本
- タイヤ 2個
- タイヤゴム　2個

子供の科学の通販サイト「KoKa Shop」 shop.kodomonokagaku.com