Time for fun!

Now that you know a few basics, it's time to put these simple bits together to do something fun. It's time to see how fast we can make Wink drive! Everyone enjoys a good race. In this lesson you'll apply the skills you've already learned to make Wink go though a racing sequence.

Before we start writing code, let's consider what we're trying to do. It's always a good idea to take a few minutes before you start writing code to plan out what the program needs to do. This will save you from writing unnecessary code. It will also help you keep your code more organized as you'll be following a plan.

I've come up with this list of steps to follow.

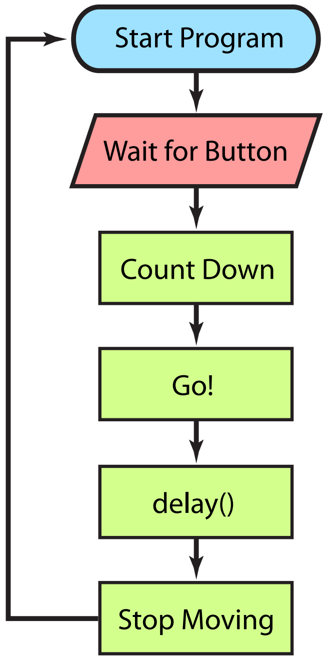
1) To begin our race, we'll want to wait for Wink's button to be pressed. This is what will start the racing sequence.

2) Once the button is pressed, we'll want some sort of count-down sequence. Kind of like "ready, set, go!"

3) After the count down, we'll want to make Wink start driving as fast as possible!

4) After the race goes for a while, we'll want to stop Wink's motors so he doesn't keep driving forever.

Planning your program can be done in many ways. You can make a simple list of steps as I've done above, or you may scribble a diagram of your program on a piece of paper (this is my own favorite method for planning things). For more complex programs, it is useful to draw a program flow chart that clearly shows all the steps of your program. I've created a flow chart of our racing program below.



Lets start with a basic example to get us started. Try this code.

```
void setup(){
  hardwareBegin();   //initialize Wink’s hardware
  playStartChirp(); //Play startup chirp and blink eyes
}

void loop(){
  waitForButton();   //wait here till’ button pressed

  //first blink
  eyesRed(100);      //make eyes red
  beep(1000);        //beep for 1 second
  eyesOff();         //turn eyes back off
  delay(200);

  //second blink
  eyesGreen(100);    //eyes green
  beep(25);          //quick start beep

  //GO!!
  motors(255,255);   //both motors max speed forward

  //wait for race to finish
  delay(1000);       //race for 1 second

  //STOP!
  beStill();         //stop moving
  eyesOff();         //turn eyes back off
}
```

Wink_Ch05Racing_Ex01

} Normal setup

} Wait for button to be pressed

} This section controls the first blink of the eyes. You can experiment with this part if you like.

} This second “blink” is the eyes going green and emitting a short beep right before starting to race.

} Drive at max speed!

} This section controls how long the “race” lasts before stopping.

} This section stops the motors. Don’t forget to turn the eyes back off also.

Load this code onto Wink and give him a try. You will likely notice two things when you try to run this. The first is that the robot probably doesn’t drive exactly straight. There is no way for Wink to know where he is facing - he’s just running his motors at max speed. With all mechanical devices (like robots), it will be true that one motor will run slightly faster than the other. It is also true that one motor will get slightly better grip with the running surface. We’ll consider this in a moment.

The second thing you will probably notice is that Wink may tend to “spin out” and loose control. This is because we’re very quickly going from motors not moving to motors moving very fast. This causes the motors to loose traction with the surface. Real race cars have the same problem if they accelerate too quickly.

Let’s consider how we may adjust a few things to solve these problems.

1 SKILL LEVEL

Controlling Acceleration...

When we run motors(255,255); we are causing the motors to instantly begin spinning at maximum speed. This causes the motor tips to loose traction with the surface, just like a race car doing a burnout. The trick is to increase the speed in a more controlled way. We can do this with the accelerateMotors() function.

This function requires you to include three values to make it work. The function basically goes from one motor speed to a different motor speed, over a period of time. We need to tell it what speed to start at, and what speed to end at. We also need to tell it how long it should take to move between these speeds. These are the three values.

You use the function like this...

```
accelerateMotors(startSpeed, endSpeed, timePeriod);
```

When you see a function, like accelerateMotors() described, you will often see descriptions of the values to be included with the function as words that describe what the values do. In the above example, I've written "startSpeed" in the space where the first value is to be written. When you write your code, you don't actually put the words "startSpeed" in this space - instead you put an actual number, like 0, or 100, or something else. We just use the words to describe what each value is used for, and what order to place the values into the function. In this case we see the first value is the starting speed.

As we're sitting still when we begin, we'll put a zero here in our actual code. The next value is the ending speed. This is the speed the motors will be going when the function completes. The final value is timePeriod, which is the amount of time (in milliseconds) Wink will take to accelerate between the startSpeed and the endSpeed.

Lets look at an actual example then describe what it will do.

```
accelerateMotors(0,100,500);
```

} Start at zero speed, and accelerate to 100 speed over a period of 500 milliseconds (which is 1/2 second).

In this example, when the function is called, it will set both motors to zero speed. The motors will then change to speed 1, then speed 2, and so on. It will spread this increase in speed evenly over the given time period. Long time periods will cause a very gradual acceleration, while short time periods will make the acceleration very quick.

In our racing example, if we make the acceleration take too long, Wink will take longer to get up to maximum speed and he may loose the race to a faster robot. On the other hand, if we make this acceleration too short, Wink will very quickly speed up his motors and loose traction. The trick to making your Wink the fastest is to find a good number that gives fast acceleration while still maintaining traction with the surface. You'll have to experiment with this on your own to find the best value.

I have found that an acceleration from 0 to 240, a timePeriod of about 350 works well.

Lets add this to our previous example and give it a try. We should see that Wink is less likely to spin out and loose control.

```
void setup(){
  hardwareBegin();  //initialize Wink’s hardware
  playStartChirp(); //Play startup chirp and blink eyes
}

void loop(){
  waitForButton();  //wait here till’ button pressed

  //first blink
  eyesRed(100);     //make eyes red
  beep(1000);       //beep for 1 second
  eyesOff();        //turn eyes back off
  delay(200);

  //second blink
  eyesGreen(100);   //eyes green
  beep(25);         //quick start beep

  //GO!!
  accelerateMotors(0,240,350); //accelerate in a controlled way
  motors(255,255);             //both motors max speed forward

  //wait for race to finish
  delay(1000);      //race for 1 second

  //STOP!
  beStill();        //stop moving
  eyesOff();        //turn eyes back off
}
```

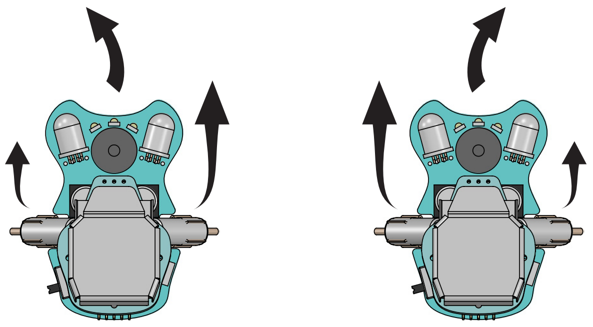
Wink_Ch05Racing_Ex02

} Add the accelerateMotors() function here. This function will run first causing Wink to accelerate in a controlled way. After the acceleration is completed, the next function motors(255,255) will run which will make both motors begin running at maximum speed forward.

Making the path more straight...

As we have discussed, when two wheels (or bug feet) are driven by two different motors, it is true that one motor will always run a bit faster than the other. There are very slight differences in the internal structure of each individual motor when manufactured, so we will always see some difference even if they are powered from the same battery. Each Wink robot will tend to behave differently in this way. You may get extremely lucky and find that your Wink wants to drive in a generally straight line when both motors are set to the same speed, but more than likely, you'll realize your Wink always wants to steer to the right, or maybe, to the left.

We can make him less likely to steer in this direction if we adjust the speed of one of the motors in our code. Have a look at the picture below.



As you can see from this picture, if your Wink tends to steer to the left, it is because his RIGHT motor is moving a bit faster than the left. The opposite is true if he wants to steer toward the right. If he always wants to steer to the right, it is because his LEFT motor is moving a bit faster.

We can adjust for this in our code. We can set different speed values for each motor in the motors() function in the GO! section of our program. Try making the changes below depending on which direction your Wink wants to go.

If your Wink wants to steer toward the LEFT, then try this change...

```
motors(255,245);  //make the RIGHT motor a bit slower
```

} Slowing the RIGHT motor will correct if Wink wants to steer left.

If your Wink wants to steer toward the RIGHT, then try this change...

```
motors(245,255);  //make the LEFT motor a bit slower
```

} Slowing the LEFT motor will correct if Wink wants to steer right.

1 SKILL LEVEL

With a bit of trial and error, you will eventually arrive at values for the motors() function that will produce a path that is generally straight for your specific Wink robot. You shouldn't have to make the values different by more than 30 in most cases. Try adjusting in steps of 5 until you dial in on the best speed for your robot.

Racing Winks!!

To race the robots, pair up with a friend who also has a Wink robot. Find a nice long racing surface like a long table or an open area of the floor. If you do race on the table, be sure to have some helpers stationed along the table to catch any Winks that try to jump off the race course. If you do race on the floor, try to designate a safe "race track" area and ask others to avoid walking in that area. We don't want any smushed bugs!

Note that you will also want to adjust the delay(1000); value in the "wait for race to finish" section depending on your racing area. If you have a big open strip of clear floor, you can make this value larger so the Winks will run a longer distance. If you're limited to a small space on a table, then reduce this a bit so the robots are less likely to jump off the end. Remember that delay(1000); is one second. To make the race run for 2 seconds, you would change this to delay(2000);.

Set up your Wink robots at one end of the race area, then one person can count off "On your mark, get set, GO!". Each racer should push the button on their Wink when they hear "GO!". You'll then have about one second to make sure your Wink is pointed straight down the track before they take off.

Sometimes you'll get lucky and have a nice clean run. Sometimes you'll still spin out, and sometimes you'll still veer off the track one way or another. This happens in real drag racing as well. There's no telling how a given race will end.

Between races you can continue to tune your code to help your Wink run at his best. If your Wink tends to turn a bit on launch then goes straight, you're probably still loosing traction and accelerating the motors too fast. If this happens, try making the timePeriod a bit longer in your accelerateMotors() function. If you're not having any trouble loosing traction, you can reduce this time a bit. This will result in a faster acceleration and a possible early lead, but you'll also be more likely to spin out. It's your call.

You may also realize that a lower top speed can result in a more controlled run. If you drive straight to the finish line you still win if your opponent looses control and leaves the race course.

Feel free to play with the values in this example all you want. You can change the color of Wink's eyes during the race by using the eyes functions you learned earlier. You can also make different eye blink sequences to flash before launching in the count down section. Get creative and have fun!

Eventually we'll learn about a cool programming idea called a "for" loop, which does a certain thing 'for' a certain amount of times. We'll eventually use the "for" loop to make the starting sequence a bit more interesting.