Marty Scratch Examples

This document shows a selection of things you can do with Marty through the Scratch interface, from the very basic to the fairly complicated. Everything here is in prototype stage, but hopefully it's a good start!

I've made some quick videos showing the scripts being run on a Marty. They're very much not a finished article (and apologies for the christmas music playing in the background), but you can see what each script looks like when it runs: https://www.youtube.com/playlist?list=PLiBgsWjIRfInizbzagjLnoQjoeb0psS33 And the individual videos are linked below

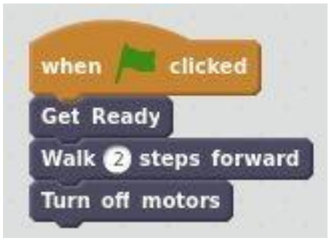
We'll be making starter projects for many of the examples, to get people started. Those should be appearing at www.robotical.io/scratch shortly

Basic commands

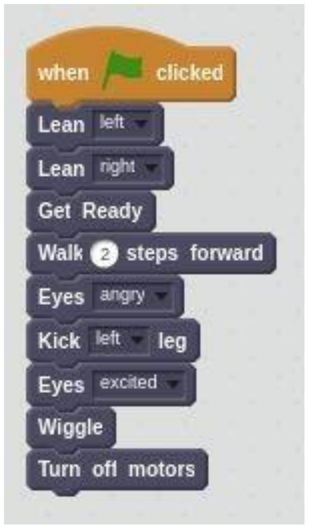
Get used to controlling Marty from scratch by playing around with the built in functions

Controlling Marty

The "Get Ready" block will return Marty to a normal standing position and do a little eyebrow wiggle. The "Turn off motors" block will… turn off the motors. It's good to call it at the end of a script.


```
when [flag] clicked
Get Ready
Walk (2) steps forward
Turn off motors
```

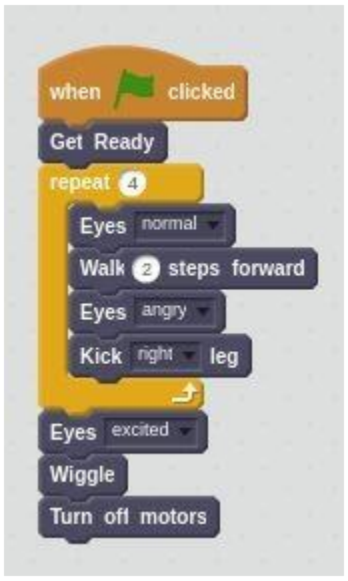
Longer chains of command (Video)



```
when [flag] clicked
Lean left
Lean right
Get Ready
Walk 2 steps forward
Eyes angry
Kick left leg
Eyes excited
Wiggle
Turn off motors
```

Have a play around with making Marty do more complicated things. Make a little dance routine or a play.

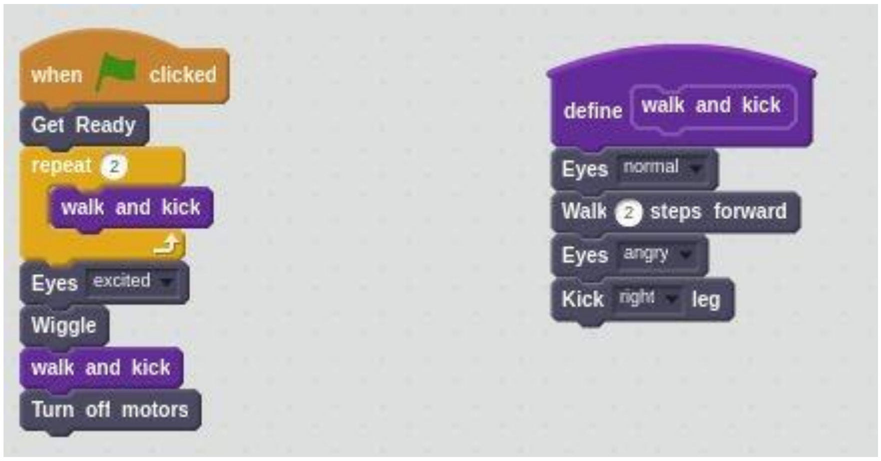
Looping (Video)



```
when [flag] clicked
Get Ready
repeat 4
    Eyes normal
    Walk 2 steps forward
    Eyes angry
    Kick right leg
Eyes excited
Wiggle
Turn off motors
```

Sometimes you want to do the same thing a few times, but copying and pasting code looks messy and is time consuming. Have a play with the “repeat” blocks to get the hang of looping

Define a function (Video)



```
when [flag] clicked
Get Ready
repeat (2)
    walk and kick
end
Eyes [excited]
Wiggle
walk and kick
Turn off motors
```

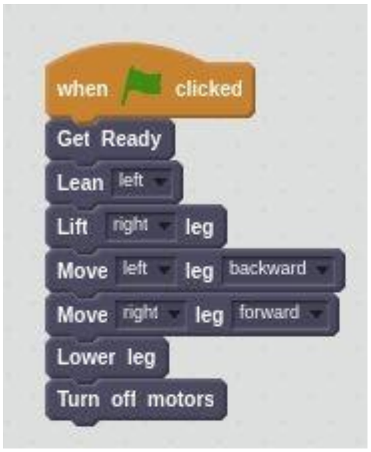
```
define walk and kick
Eyes [normal]
Walk (2) steps forward
Eyes [angry]
Kick [right] leg
```

Another way to avoid repeating code is to make a “function”

Make Marty walk!

There is a built in walk function, but that’s no fun. In this challenge you have to teach Marty to walk! You’ll have to think about balance and how individual joints need to move to create a complex movement.

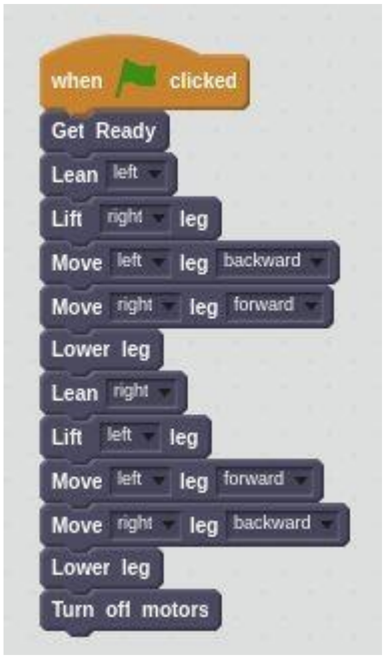
Take a step (Video)



```
when [flag] clicked
Get Ready
Lean [left]
Lift [right] leg
Move [left] leg [backward]
Move [right] leg [forward]
Lower leg
Turn off motors
```

The basic movements for taking a step - in this case with the right leg

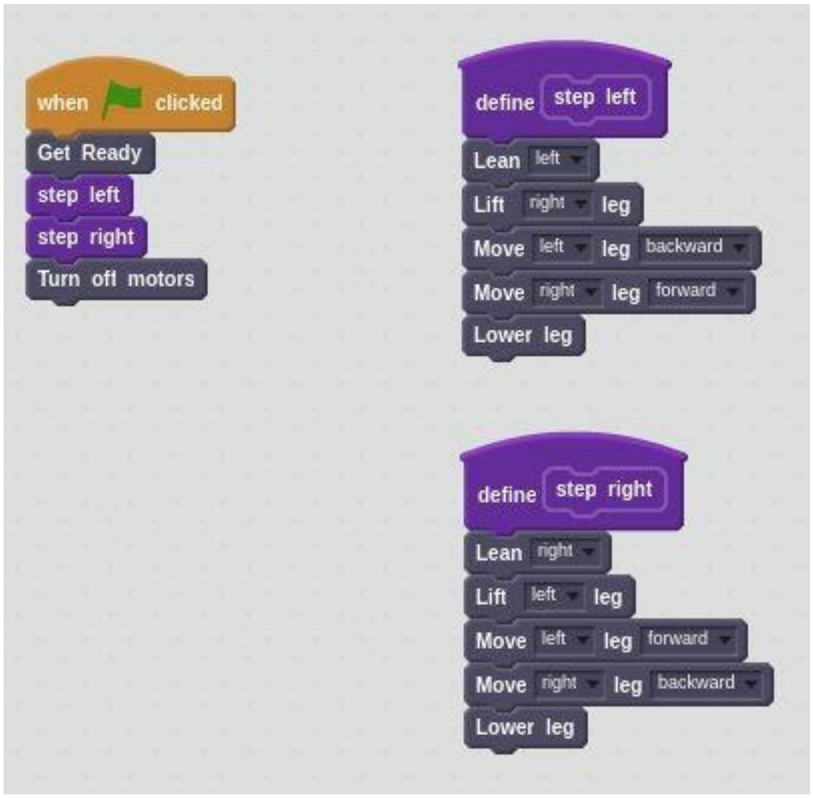
Take two (Video)



```
when [flag] clicked
Get Ready
Lean left
Lift right leg
Move left leg backward
Move right leg forward
Lower leg
Lean right
Lift left leg
Move left leg forward
Move right leg backward
Lower leg
Turn off motors
```

And then add a step with the other leg

Step functions (Video)



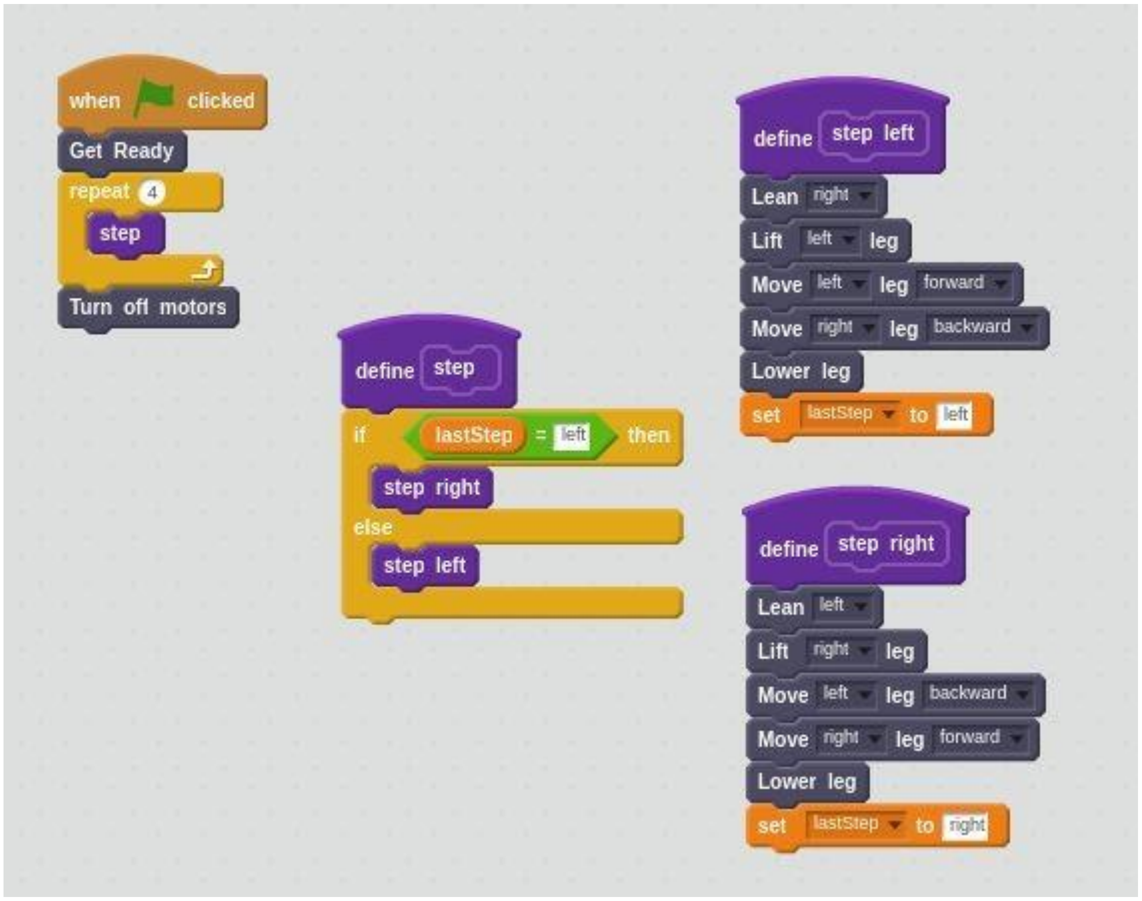
```
when [flag] clicked
Get Ready
step left
step right
Turn off motors
```

```
define step left
Lean [left]
Lift [right] leg
Move [left] leg [backward]
Move [right] leg [forward]
Lower leg
```

```
define step right
Lean [right]
Lift [left] leg
Move [left] leg [forward]
Move [right] leg [backward]
Lower leg
```

Move the individual steps into their own functions

Put the best foot forward (Video)

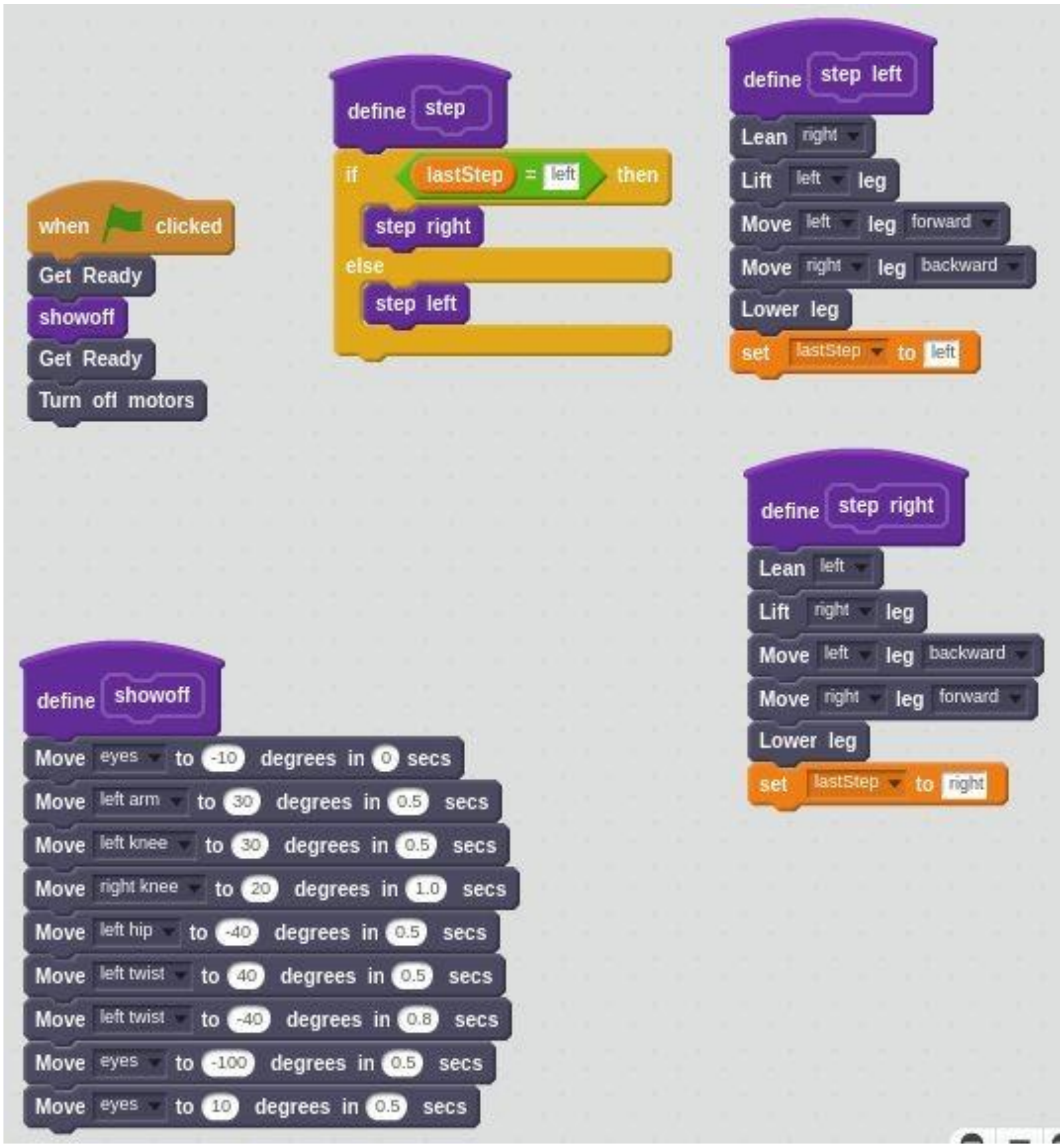


```
when ⚑ clicked
Get Ready
repeat 4
    step
Turn off motors

define step
if lastStep = left then
    step right
else
    step left

define step left
Lean right
Lift left leg
Move left leg forward
Move right leg backward
Lower leg
set lastStep to left

define step right
Lean left
Lift right leg
Move left leg backward
Move right leg forward
Lower leg
set lastStep to right
```

It would be nicer to just be able to say “step” rather than specifically stating which leg to step with. This can be done by creating a variable to keep track of which foot the last step used

Source : https://robotical.io/

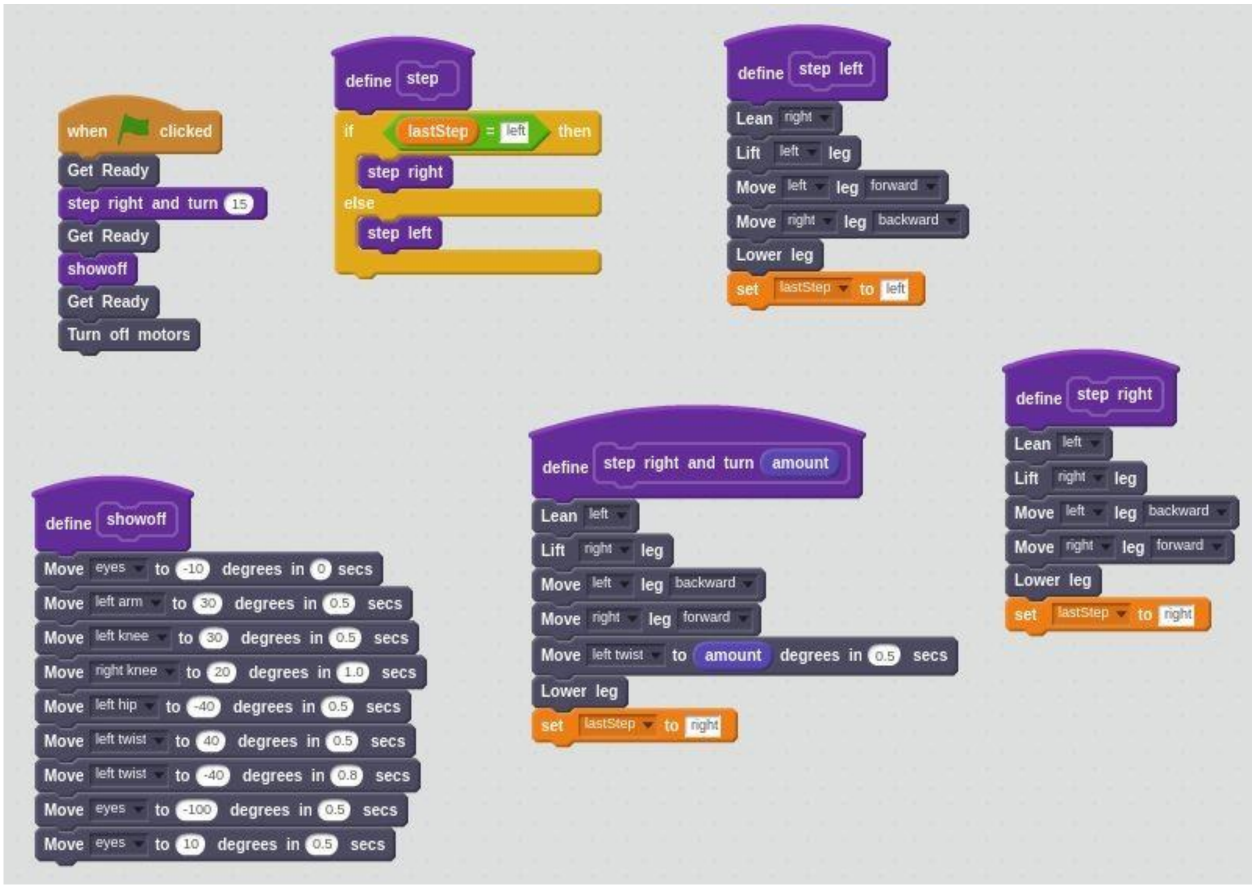
Make your own movement (Video)



```
when clicked
Get Ready
showoff
Get Ready
Turn off motors

define step
if lastStep = left then
    step right
else
    step left

define step left
Lean right
Lift left leg
Move left leg forward
Move right leg backward
Lower leg
set lastStep to left

define step right
Lean left
Lift right leg
Move left leg backward
Move right leg forward
Lower leg
set lastStep to right

define showoff
Move eyes to -10 degrees in 0 secs
Move left arm to 30 degrees in 0.5 secs
Move left knee to 30 degrees in 0.5 secs
Move right knee to 20 degrees in 1.0 secs
Move left hip to -40 degrees in 0.5 secs
Move left twist to 40 degrees in 0.5 secs
Move left twist to -40 degrees in 0.8 secs
Move eyes to -100 degrees in 0.5 secs
Move eyes to 10 degrees in 0.5 secs
```

Have a play with the “Move Joint” block to create a more complicated custom movement function

A turning point (Video)



```
when [flag] clicked
Get Ready
step right and turn (15)
Get Ready
showoff
Get Ready
Turn off motors
```

```
define step
if <(lastStep) = [left]> then
    step right
else
    step left
```

```
define step left
Lean [right]
Lift [left] leg
Move [left] leg [forward]
Move [right] leg [backward]
Lower leg
set [lastStep] to [left]
```

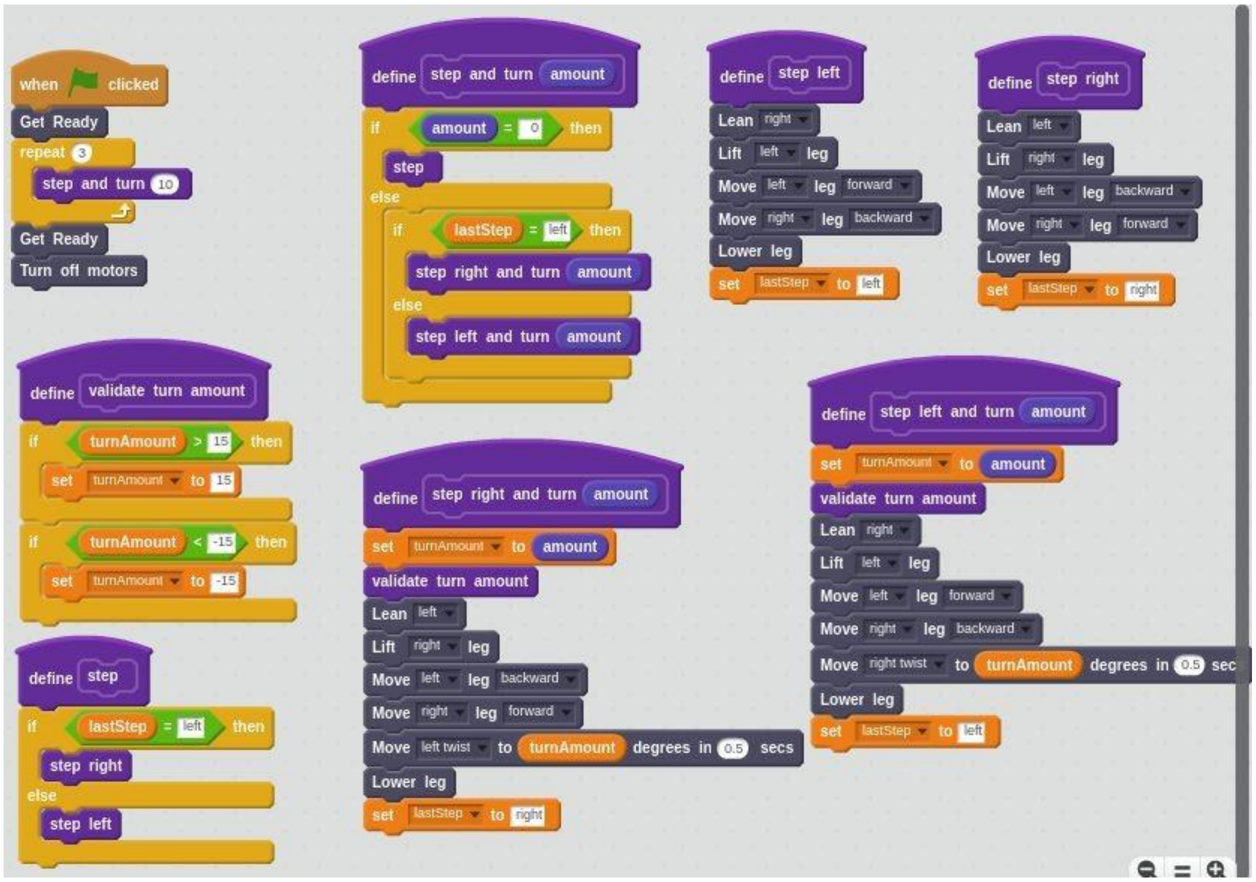
```
define step right
Lean [left]
Lift [right] leg
Move [left] leg [backward]
Move [right] leg [forward]
Lower leg
set [lastStep] to [right]
```

```
define step right and turn (amount)
Lean [left]
Lift [right] leg
Move [left] leg [backward]
Move [right] leg [forward]
Move [left twist] to (amount) degrees in (0.5) secs
Lower leg
set [lastStep] to [right]
```

```
define showoff
Move [eyes] to (-10) degrees in (0) secs
Move [left arm] to (30) degrees in (0.5) secs
Move [left knee] to (30) degrees in (0.5) secs
Move [right knee] to (20) degrees in (1.0) secs
Move [left hip] to (-40) degrees in (0.5) secs
Move [left twist] to (40) degrees in (0.5) secs
Move [left twist] to (-40) degrees in (0.8) secs
Move [eyes] to (-100) degrees in (0.5) secs
Move [eyes] to (10) degrees in (0.5) secs
```

Make a function which steps with a turn, this function can take an argument to define the amount of turning

Parameter validation, and function-ception (Video)



```
when clicked
Get Ready
repeat 3
    step and turn 10
Get Ready
Turn off motors

define step and turn amount
if amount = 0 then
    step
else
    if lastStep = left then
        step right and turn amount
    else
        step left and turn amount

define step left
Lean right
Lift left leg
Move left leg forward
Move right leg backward
Lower leg
set lastStep to left

define step right
Lean left
Lift right leg
Move left leg backward
Move right leg forward
Lower leg
set lastStep to right

define validate turn amount
if turnAmount > 15 then
    set turnAmount to 15
if turnAmount < -15 then
    set turnAmount to -15

define step
if lastStep = left then
    step right
else
    step left

define step right and turn amount
set turnAmount to amount
validate turn amount
Lean left
Lift right leg
Move left leg backward
Move right leg forward
Move left twist to turnAmount degrees in 0.5 secs
Lower leg
set lastStep to right

define step left and turn amount
set turnAmount to amount
validate turn amount
Lean right
Lift left leg
Move left leg forward
Move right leg backward
Move right twist to turnAmount degrees in 0.5 sec
Lower leg
set lastStep to left
```

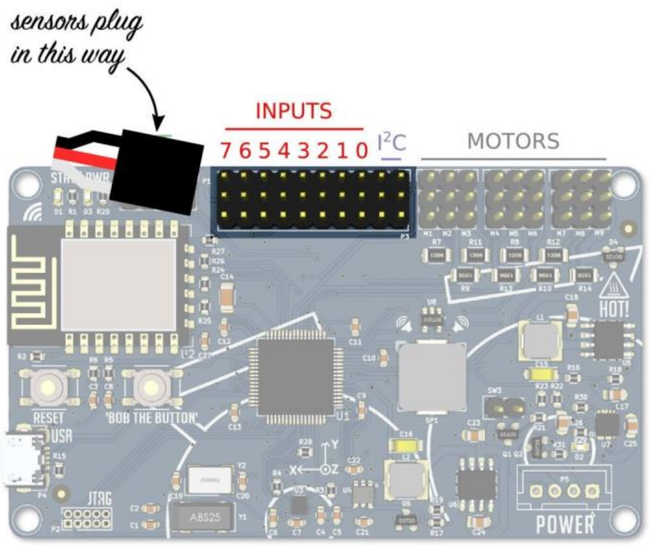
There are a couple of things going on here:

- Parameter validation: If you tell Marty to turn too much, his feet will hit each other and bad things will happen. To avoid accidentally sending a bad command, we can “bound” the turn amount to within certain limits
- Making more general functions. Functions can call other functions - in this case the “step and turn” function wraps the “step” function, as well as adding some functionality.

Sensors

Marty has a few sensors built in - electrical current sensing on most of the motors, an accelerometer for measuring tilt and acceleration, and a set of ports to which sensors like bump switches can be attached.

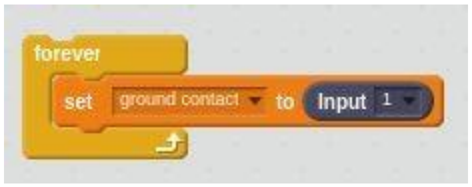
The electronics wiring diagram below shows how sensors can be attached, but these examples assume there is a bump switch on port 0 which is at the front of a foot, and a bump switch on port 1 which is on the bottom of a foot.



Connecting sensors to Marty. Cables go in with the black wire at the top, and the numbers here correspond to the "Input *n*" block in Scratch.

Reading a sensor (Video)

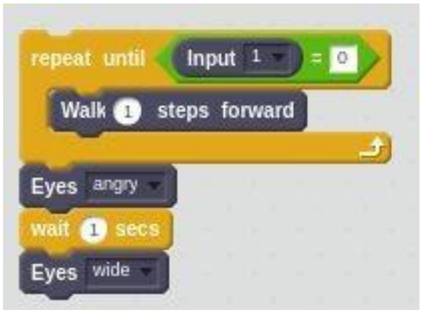
The "Input" block will let you read the current state of a switch. By assigning the sensor reading to a variable you can observe it in Scratch.
The switch will read "1" when pressed, and "0" when not pressed


```
forever
    set ground contact to Input 1
```

Walking to the edge (Video 1) (Video 2)

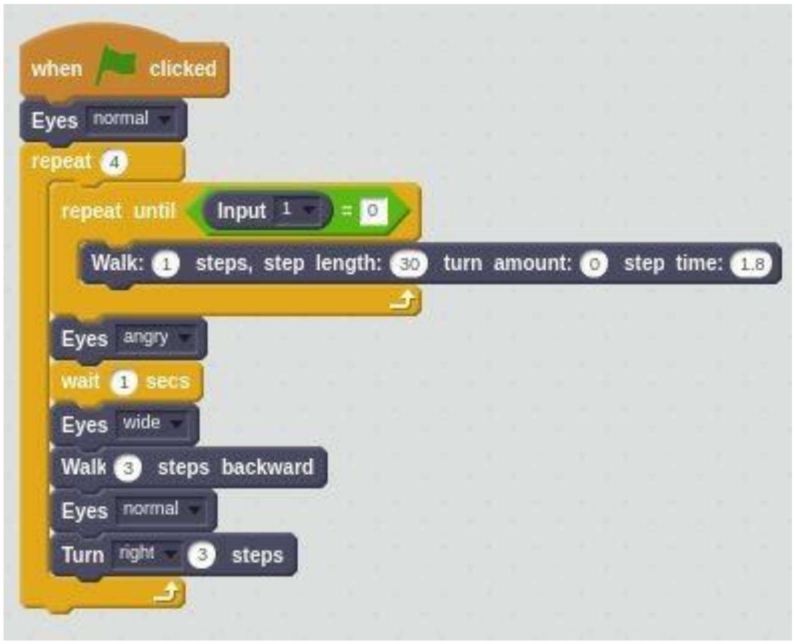
This example assumes there is a bump switch on the bottom of a foot and attached to port 1. At the end of each step, the switch can be checked to see if it is pressed. If it is a "1", the foot is touching the ground, if it is a "0" then it's not!
We can use this to check if Marty has come to a ledge

Source : https://robotical.io/



```
repeat until <(Input 1) = 0>
    Walk (1) steps forward
end
Eyes angry
wait (1) secs
Eyes wide
```

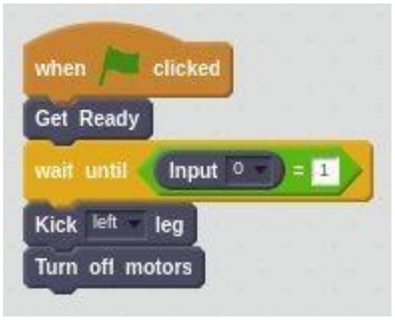
Walking back from the edge (Video)



```
when [flag] clicked
Eyes normal
repeat (4)
    repeat until <(Input 1) = 0>
        Walk: (1) steps, step length: (30) turn amount: (0) step time: (1.8)
    end
    Eyes angry
    wait (1) secs
    Eyes wide
    Walk (3) steps backward
    Eyes normal
    Turn right (3) steps
end
```

It’s useful to do something a bit better than just reacting. Add some more blocks to make Marty do something useful when he reaches an edge. In the example above he’ll walk back from the edge and turn around before walking more

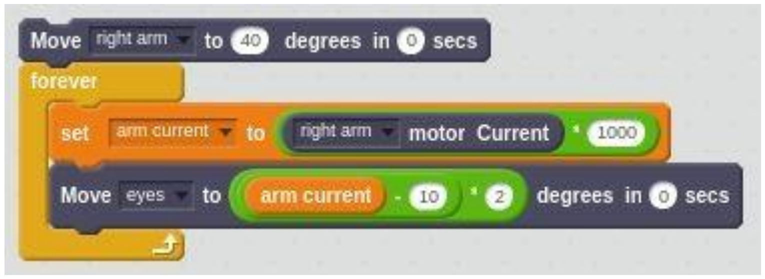
Responding to a button press (Video)



```
when [flag] clicked
Get Ready
wait until <(Input 0) = 1>
Kick left leg
Turn off motors
```

This is a simple example of reacting to a button being pushed. Marty will wait until something touches the bump switch and will then kick

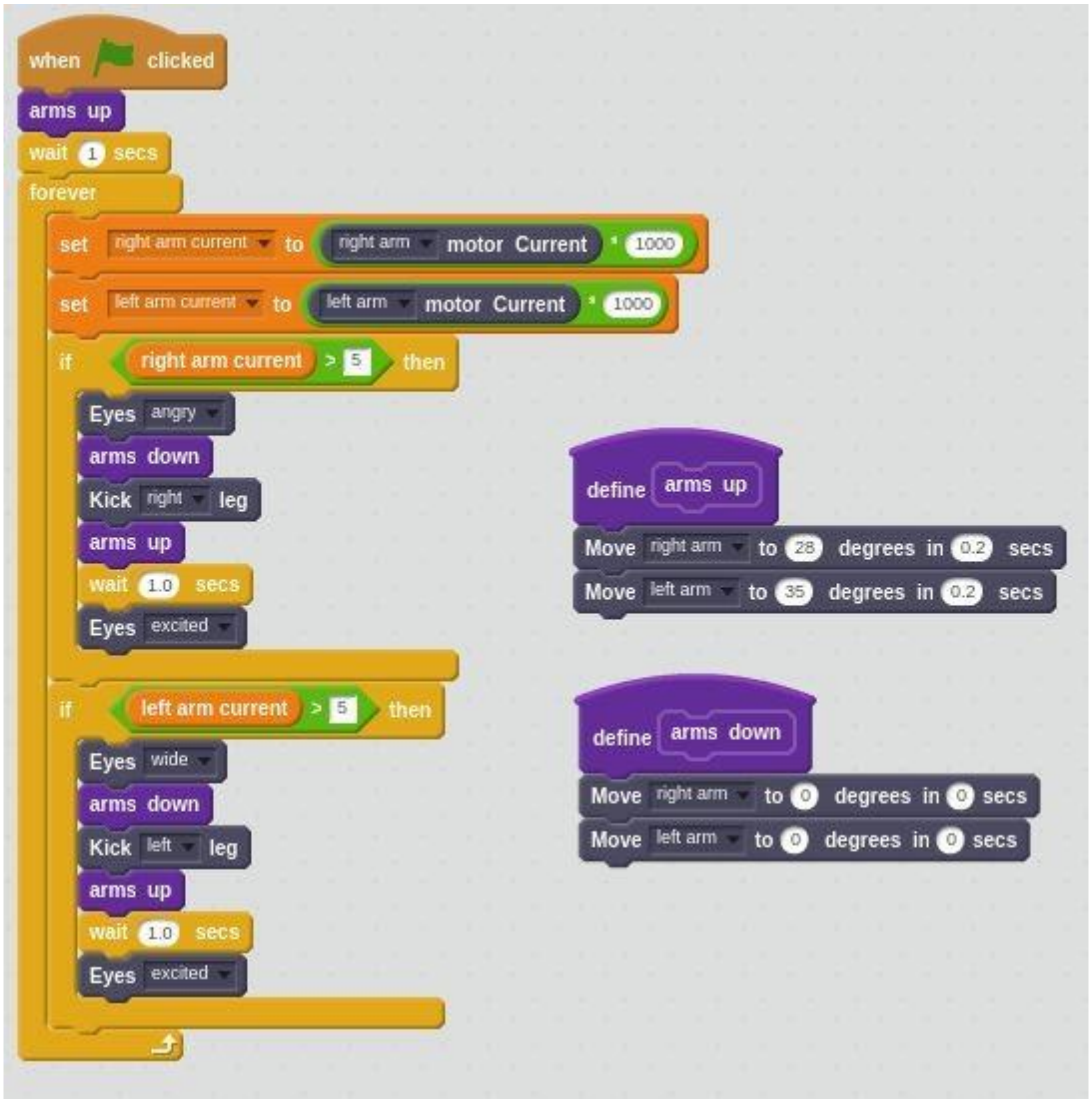
Reading motor current (Video)



```
Move [right arm] to (40) degrees in (0) secs
forever
    set [arm current] to ([right arm] motor Current * 1000)
    Move [eyes] to ((arm current - 10) * 2) degrees in (0) secs
```

Motor current sensing lets you see roughly how much torque a motor is producing. The sensor reads a small value so you'll need to multiply it up as shown above.
In this example we link the torque on the right arm to the eyebrow position - push Marty's arm or put something heavy on it and he'll get angry!

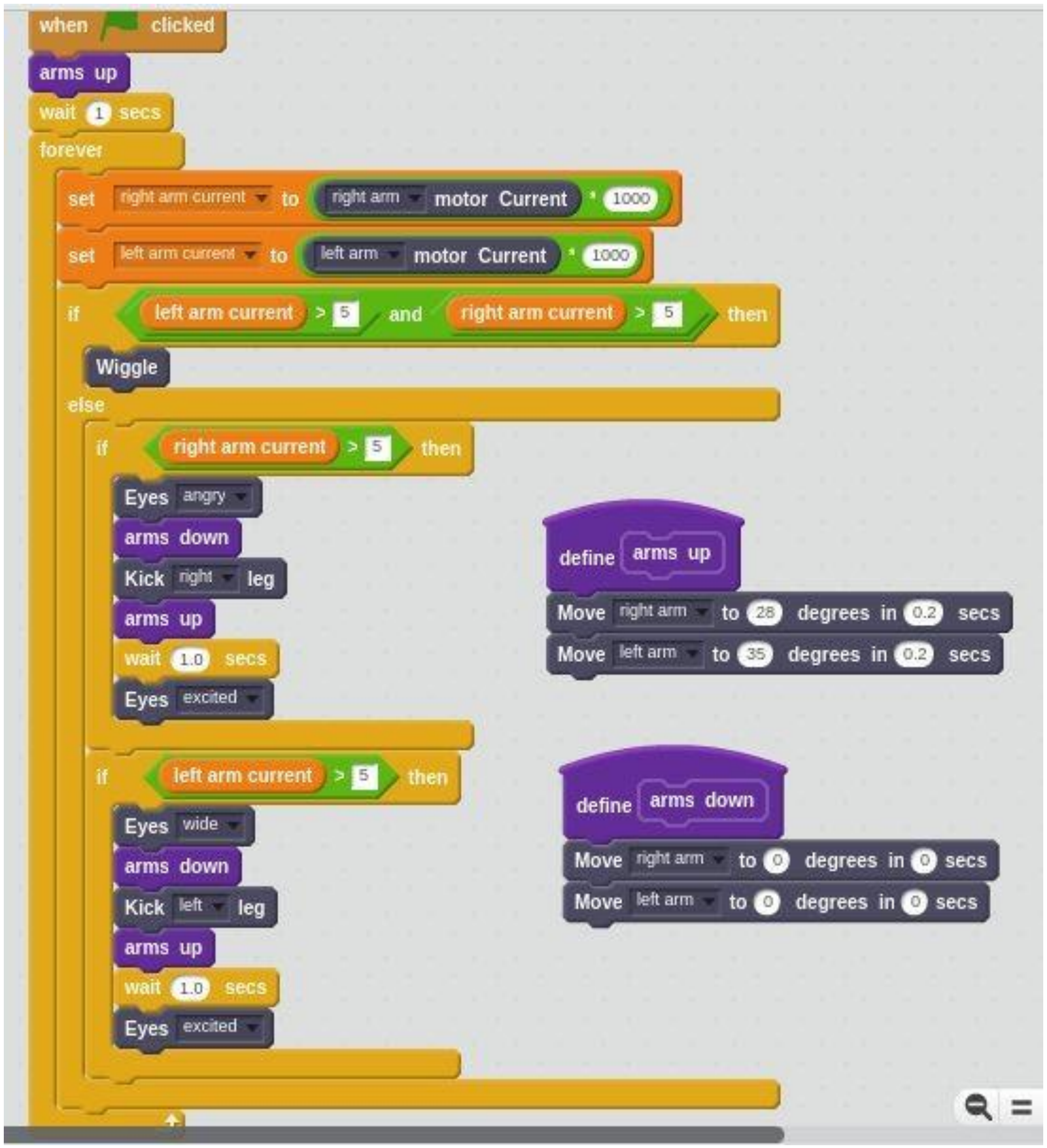
Using motor current as an input (Video)



```
when [flag] clicked
arms up
wait 1 secs
forever
    set right arm current to (right arm motor Current * 1000)
    set left arm current to (left arm motor Current * 1000)
    if right arm current > 5 then
        Eyes angry
        arms down
        Kick right leg
        arms up
        wait 1.0 secs
        Eyes excited
    if left arm current > 5 then
        Eyes wide
        arms down
        Kick left leg
        arms up
        wait 1.0 secs
        Eyes excited

define arms up
Move right arm to 28 degrees in 0.2 secs
Move left arm to 35 degrees in 0.2 secs

define arms down
Move right arm to 0 degrees in 0 secs
Move left arm to 0 degrees in 0 secs
```

In this example, we’re letting somebody give Marty instructions by pushing down on one of his arms. Pushing the right arm will make him kick his right leg, and pushing the left arm will make him kick the left leg.

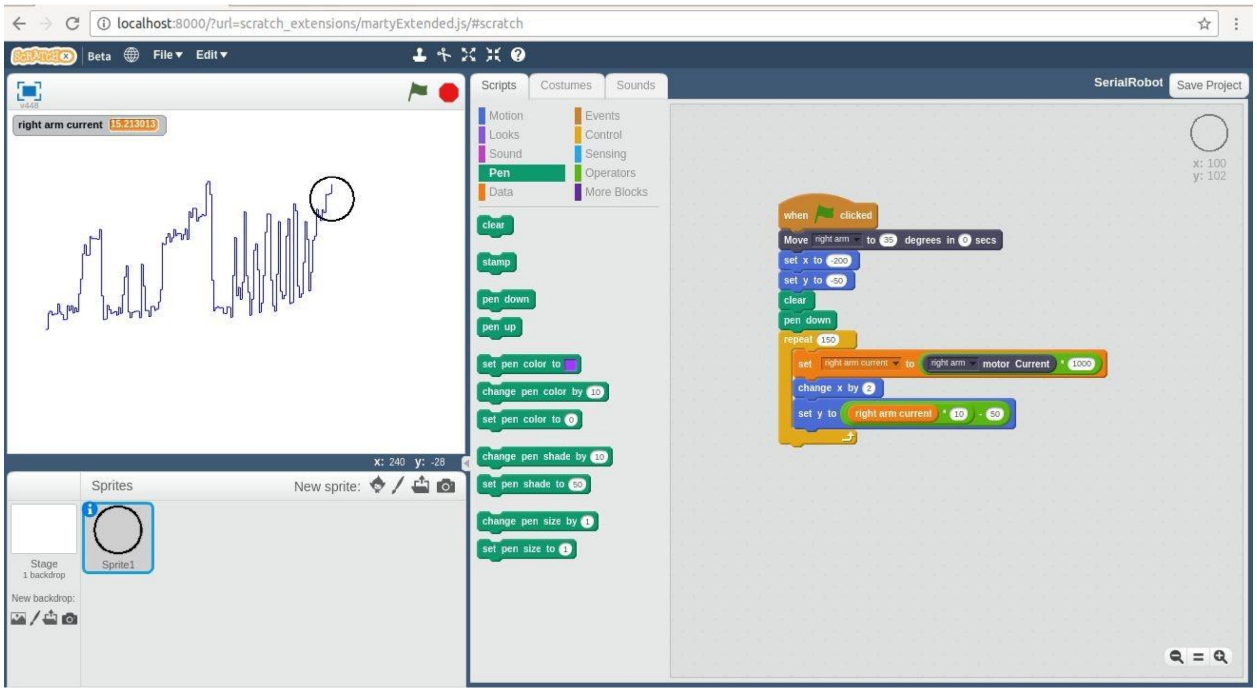
AND behaviour (Video)



```
when [flag] clicked
arms up
wait 1 secs
forever
    set right arm current to (right arm motor Current * 1000)
    set left arm current to (left arm motor Current * 1000)
    if <<left arm current > 5> and <right arm current > 5>> then
        Wiggle
    else
        if <right arm current > 5> then
            Eyes angry
            arms down
            Kick right leg
            arms up
            wait 1.0 secs
            Eyes excited
        if <left arm current > 5> then
            Eyes wide
            arms down
            Kick left leg
            arms up
            wait 1.0 secs
            Eyes excited

define arms up
Move right arm to 28 degrees in 0.2 secs
Move left arm to 35 degrees in 0.2 secs

define arms down
Move right arm to 0 degrees in 0 secs
Move left arm to 0 degrees in 0 secs
```

If you want to get really snazzy you can make Marty do something different when *both* arms are pushed at the same time

Getting Graphic (Video)



```
when green flag clicked
Move (right arm) to (35) degrees in (0) secs
set x to (-200)
set y to (-50)
clear
pen down
repeat (150)
    set [right arm current] to (([right arm] motor Current) * (1000))
    change x by (2)
    set y to (((right arm current) * (10)) - (50))
```

Scratch is also cool for displaying sensor readings. In this example we’re showing the output from one of the motor current sensors as a graph.
To do this we create a sprite which is a simple circle in the center, then animate it with a pen