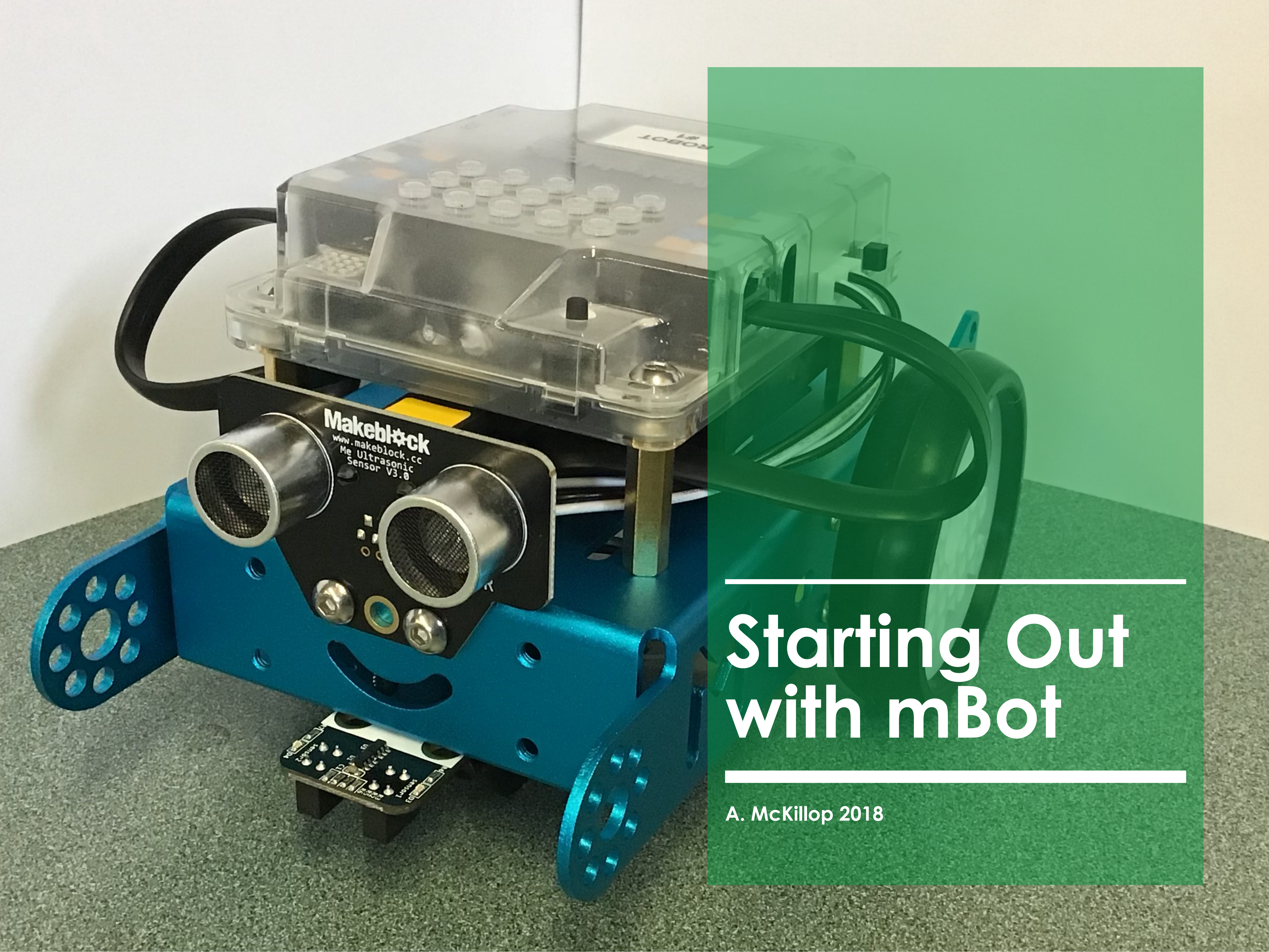


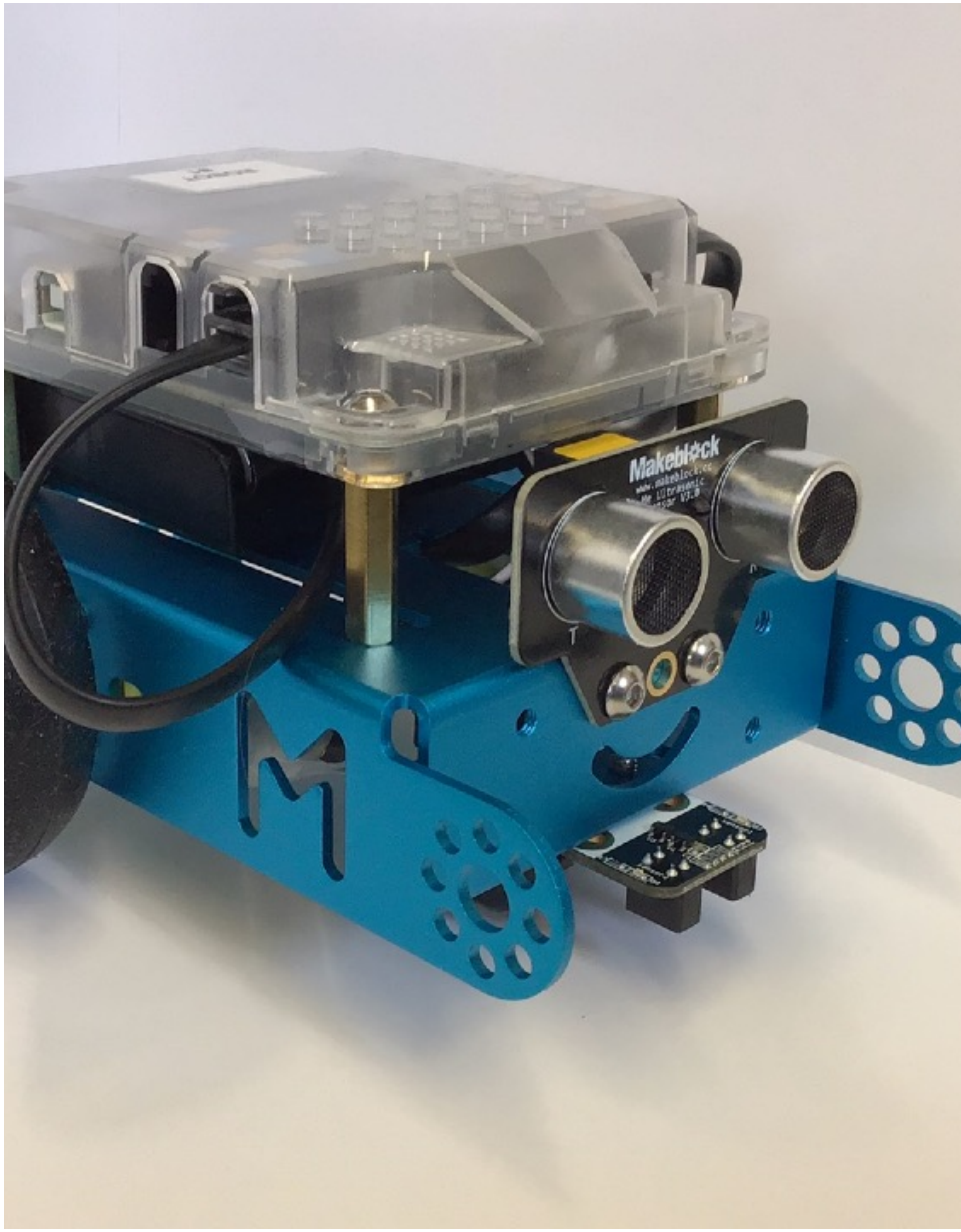
Starting Out with mBot

A. McKillop 2018

What is the mBot?

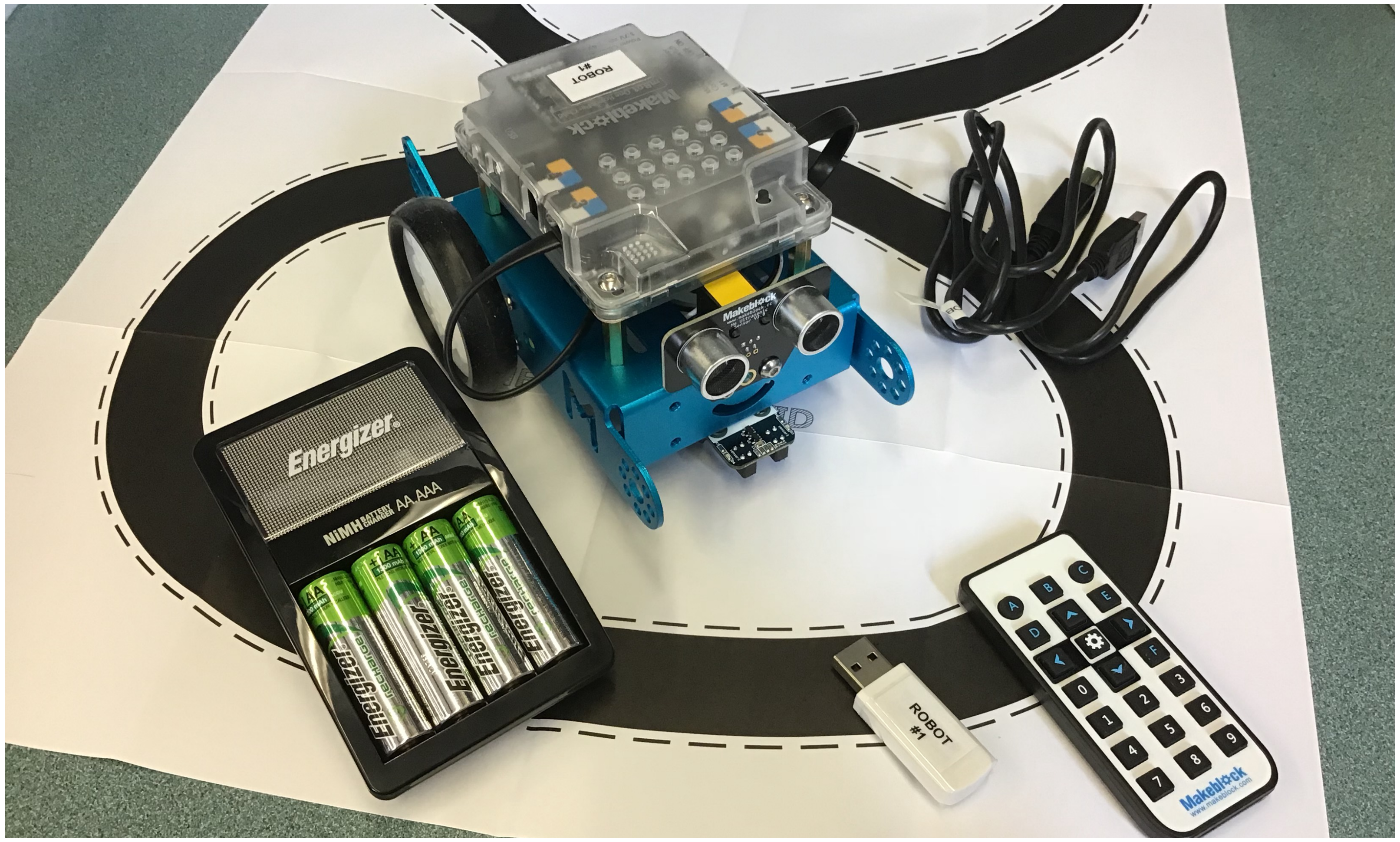
The mBot is a simple robot that students can use to learn basic programming skills. It can be controlled by a laptop, infrared remote, or by code written by students. This guide will walk you through the setup and use of the District Library Resource Centre mBot Kit.

Using mBot in your classroom



The DLRC mBot kit contains

- 15 mBots (each mBot comes with IR remote control, wireless dongle for laptop connection, figure 8 floor mat, and USB cable)
- 15 laptops with charge cables inside a laptop cart
- 15 sets of 4 rechargeable AA batteries
- 6 battery chargers


Each mBot in the DLRC kit comes these items

The kit is designed to be used by a maximum of 30 students, with each robot and laptop set being used by a pair of students.

Things to note:

- Before using the mBots with your class, check that the batteries are charged.
- The laptops are stored inside a laptop cart. Check to make sure that each laptop is plugged in to its charger, and that you plug in the cart itself!
- There shouldn't be any loose wires on the mBot. There is a picture of where everything should be plugged in on page 7
- The mBots are not meant to be disassembled! Please do not have your class take the robots apart! If this is what you would like them to do, please consider booking the Vex IQ kits from the DLRC instead!

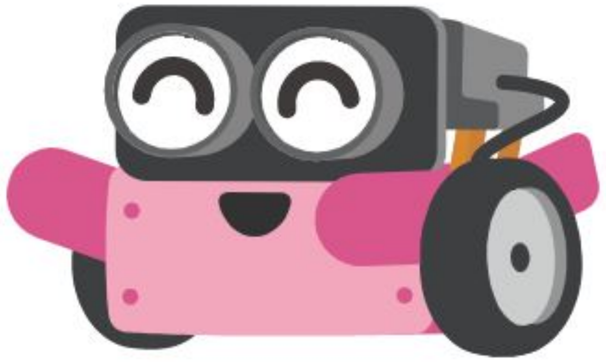
What can the mBot do?

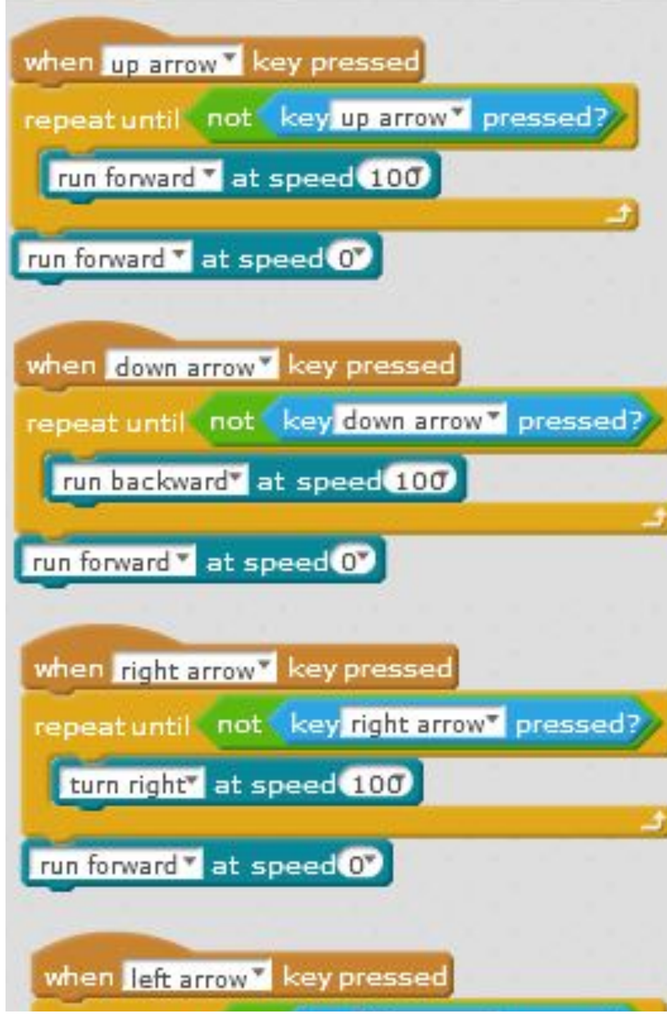
The mBot is a simple, yet powerful robot. It can be programmed by mBlock, a program based on Scratch, or using the programming language C. There are multiple sensors built in to the mBot that can be programmed, including a line following sensor, an ultrasonic sensor for measuring distances, and a light sensor.

The default mBot performs 3 different functions: using the IR remotes a controller, obstacle avoiding and line following.

The mBot is LEGO compatible, so students can make modifications to their mBot depending on the task at hand using standard or Technic pieces.

Whenever possible, work with the mBots on the floor so that they don't accidentally drive off your table!



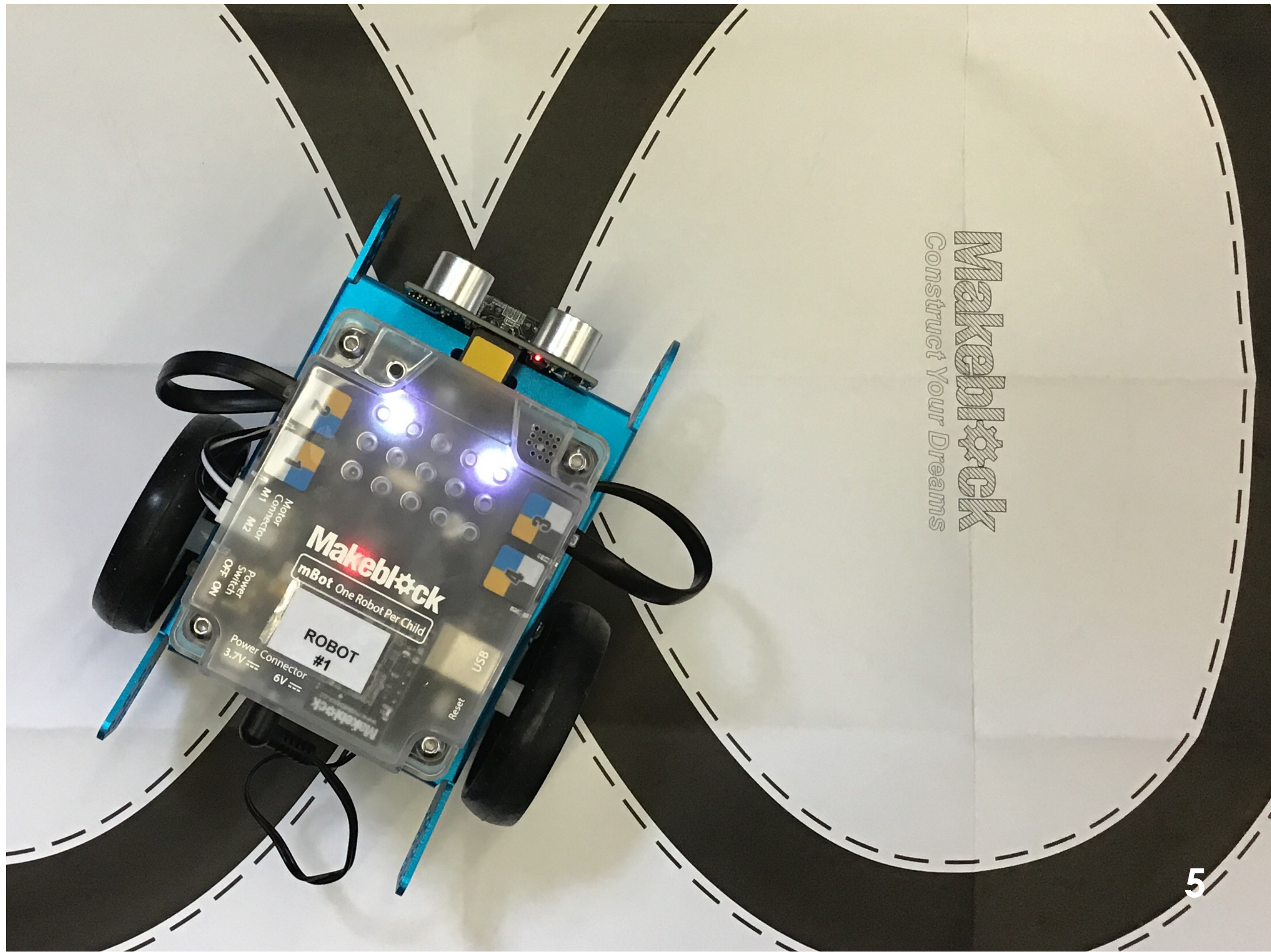


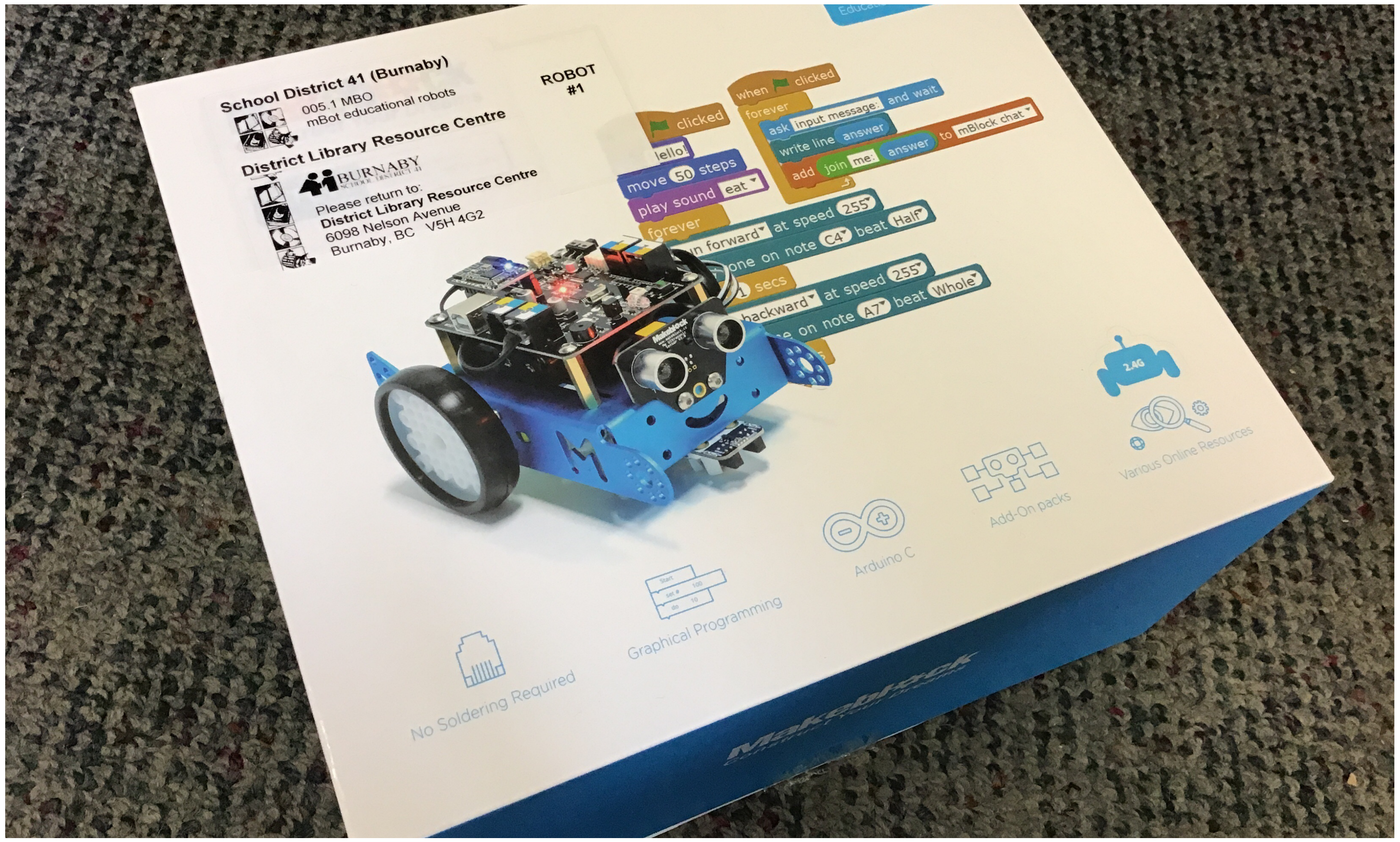
```
when up arrow key pressed
repeat until not key up arrow pressed?
    run forward at speed 100
run forward at speed 0

when down arrow key pressed
repeat until not key down arrow pressed?
    run backward at speed 100
run forward at speed 0

when right arrow key pressed
repeat until not key right arrow pressed?
    turn right at speed 100
run forward at speed 0

when left arrow key pressed
```





Each mBot is stored inside it's own box

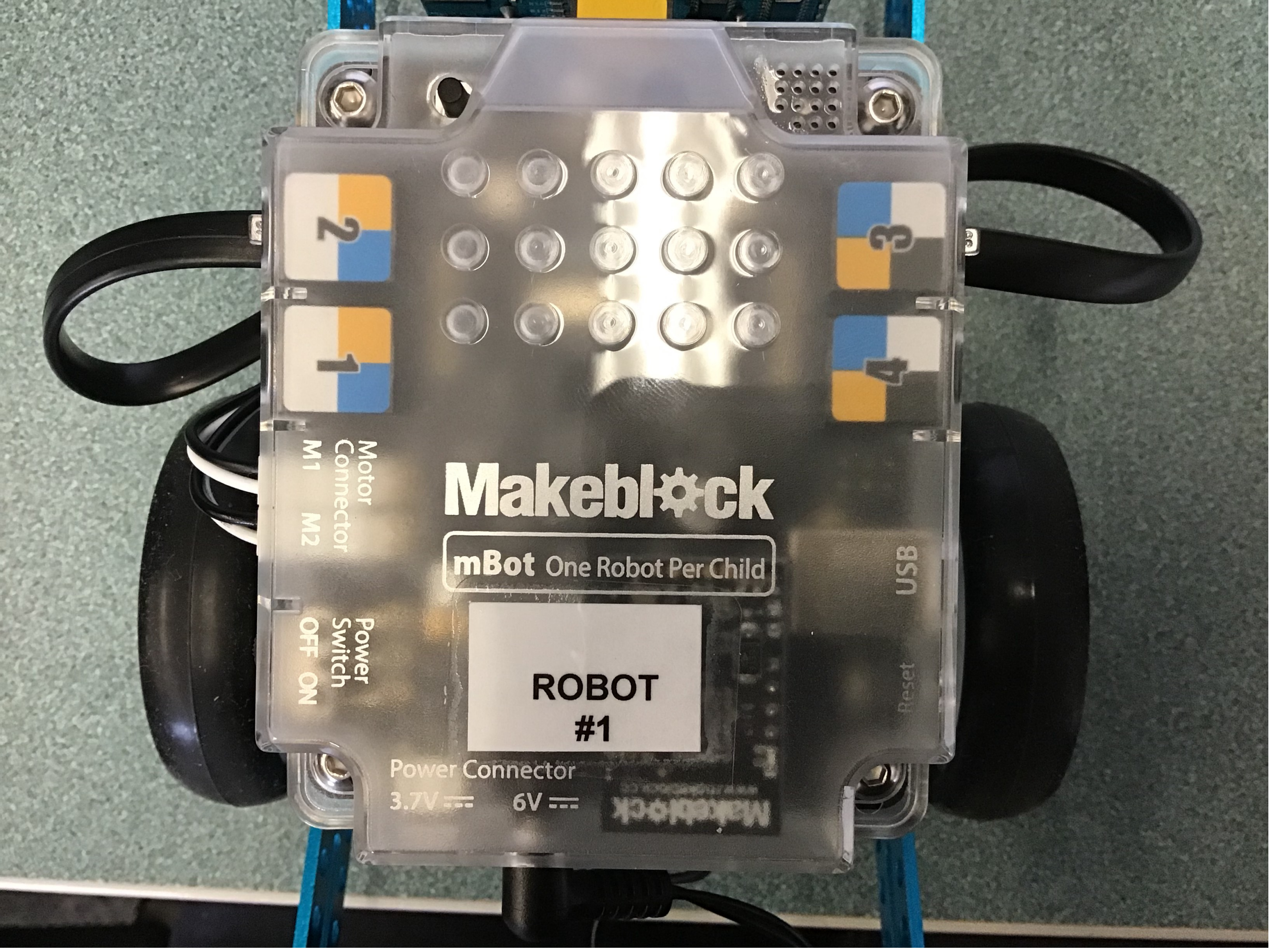
First Steps

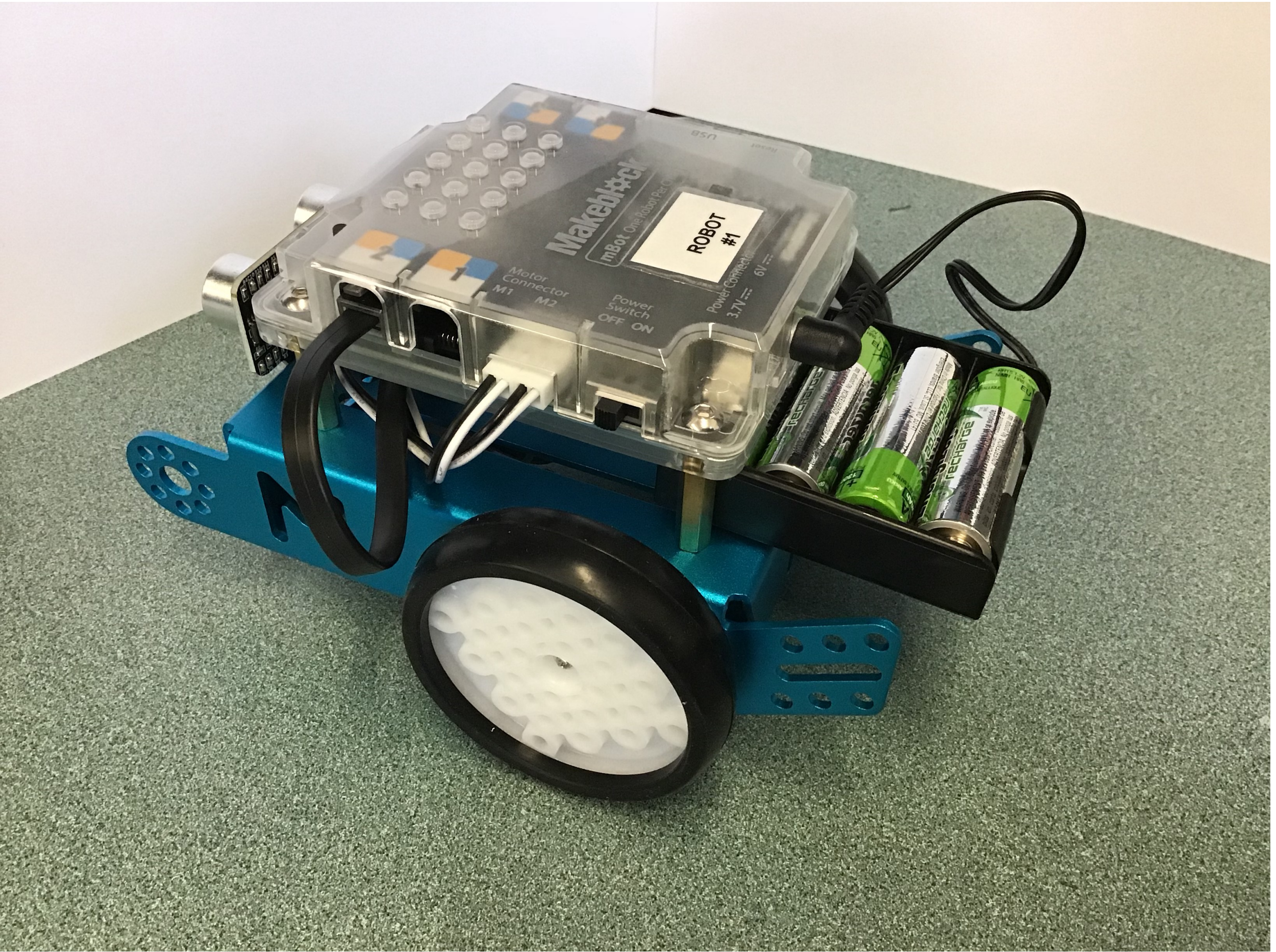
Once you've received the kit, it would be wise to make sure that the batteries for the robots and the laptops are charged up. The next step would be to put batteries into mBot. The battery pack is located underneath the "mCore" board. It is held in place by Velcro, and can be tricky to get out. (Photos on page 7)

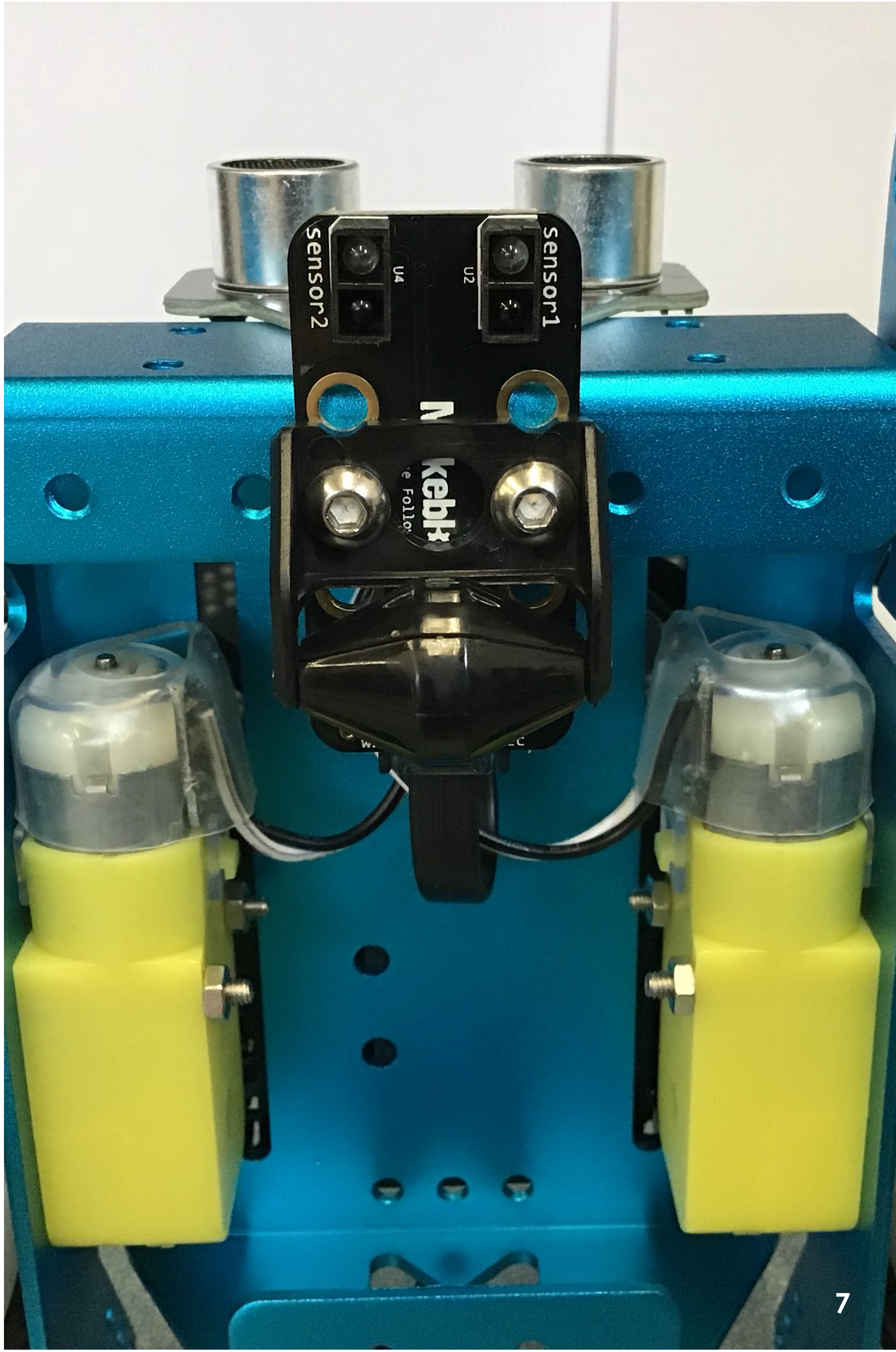
If you turn the switch on the mBot to the on position, it will begin to do whatever it was programmed to the last time it was used. To return it to it's default state, it needs to be connected to a laptop.

On the laptop, open mBlock. Next, plug the supplied USB cable into the computer and the mBot. I usually lay the mBot upside down on the table when it is tethered to the computer so that it doesn't accidentally drive away.

On the laptop, under the Connect menu, highlight Serial Port. In this menu, you will be able to select the connection between the laptop and the mBot (usually named COM9). If there is already a COM (short for COMmunication) listed, this will NOT be the one connecting to the mBot. Switch the mBot power switch to the ON position. This will open a connection between the two devices. The computer may take a minute or so to recognize the mBot. You may also need to reopen the menu. Select the COM port.


Makeblock
mBot One Robot Per Child
ROBOT #1
Motor Connector
M1 M2
Power Switch
OFF ON
Power Connector
3.7V 6V
USB
Reset


ROBOT #1


sensor2
sensor1
U4
U2

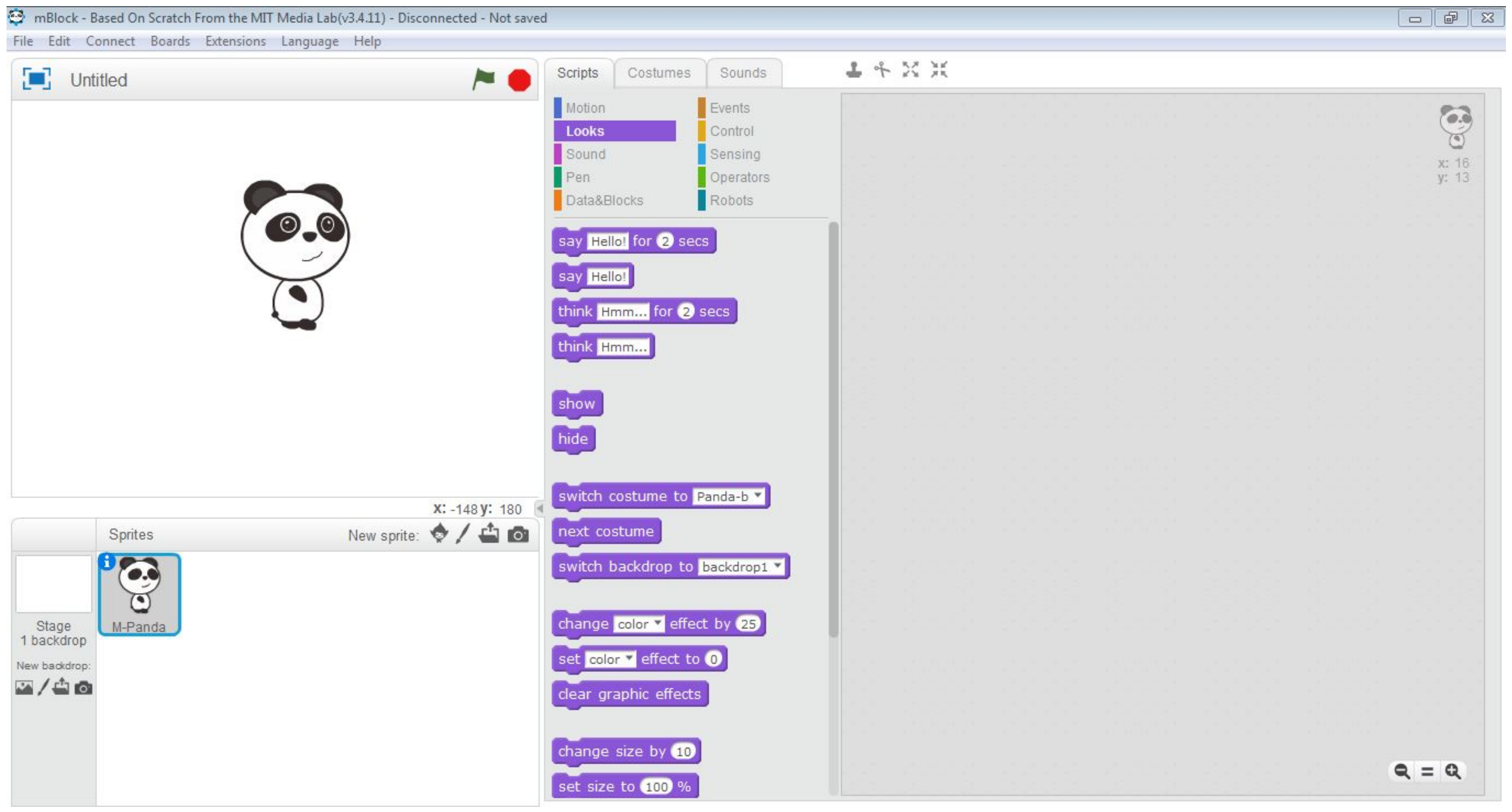
Using mBlock

The program we will be using to program the mBot is called mBlock. It is designed to use the same interface as Scratch, a block based programming language.

Once you have opened mBlock, it will bring up a blank program. The screen is divided into 4 main sections.

The “stage” is in the top left corner. This is where we can create interactive elements on the computer itself. These elements, called Sprites, can be used to interact with the mBot, or just be used on their own.

The “sprites” section is located below the stage. This is where you can keep track of all of the Sprites in your program. Each Sprite can be controlled by its own code. They can move, talk, make sounds and change their costumes.

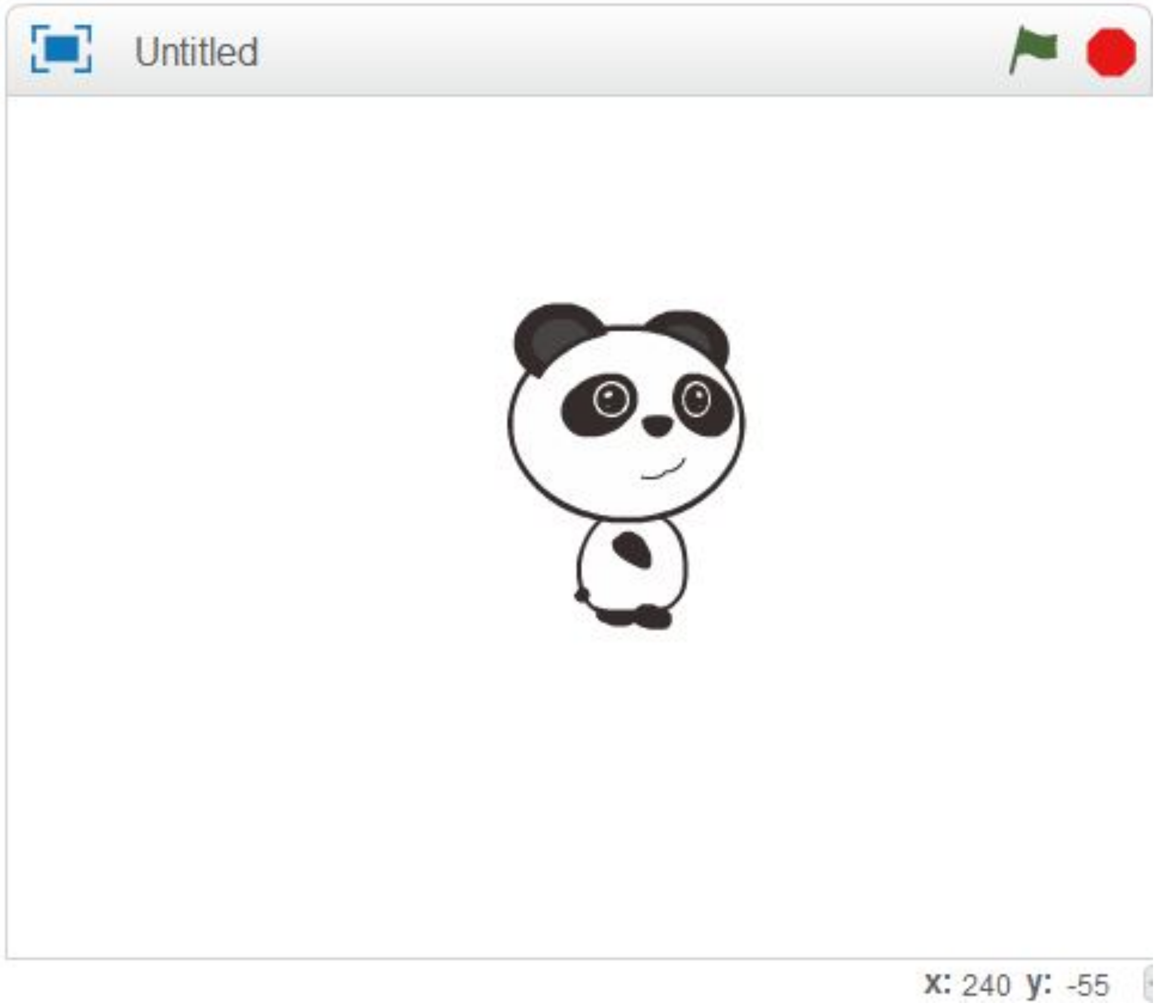

mBlock - Based On Scratch From the MIT Media Lab(v3.4.11) - Disconnected - Not saved
File Edit Connect Boards Extensions Language Help
Untitled
Scripts Costumes Sounds
Motion Looks Sound Pen Data&Blocks
Events Control Sensing Operators Robots
say Hello! for 2 secs
say Hello!
think Hmm... for 2 secs
think Hmm...
show
hide
switch costume to Panda-b
next costume
switch backdrop to backdrop1
change color effect by 25
set color effect to 0
clear graphic effects
change size by 10
set size to 100 %
x: -148 y: 180
Sprites
New sprite:
Stage
1 backdrop
New backdrop:
M-Panda
x: 16
y: 13

The home screen of mBlock

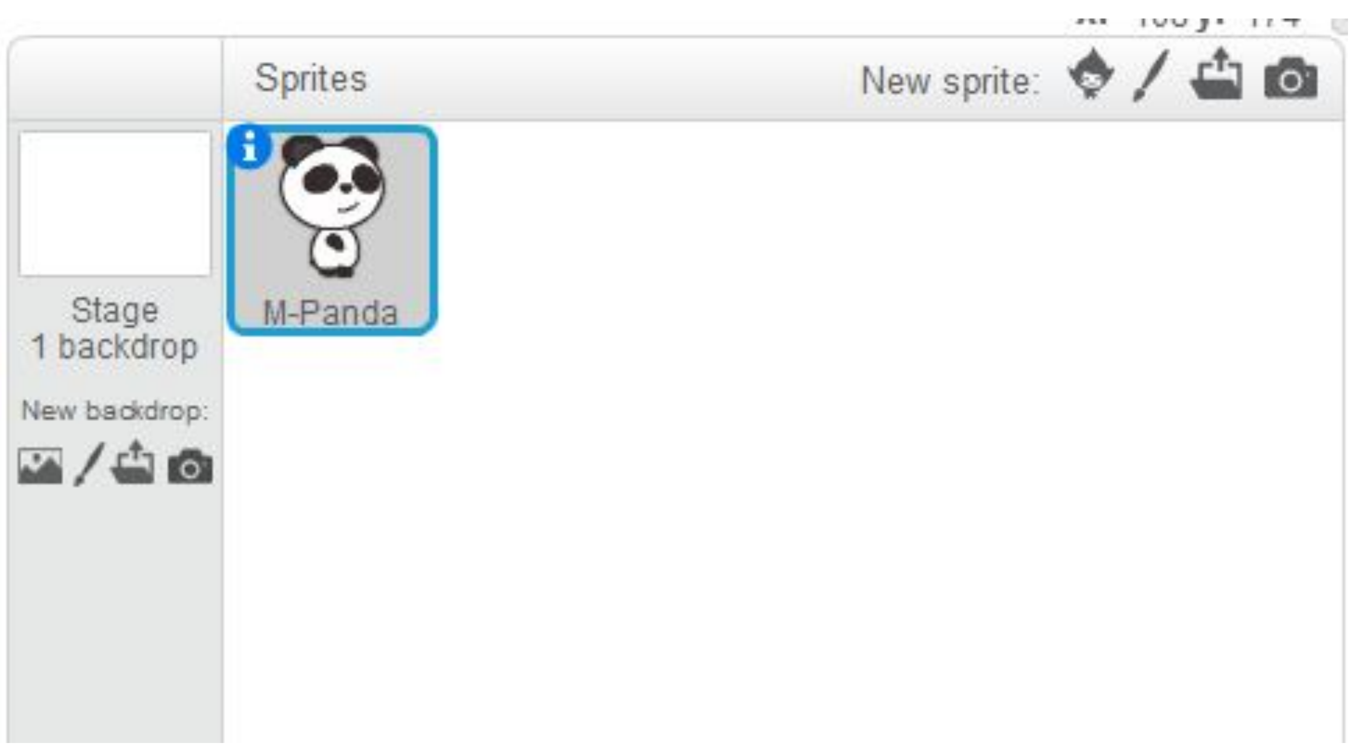
The “toolbox” is the middle column of the screen. This area holds all of the programming blocks. The blocks are sorted into sections to help you find what you are looking for more quickly.

The entire right hand side of the screen is taken up by the “workspace”. This area is where you will drag and drop the programming blocks. It may look big, but it can fill up quickly! You are able to zoom in and out to give yourself a bit more space to work. Each Sprite gets it’s own workspace, so make sure if you are working with more than one Sprite, you select the one you want to program in the Sprites menu!

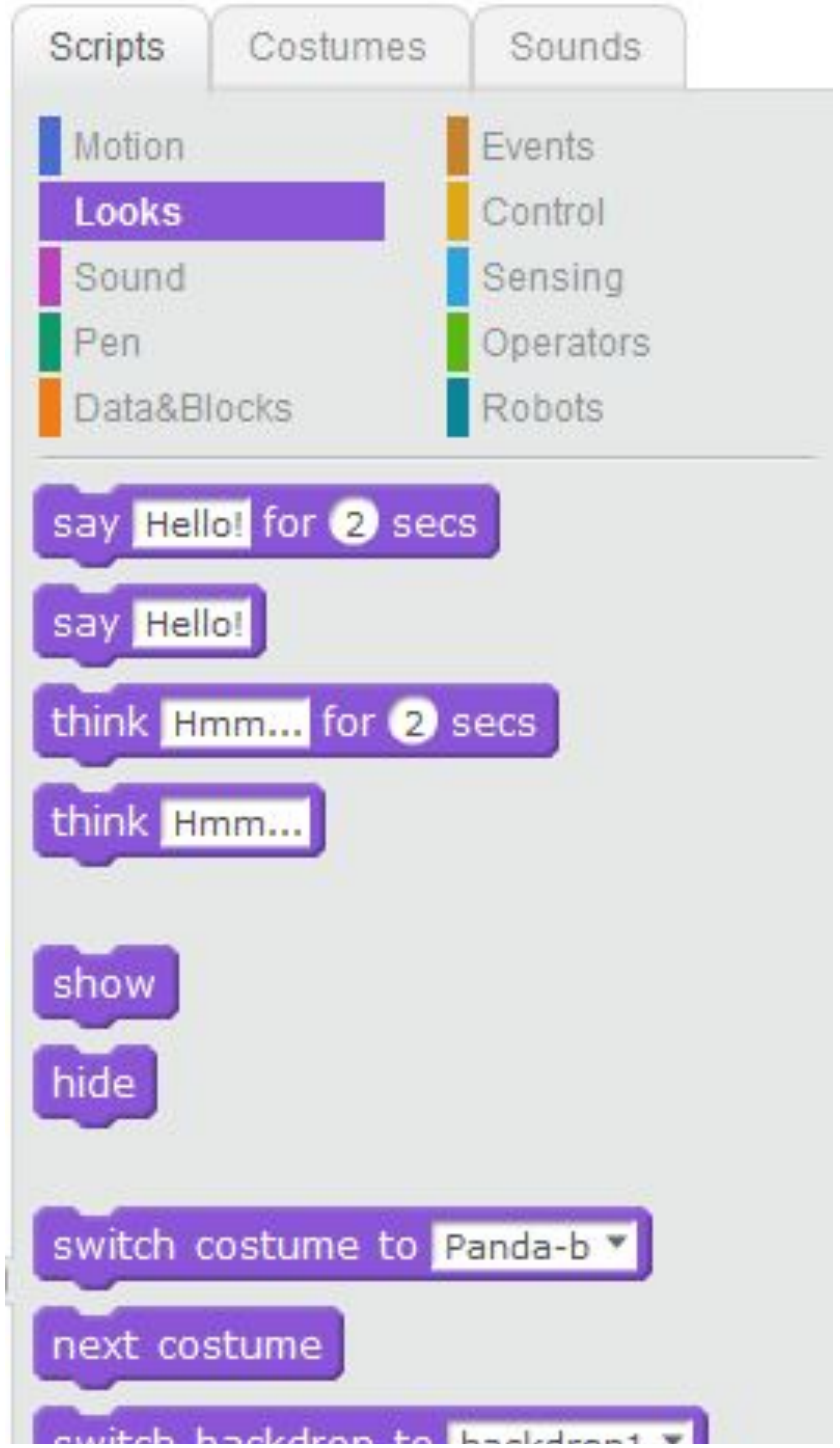
One of the most important things to



The Stage



The Sprites menu



The Toolbox

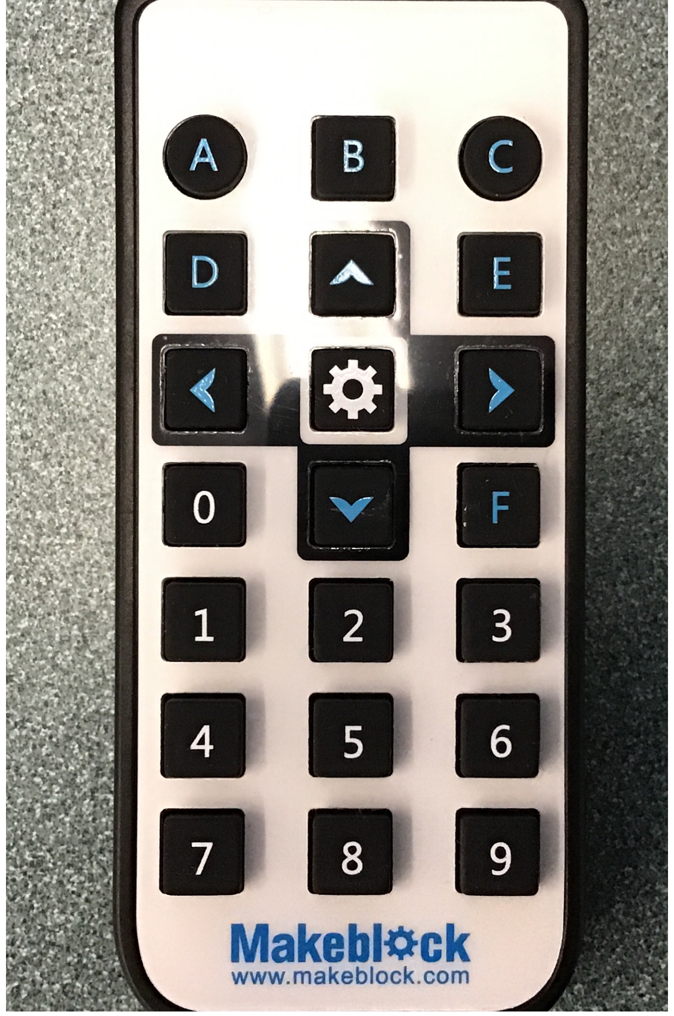
Default mBot Functions

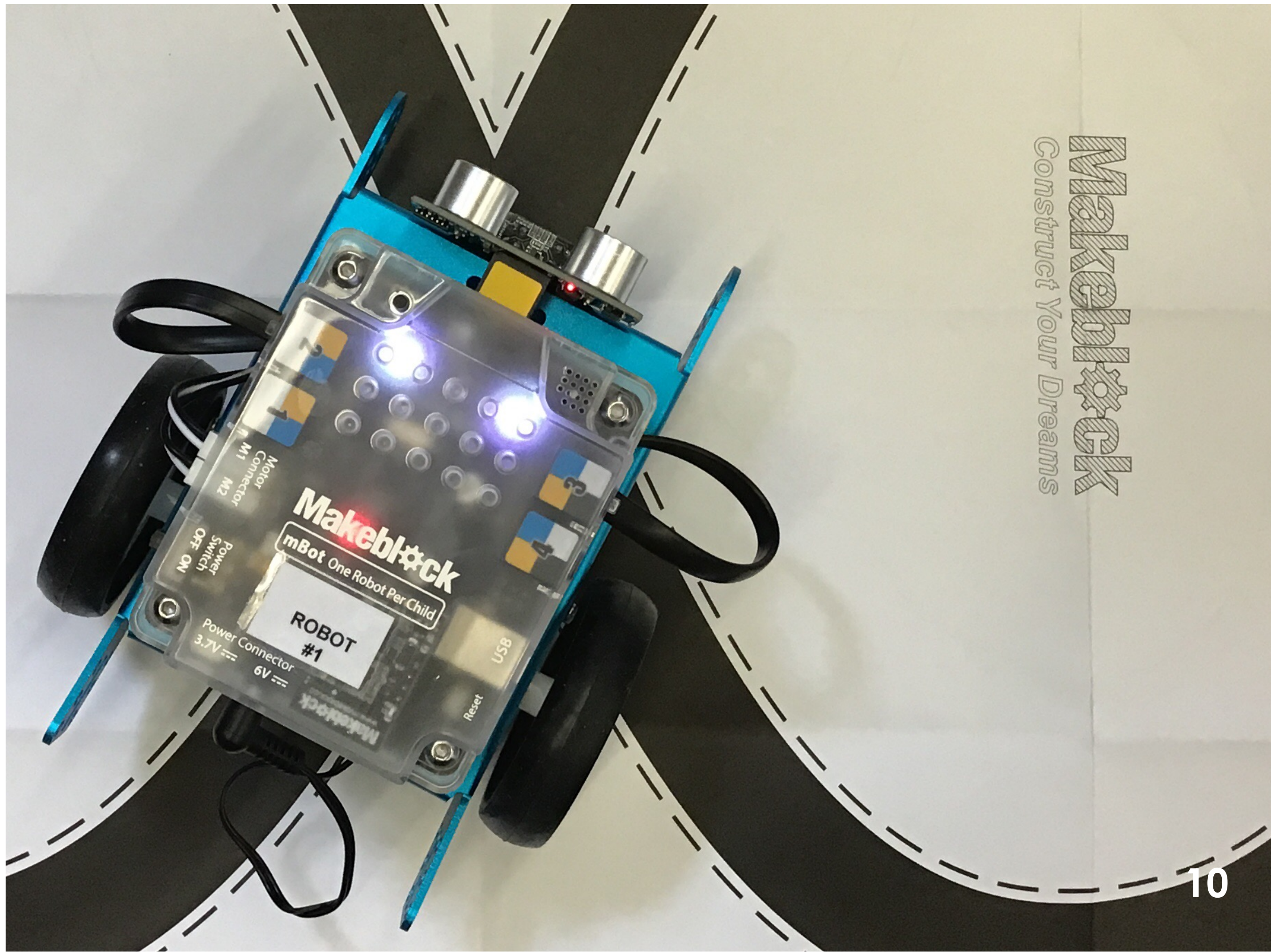
There are three default mBot programs that can be selected by pressing the A, B, or C buttons on the remote control, or by pressing the onboard button on the mBot itself.

The first program is manual remote control. The arrow buttons control its direction and the numbers control the speed (1 being the slowest, 9 the fastest).

The second is a range detector. The mBot will drive forwards until it detects an obstacle ahead. When this happens, it will stop, turn and continue moving in the new direction.

The third program is line detection. Place the mBot on a figure 8 mat, and it will follow the black line continuously.



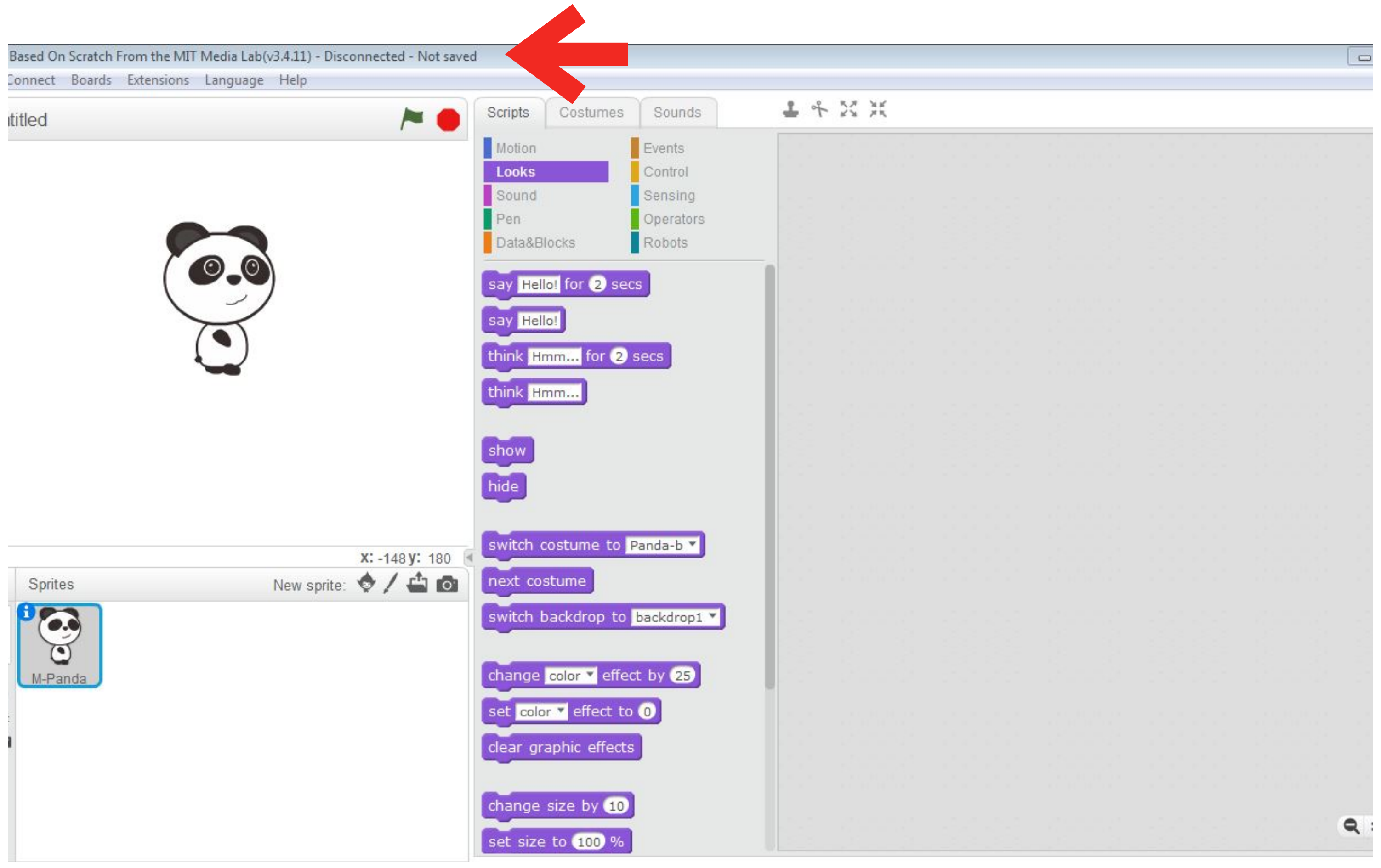


Connecting Wirelessly to mBot

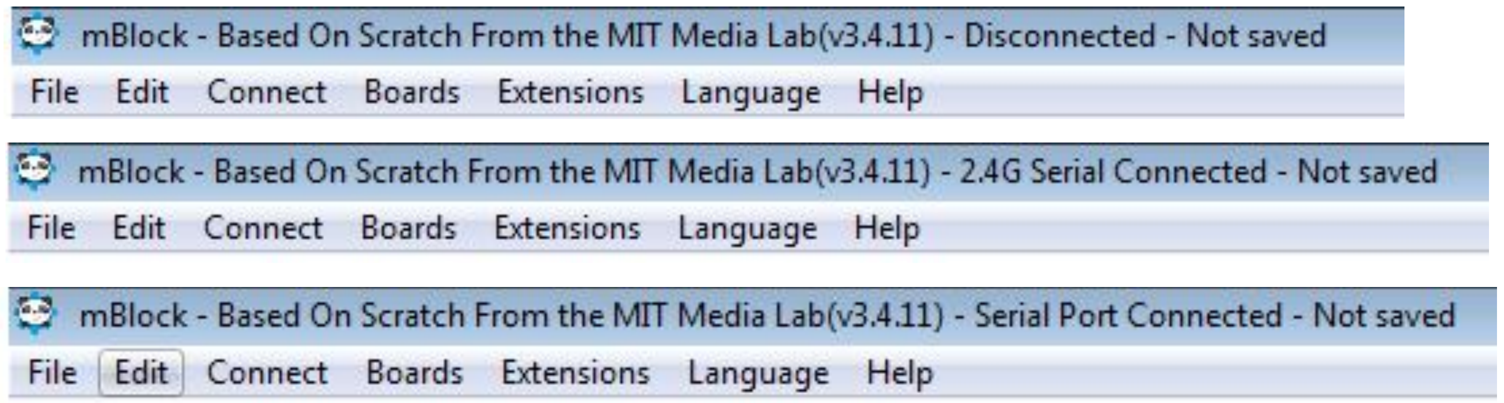
Once you are familiar with the default programs, you will probably want to start creating some programs of your own.

You could continue using the USB cable to connect to the mBot, but it can be connected to wirelessly as well. To do this, turn off the mBot. Save your work in mBlock and close the program. Plug the wireless dongle into the laptop and unplug the USB cable. Turn the mBot back on and open mBlock. Under the Connect menu, select 2.4G Serial. Click the "Connect" option.

With the 2.4G connection, you are able to modify and test programs in without having to reconnect the mBot to the laptop. This is especially handy if you are writing programs where keys on the laptop are used to control the mBot.


Based On Scratch From the MIT Media Lab(v3.4.11) - Disconnected - Not saved

The top bar of the mBlock window will show the type of connection used with the mBot and whether the program has been saved.


mBlock - Based On Scratch From the MIT Media Lab(v3.4.11) - Disconnected - Not saved
File Edit Connect Boards Extensions Language Help

mBlock - Based On Scratch From the MIT Media Lab(v3.4.11) - 2.4G Serial Connected - Not saved
File Edit Connect Boards Extensions Language Help

mBlock - Based On Scratch From the MIT Media Lab(v3.4.11) - Serial Port Connected - Not saved
File Edit Connect Boards Extensions Language Help

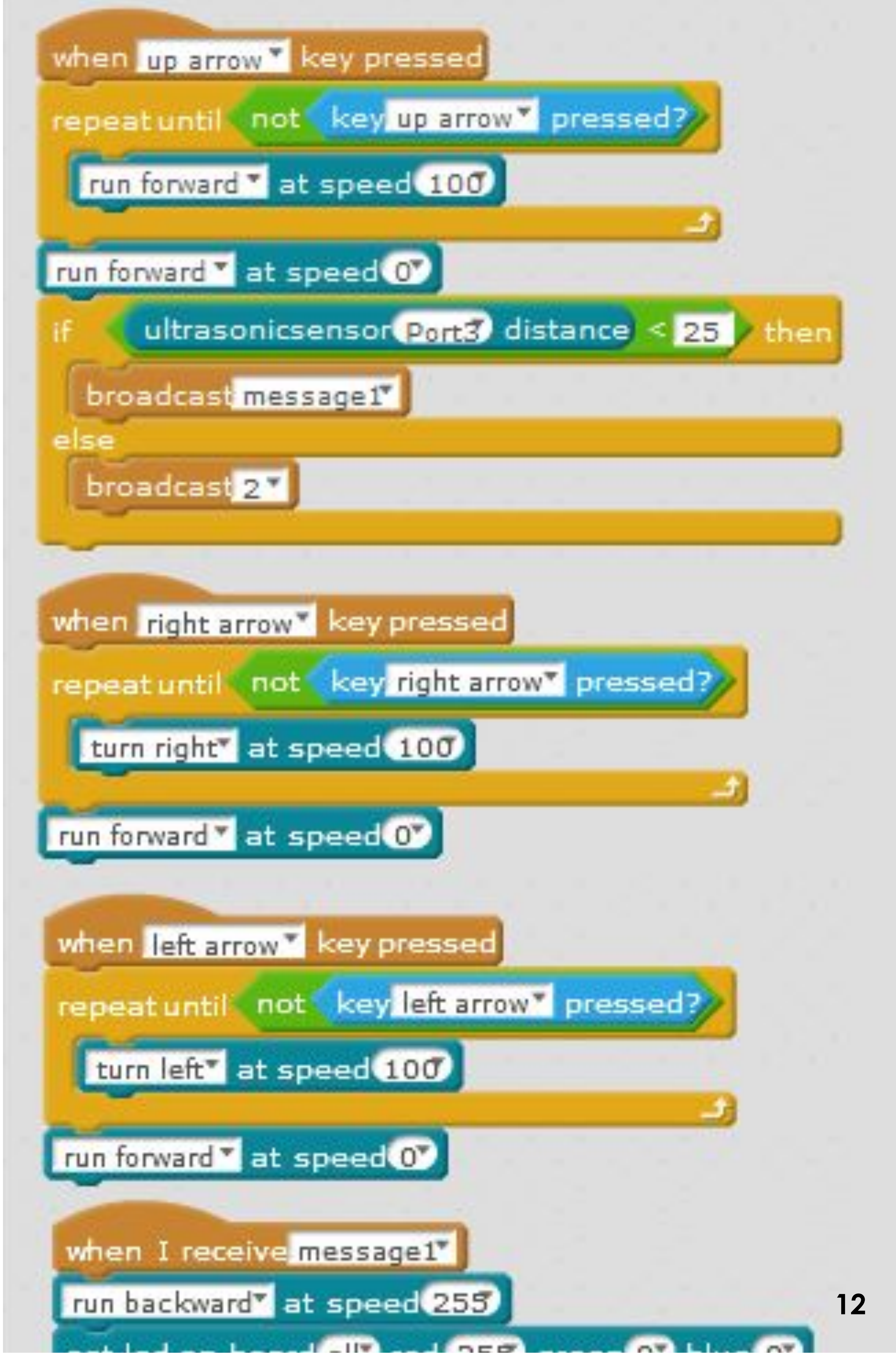
Creating Your Own Programs

Controlling mBot with keyboard commands

Driving mBot using a virtual controller

Making mBot independent

Using mBot's sensors to avoid obstacles & follow lines



```
when up arrow key pressed
repeat until not key up arrow pressed?
    run forward at speed 100
run forward at speed 0
if ultrasonicsensor Port3 distance < 25 then
    broadcast message1
else
    broadcast 2

when right arrow key pressed
repeat until not key right arrow pressed?
    turn right at speed 100
run forward at speed 0

when left arrow key pressed
repeat until not key left arrow pressed?
    turn left at speed 100
run forward at speed 0

when I receive message1
run backward at speed 255
```

Controlling mBot With a Keyboard

A good place to start out with programming the mBot is to control it using the arrow keys of a laptop. If you haven't been using the 2.4G wireless connection yet, now would be a good time to try it out (so you don't have to chase the mBot around with the laptop plugged into it).

It is important to note that mBots (and anything else you can program) are not very smart. Everything you want it to do must be broken down into small steps. For example, if you want the mBot to move, it needs to be told when to start moving, which direction you want it to go, how fast you want it to move, and when you want it to stop!

Our program will use the arrow keys, but feel free to experiment with using other keys!

Starting with a blank mBlock screen, connect your mBot. Grab an event "Hat" that says "When _______ key pressed". A "hat" is what Scratch (and mBlock) call the blocks that have curvy tops. These blocks are used to start out our programs. In the block, you are able to select which key you would like to press. I generally start by using the Up Arrow, but you could use whichever key you like.

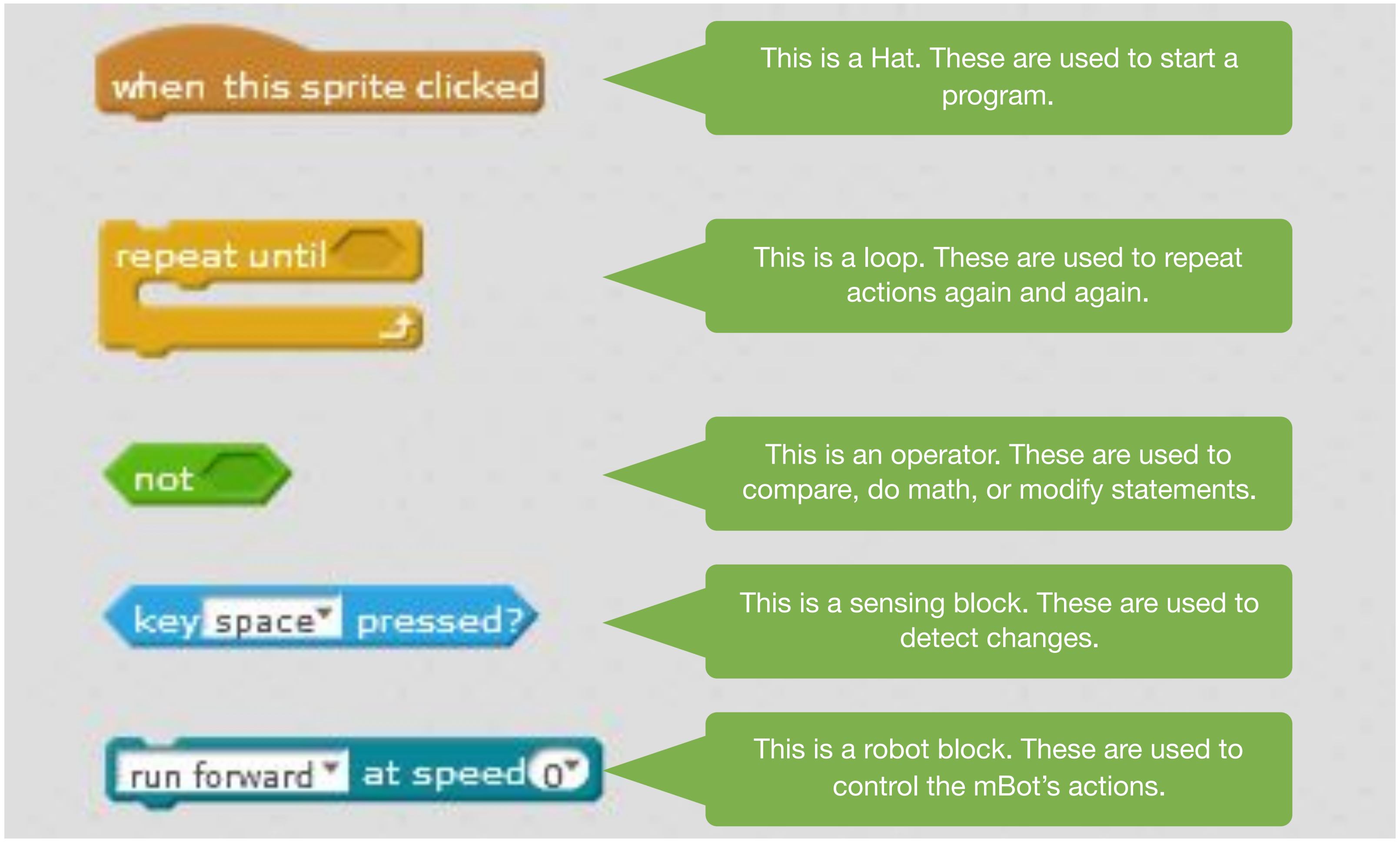
Drag the hat into your workspace. The mBot will respond to you pressing the Up Arrow by doing whatever you connect to the bottom of the hat.

In the Control section, drag out a "Repeat until" loop. Anything placed inside of this block will happen repeatedly until a specific condition is met. We want the mBot to move forward slowly (more on speed later) until the Up Arrow is released.

You will need 4 blocks to accomplish this. In the Operators section, we need a "Not" block, in the Sensing section, we need a "Key _____ pressed?" block, and in the Robots section we need a "run forward at speed ___" block.

Start by nesting the "Key (Up Arrow) Pressed" block inside of the Not block. Next, place these two blocks into the opening of the Repeat Until loop. Now, the mBot will repeatedly perform the task you give it inside the loop until the Up Arrow is not pressed anymore.

The task we want the mBot to perform is to "run forward". To set the speed, you can enter a number between -255 and 255. The bigger the number, the faster mBot will drive. I recommend starting out with a speed


when this sprite clicked
This is a Hat. These are used to start a program.
repeat until
This is a loop. These are used to repeat actions again and again.
not
This is an operator. These are used to compare, do math, or modify statements.
key space pressed?
This is a sensing block. These are used to detect changes.
run forward at speed 0
This is a robot block. These are used to control the mBot's actions.

Different types of programming blocks

of around 100 until you have made sure the program works successfully.

If you were to press the Up Arrow at this time, the mBot will begin moving forward, but it does not stop when you release the Up key. This is because we haven't told the mBot what to do once we've release the Up key, so it continues to do the only thing we've told it how to do! To get it to stop moving forward, drag a second "run forward" block and connect it to the bottom of the "Repeat Until" block. Set the speed on this one to 0. Now, the mBot knows we want it to stop moving once we release the key.

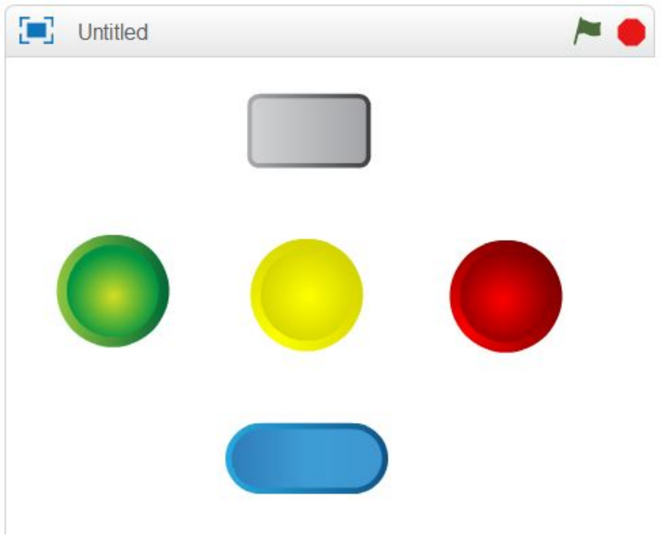
This block of code can be repeated 3 more times; replacing Up with Left, Right and Back (Down).

Creating a Virtual Controller

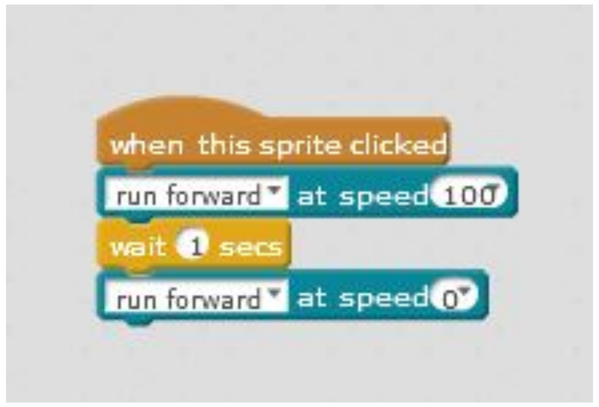
In addition to being controlled by the keyboard, the mBot can be controlled using a virtual keyboard created in mBlock. By using Sprites we will create a program that will allow us to control the mBot's motion, it's built in LEDs (Light Emitting Diodes) and it's buzzer.

To start, open a new mBlock window and connect your mBot. You could use any Sprite you like, but I like to use ones that look more like buttons. To get the hang of this, I will outline the steps to get a button to move the mBot forward, one for backwards, one to "dance", one for the lights and one for the buzzer. You will need a total of 5 Sprites (it's best if they all look different).

Once you have spaced out your buttons, highlight the one you'd like to control the forward motion in the Sprites menu. From the Events section, grab the Hat called "when this sprite is clicked". Next you need to tell the mBot what you want it to do once that Sprite is clicked. Using two "Run forward" blocks and a "Wait" block, we can tell the mBot to move forward at a set speed for 1 second before stopping.

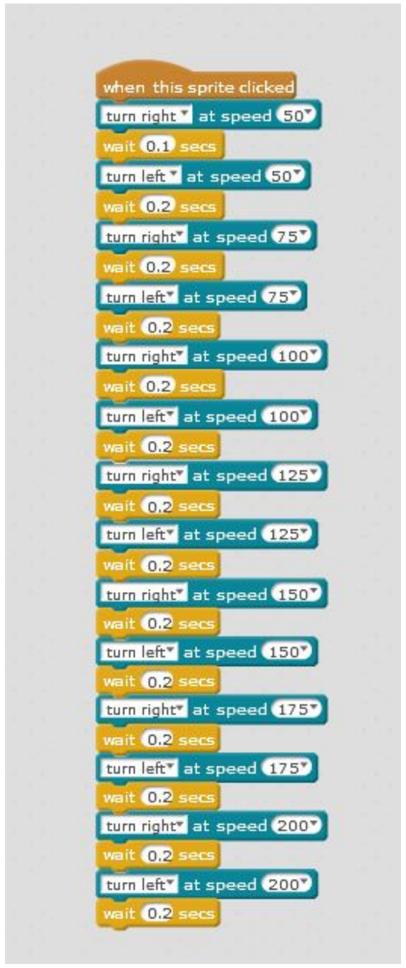


Sample Virtual Controller



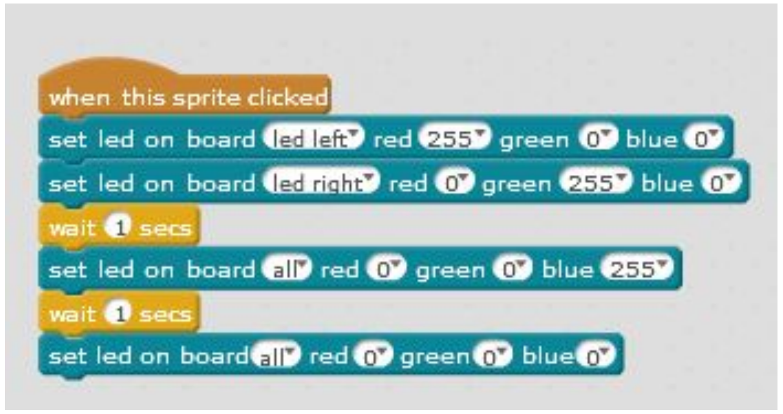
```
when this sprite clicked
run forward ▾ at speed 100
wait 1 secs
run forward ▾ at speed 0 ▾
```

Code for the Forward Button



```
when this sprite clicked
turn right ▾ at speed 50 ▾
wait 0.1 secs
turn left ▾ at speed 50 ▾
wait 0.2 secs
turn right ▾ at speed 75 ▾
wait 0.2 secs
turn left ▾ at speed 75 ▾
wait 0.2 secs
turn right ▾ at speed 100 ▾
wait 0.2 secs
turn left ▾ at speed 100 ▾
wait 0.2 secs
turn right ▾ at speed 125 ▾
wait 0.2 secs
turn left ▾ at speed 125 ▾
wait 0.2 secs
turn right ▾ at speed 150 ▾
wait 0.2 secs
turn left ▾ at speed 150 ▾
wait 0.2 secs
turn right ▾ at speed 175 ▾
wait 0.2 secs
turn left ▾ at speed 175 ▾
wait 0.2 secs
turn right ▾ at speed 200 ▾
wait 0.2 secs
turn left ▾ at speed 200 ▾
wait 0.2 secs
```

Code for Dancing mBot



```
when this sprite clicked
set led on board (led left) red (255) green (0) blue (0)
set led on board (led right) red (0) green (255) blue (0)
wait (1) secs
set led on board (all) red (0) green (0) blue (255)
wait (1) secs
set led on board (all) red (0) green (0) blue (0)
```

Code for the Lights

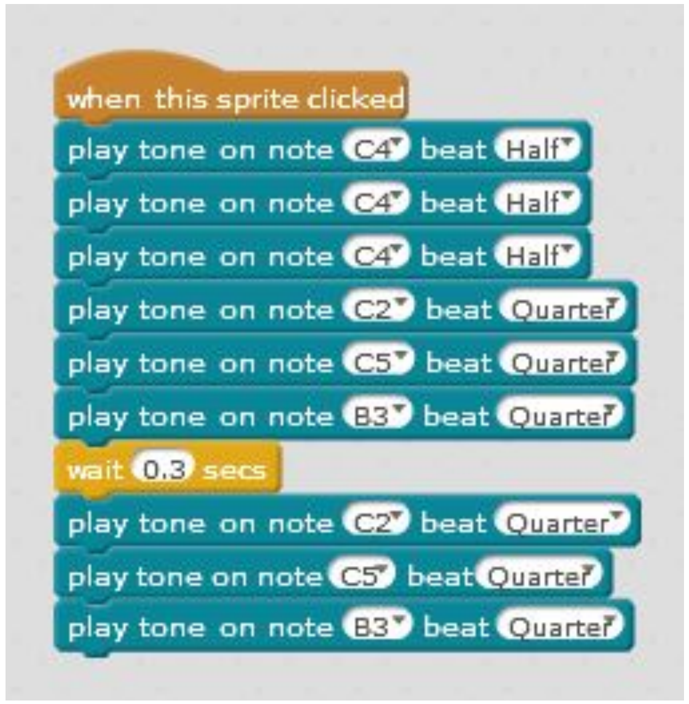
The same procedure can be applied to other directions as well. To create a “Dance” button, you can give the mBot a sequence of actions, which it will complete in the order you list them. Keep in mind that “Dancing” is used loosely! You could select a piece of music and actually choreograph the mBot’s movements to the beat, but my sample is simply making the mBot move in a pattern to the left and right.

To control the lights onboard the mBot, drag a “set led on board (all) red (0) green (0) blue(0)” block from the Robots section and connect it to a “sprite clicked” hat for your Lights Button. The Lights on the mBot can create a huge number of colours by mixing Red, Green and Blue. Each colour can be set to a value of 0 - 255. Adjusting these values will change the colour that the LED produces. If you set Red to 255, and the other two to 0, the LED will be bright red. The same will be true if you set the Blue to 255 or the Green to 255.

In addition to controlling the Red, Blue and Green values, you are able to control the left and right LEDs independently. The sample on the previous page simply cycles the LEDs through each colour for one second at a time.

To use the button to control the buzzer sounds, we need to tell the mBot which note to play, along with the duration of the note. To begin, select the Sprite you intend to use as your Song button. Drag a “when this sprite is clicked” hat into your workspace. You can experiment with the different notes and durations, or you can try to recreate your favourite tune! The sample is a quick and dirty rendition of the Imperial March.

These activities can be extended in a variety of ways; creating a buttons for all directions, creating buttons to perform pre-programmed actions (ie Figure of 8, circles, etc). Sprites could also be used to create animations on the computer (like you would do with Scratch) which are able to interact with the mBot. Possibilities are only limited by imagination!



```
when this sprite clicked
play tone on note (C4) beat (Half)
play tone on note (C4) beat (Half)
play tone on note (C4) beat (Half)
play tone on note (C2) beat (Quarter)
play tone on note (C5) beat (Quarter)
play tone on note (B3) beat (Quarter)
wait (0.3) secs
play tone on note (C2) beat (Quarter)
play tone on note (C5) beat (Quarter)
play tone on note (B3) beat (Quarter)
```

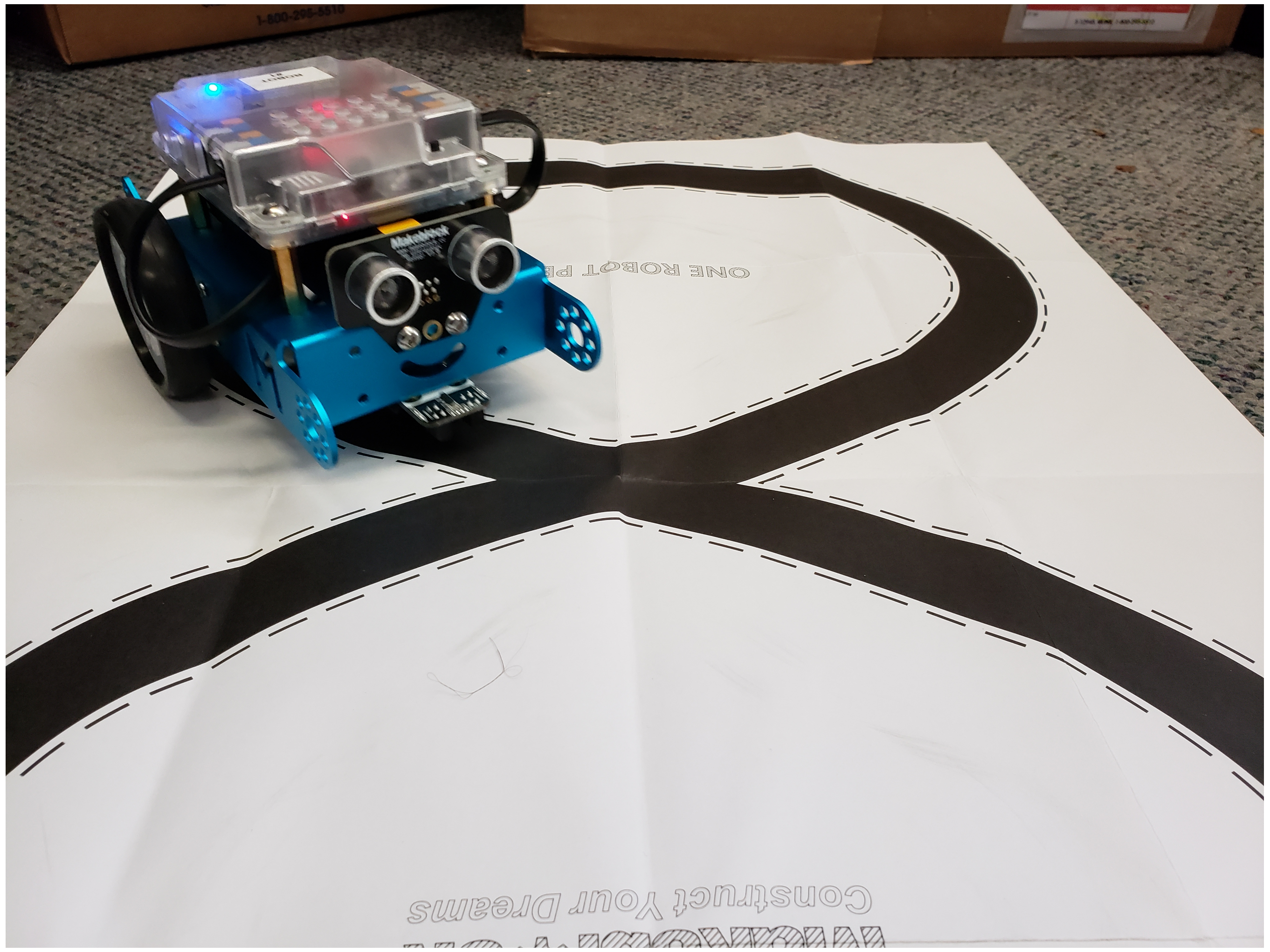
Code for the Wake Up Song Button

Independent mBot

While it is fun to control the mBot with buttons, remotes and keyboards, an important part of robotics is creating programs that make the device operate independently. This is called automation, and a robot that is programmed to do this is considered to be autonomous.

To make your mBot behave autonomously, we need to teach it to make decisions. It helps to have a specific task for the mBot to accomplish, along with some backup plans in case it isn't able to do it's task.

A good way to make our mBot behave autonomously is it use it's built in sensors to help it make decisions. We will be using the ultrasonic, line following and light sensors to keep our mBots out of danger.



A closeup of the mBot's "face". The eyes are the Ultrasonic sensor.

In order to use the mBot sensors, we need to know a bit about how they work, and the kind of information they can collect.

An ultrasonic sensor uses sound waves bouncing off objects to determine the distance between itself and things around it. We aren't able to hear the sound it uses to do this because the human ear can't pick up that frequency. It works a lot like echo location used by bats. The sensor sends this information to the mBot or computer in the form of a number. Conveniently, the number is the distance from the sensor to the nearest object, given in centimetres.

The maximum distance the sensor can detect is 400cm, and the minimum distance is 1cm.

The line following sensor works by using ultraviolet light. The sensor sends out ultraviolet light and tries to detect any light that bounces back to it. Depending on whether the light is reflected back, the sensor will send the mBot or computer a 0 (for when the light isn't reflected) or a 1 (for when the light is reflected). Ultraviolet light is reflected best by white, and is absorbed by black.

There are two of these units built into our mBot sensor, one for the left side and one for the right side. This is interpreted by our program and can tell the mBot which way to turn in order to follow the line.

The light sensor works by detecting the brightness of light surrounding the sensor. This is sent to themBot or computer in the form of a number between 0 and 1023, with 0 being no light at all.

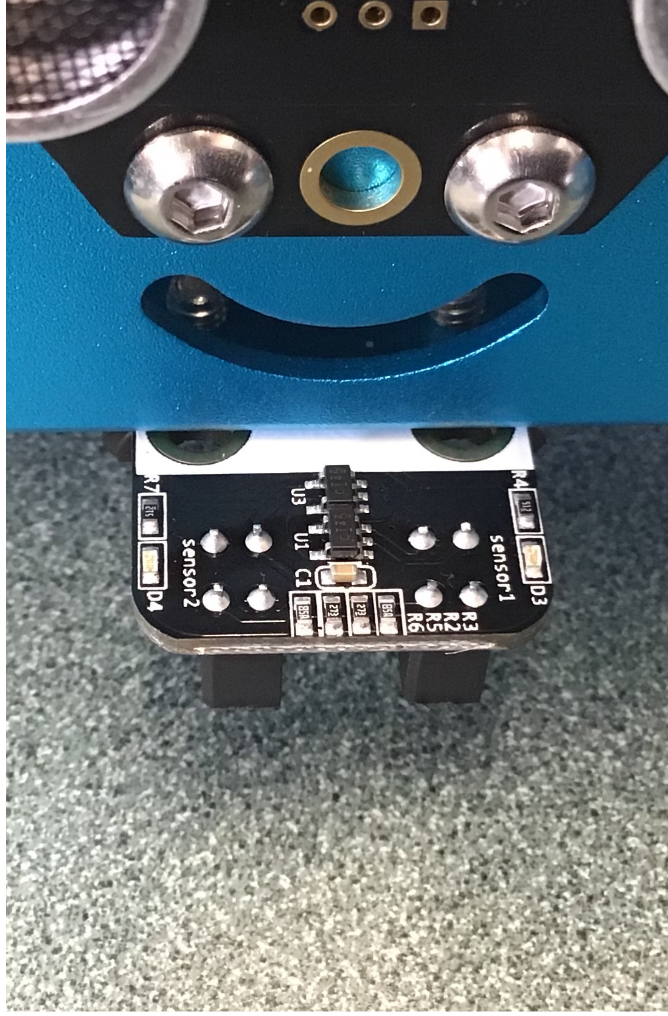
Using Sensors

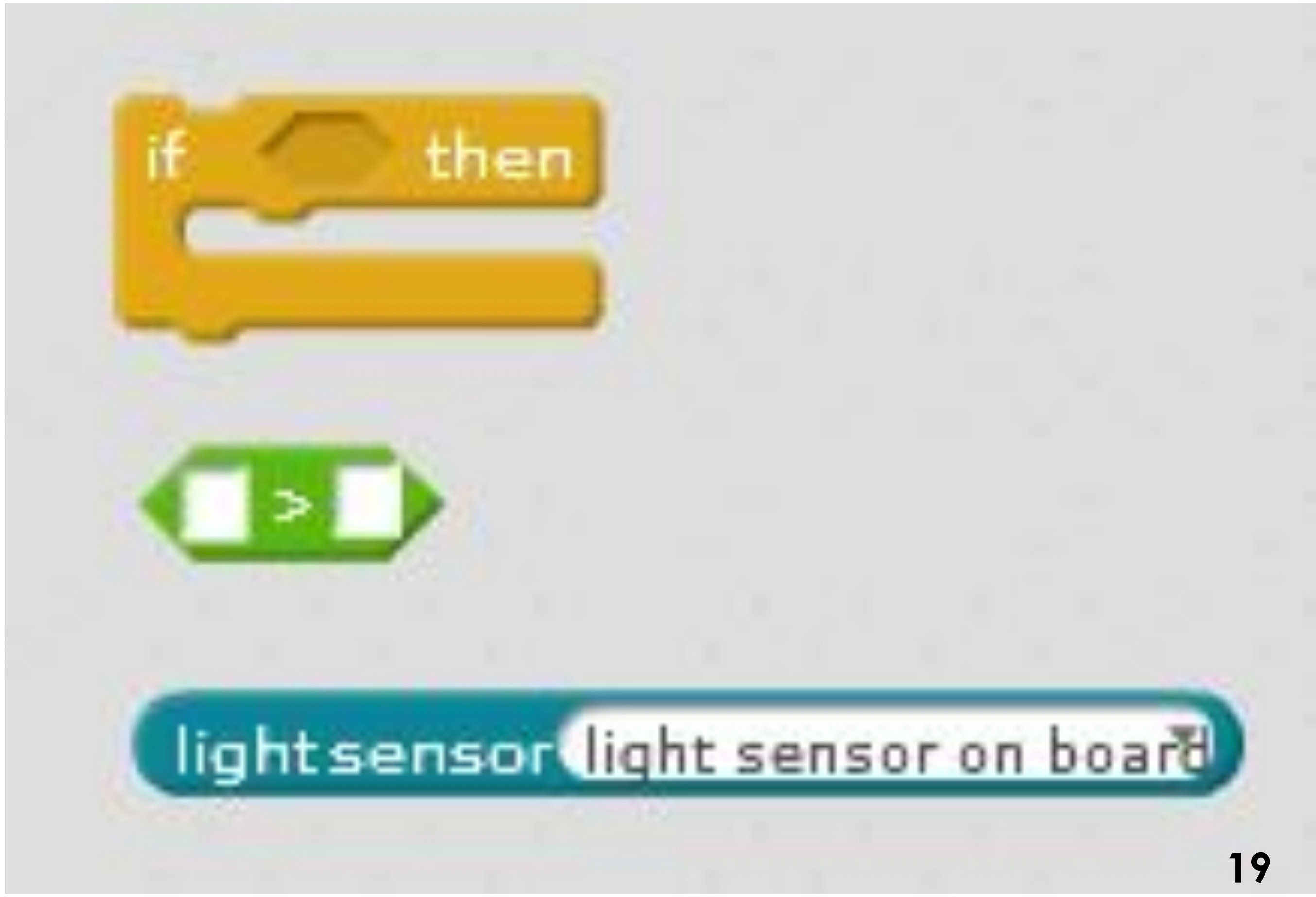
We will start with using the light sensor to tell the mBot when it is time to wake up. Our goal is to program the mBot to wake up, sing it's morning song and stretch it's wheels once it is bright enough.

To begin with, we need to break down these tasks into terms our mBot can understand. We want our mBot to be asleep if it is dark out. If it is bright, we want out mBot to wake up, then sing it's wake up song, and then stretch.

Start by dragging an "if _____ then" loop into your workspace. In the blank space between if and then, we need to tell the mBot what conditions to look for. The mBot is supposed to be checking the light sensor value, so we need the "light sensor" block from the Robot section. We want the mBot to decide if it is bright enough to wake up, so we need to tell it what "bright" is.

The light sensor gives a number between 0 and 1023. For this example, we will use the numbers 500 and up to define "bright". To program this, we need a ">" block from the Operators section. To the left of the > symbol, put the "light sensor" block. To the right, type 500. This whole thing can be put between the if and then in our loop.





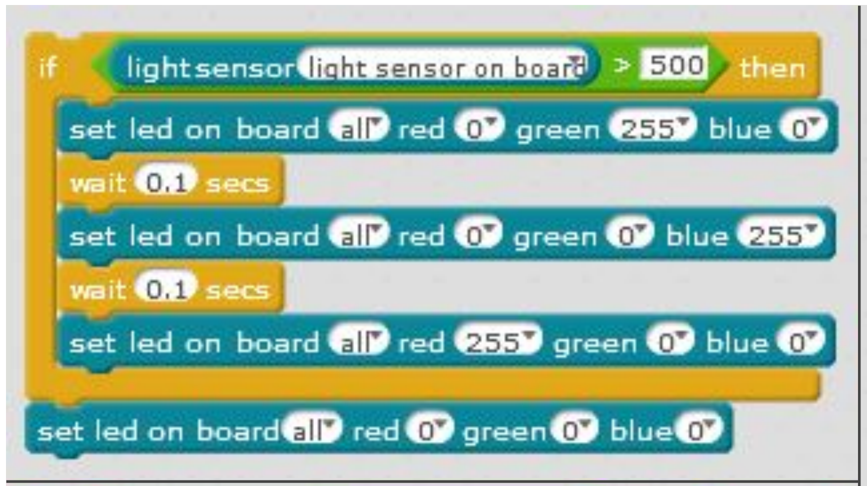
Light Sensor

Finally, we need to tell mBot what to do when it gets a light value over 500. This will be placed in the loop part of our “if ____ then” block.

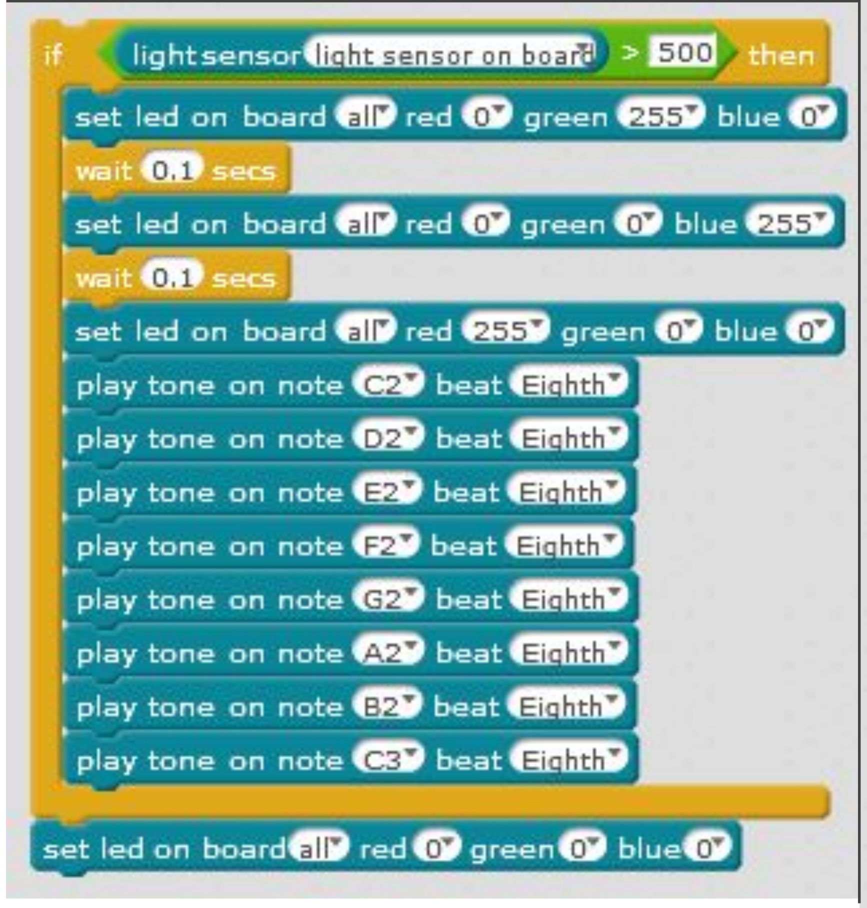
Remember, the mBot will work through the commands in the order we list them, so it is important to place things in the order we want them to happen. Our mBot should wake up first, sing second and stretch third.

To “wake up” our mBot, we will flash the lights green, then blue and finally red. We want to make sure that we can actually see each colour, so we are going to use a “wait” block to tell the mBot to pause in between each colour.

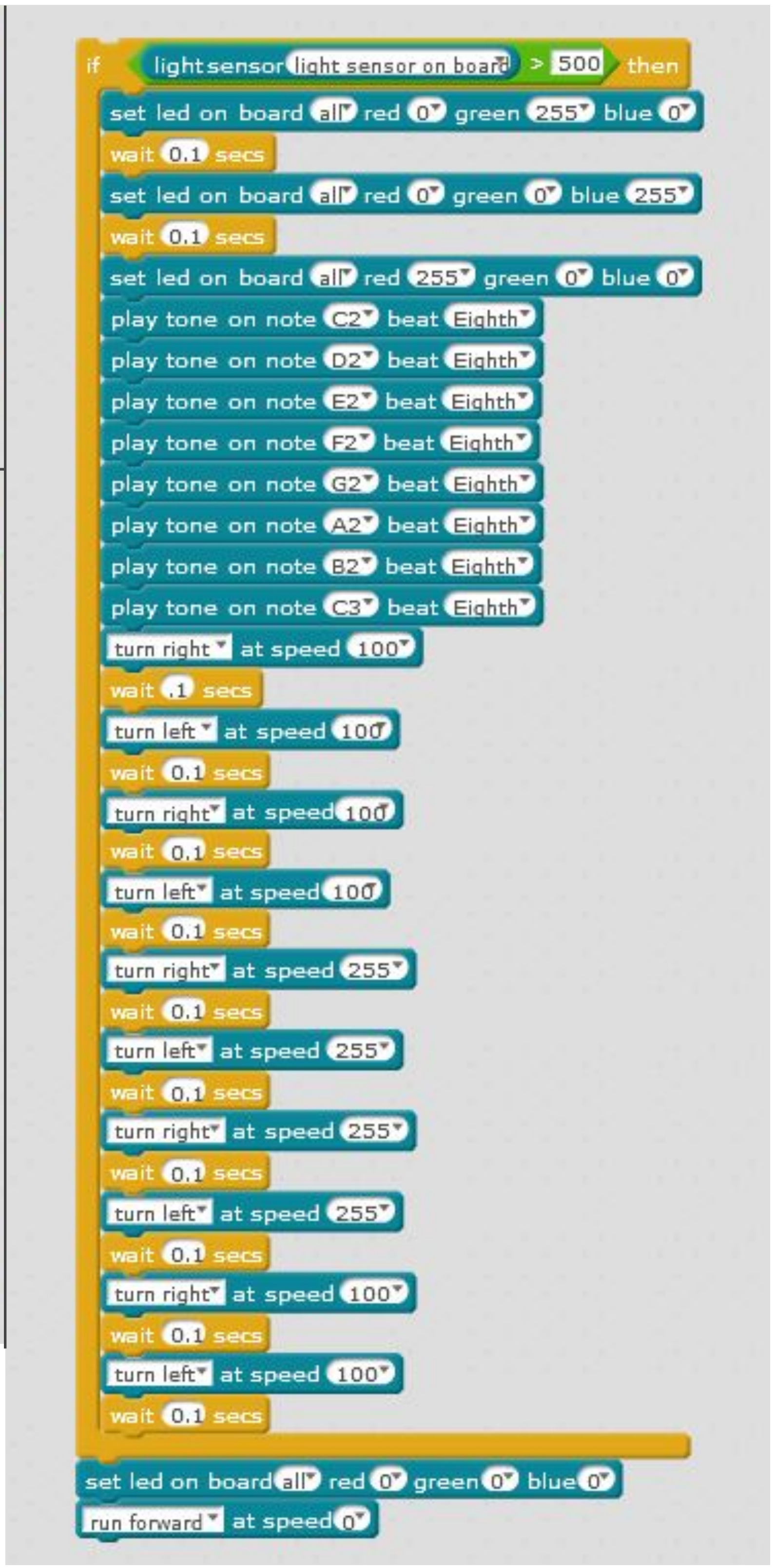
Remember, we need to give mBot something to do if the Light Sensor reads a value of less than 500. We can place these instructions outside of our “If” loop. We need to specifically tell it to keep it’s lights off, and to not be moving. We do not need to tell it to make no noise because the “play tone” block has a built in time (beat length).



```
if lightsensor light sensor on board > 500 then
    set led on board all red 0 green 255 blue 0
    wait 0.1 secs
    set led on board all red 0 green 0 blue 255
    wait 0.1 secs
    set led on board all red 255 green 0 blue 0
set led on board all red 0 green 0 blue 0
```



```
if lightsensor light sensor on board > 500 then
    set led on board all red 0 green 255 blue 0
    wait 0.1 secs
    set led on board all red 0 green 0 blue 255
    wait 0.1 secs
    set led on board all red 255 green 0 blue 0
    play tone on note C2 beat Eighth
    play tone on note D2 beat Eighth
    play tone on note E2 beat Eighth
    play tone on note F2 beat Eighth
    play tone on note G2 beat Eighth
    play tone on note A2 beat Eighth
    play tone on note B2 beat Eighth
    play tone on note C3 beat Eighth
set led on board all red 0 green 0 blue 0
```



```
if lightsensor light sensor on board > 500 then
    set led on board all red 0 green 255 blue 0
    wait 0.1 secs
    set led on board all red 0 green 0 blue 255
    wait 0.1 secs
    set led on board all red 255 green 0 blue 0
    play tone on note C2 beat Eighth
    play tone on note D2 beat Eighth
    play tone on note E2 beat Eighth
    play tone on note F2 beat Eighth
    play tone on note G2 beat Eighth
    play tone on note A2 beat Eighth
    play tone on note B2 beat Eighth
    play tone on note C3 beat Eighth
    turn right at speed 100
    wait .1 secs
    turn left at speed 100
    wait 0.1 secs
    turn right at speed 100
    wait 0.1 secs
    turn left at speed 100
    wait 0.1 secs
    turn right at speed 255
    wait 0.1 secs
    turn left at speed 255
    wait 0.1 secs
    turn right at speed 255
    wait 0.1 secs
    turn left at speed 255
    wait 0.1 secs
    turn right at speed 100
    wait 0.1 secs
    turn left at speed 100
    wait 0.1 secs
set led on board all red 0 green 0 blue 0
run forward at speed 0
```

Top Left: *Wake up Lights*

Right: *Wake up Lights, Song and Stretch*

Bottom Left: *Wake up Lights and Song*

The mBot box makes a great obstacle to practice with.

Choosing the Right Operator

In this activity, we want the mBot stop when the distance between it and an obstacle gets too small. The = operator and the < operator would both work, but which one is correct?

If the = block is used, the mBot will only react if that exact distance is met. If a distance of 20 cm is set as the stopping point, and an obstacle appears only 10cm away, the mBot will not stop for it!

When the < block is used, any distance less than 20 cm will cause the mBot to stop. Remember, the mBot only knows what we tell it, so we need to be careful when we give it instructions!

Ultrasonic Sensor

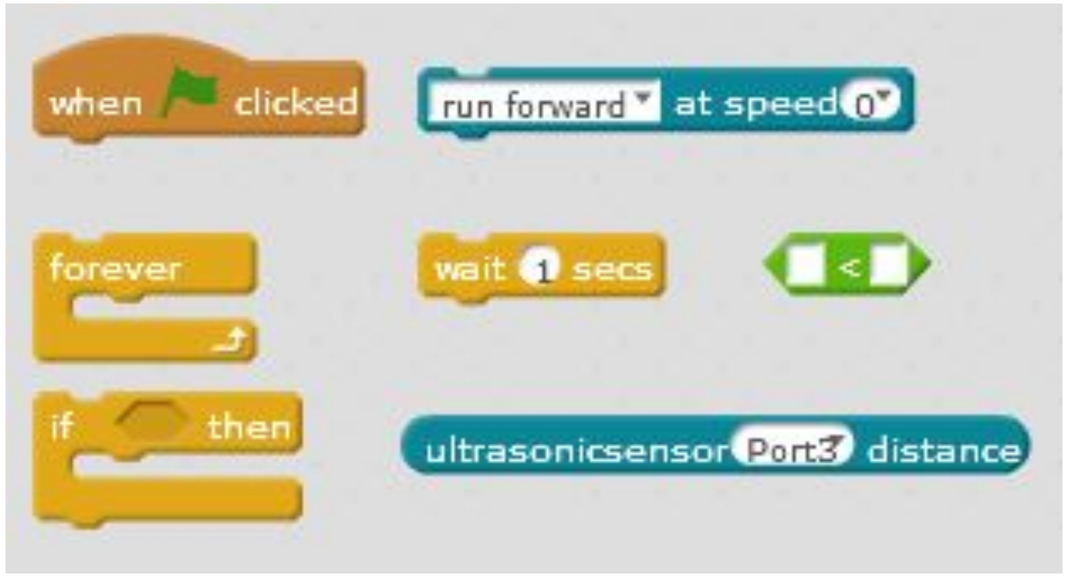
To keep our mBot from crashing into obstacles, we need to program it to use the Ultrasonic sensor. For this activity, it is a good idea to clear a space on the floor and set up one or two obstacles for the mBot to avoid. The mBot needs to be told that we want it to move forward until it detects an obstacle. Once it finds an obstacle, we need to tell it how to avoid it.

To begin, we will need a hat to start the program, a “forever” loop, an “if ___ then” loop, an “ultrasonic distance” block, a < operator, movement blocks and wait blocks.

Start out by dragging your chosen hat and the “forever” loop into your workspace. The first block we want inside this loop is a “run forward” block, because we want our mBot to start moving as soon as the program begins. Set the speed to 100. Next, we want to nest our “if ___ then” loop inside the “forever” loop because we *always* want our mBot to avoid obstacles.

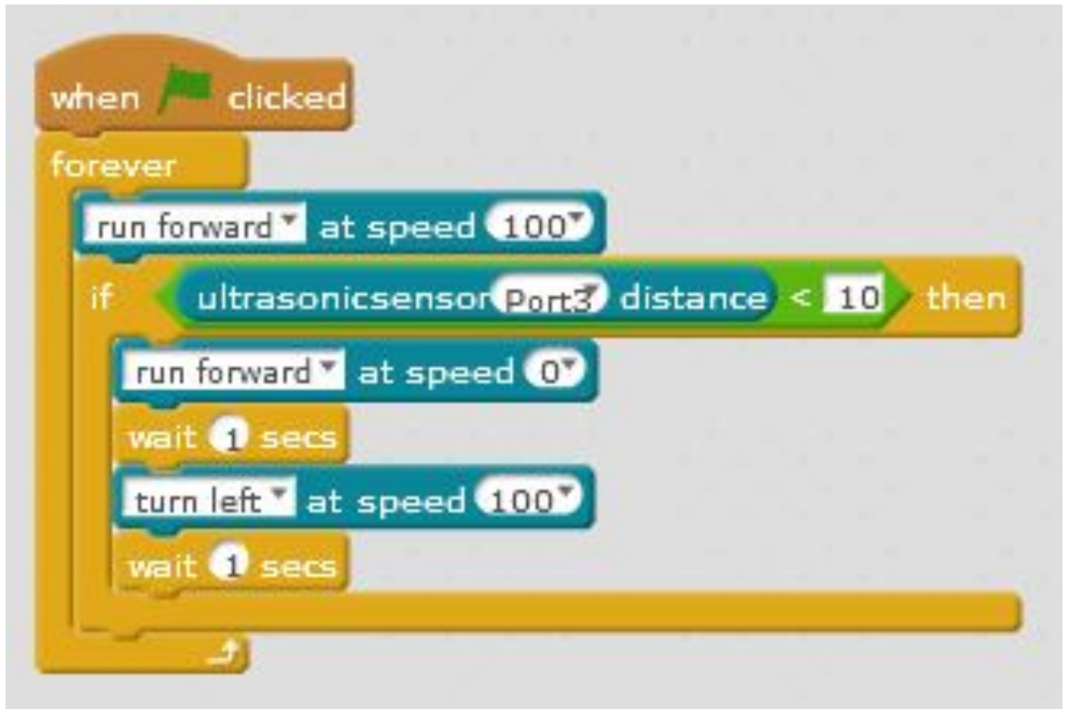
The “ultrasonicsensor” block should be nested into the < operator block. This will make it so the value our sensor returns will be compared against a number the we get to choose. (See the inset text on the previous page about choosing operators). We want our mBot to react once the Ultrasonic Sensor detects an obstacle that is 10 cm away, so we should type that in to the right of the < symbol.

These two blocks can go into our “if ___ then” loop. So far, the mBot will start out by driving forward at a speed of 100 as soon as we click the flag icon. The mBot will constantly check the value of the Ultrasonic sensor, and if it becomes less than 10 cm, it will do something. Our next step is to tell the mBot what that something is!



```
when clicked
run forward at speed 0
forever
wait 1 secs
[ ] < [ ]
if < > then
ultrasonicsensor Port3 distance
```

Blocks for Ultrasonic



```
when clicked
forever
    run forward at speed 100
    if ultrasonicsensor Port3 distance < 10 then
        run forward at speed 0
        wait 1 secs
        turn left at speed 100
        wait 1 secs
```

Code for Ultrasonic

To begin with, we want to make the mBot stop moving forward once it has picked up an obstacle. Drag a “run forward” block into the "If ___ then” loop. The direction could be set to anything, but the speed must be set to 0.

If we ran the code as is, the bot would drive up to the obstacle, stop, then try to move forward again. It would immediately detect the obstacle again and stop. It will stutter step it’s way up to the obstacle until it hits it! What we need to do is teach the mBot to avoid the obstacle somehow!

We can do this a number of different ways, but what we are going to start with is telling the mBot to stop, wait 1 second and then turn to the left for one second. This should help our mBot find a clear path. Be sure to place these command blocks beneath the “run forward at speed 0” block! Thats it!

Line Following Sensor

By using the line following sensor, we can teach our mBot to follow a path! The mBot is able to detect white or black, and this allows it to follow a black line!

Before starting programming, you'll need to get out the Figure of 8 paper inside your mBot box. Find space on the floor where you won't crash into anyone else's mBot and spread out the paper with the '8' facing up.

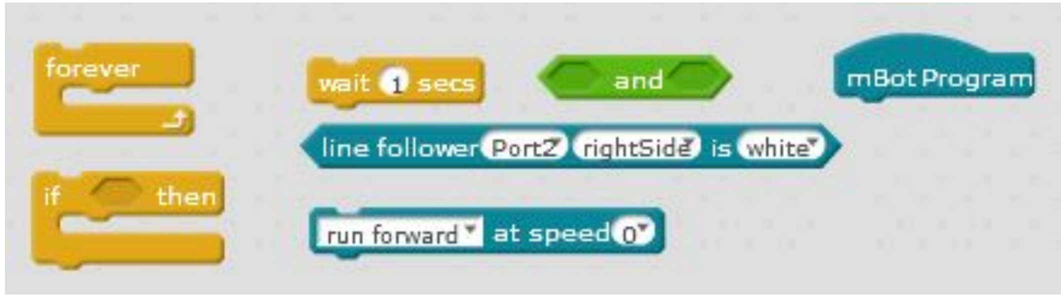
For this activity, we will need to upload our code to the mBot via the USB cable. In our previous activities, the mBot and computer were talking back and forth to one another. Since our figure of 8 is quite small, the mBot needs to be able to react to changes in the line detector more quickly than when it talks back and forth to the computer.

To achieve this, we need to use a special hat that is listed under the Robots section called "mBot Program". This tells the mBot that it will not be communicating with the computer after the code is uploaded.

First, we want to connect a "forever" loop to the hat. This will make sure that the mBot will continuously be checking the value of the Line Sensor.

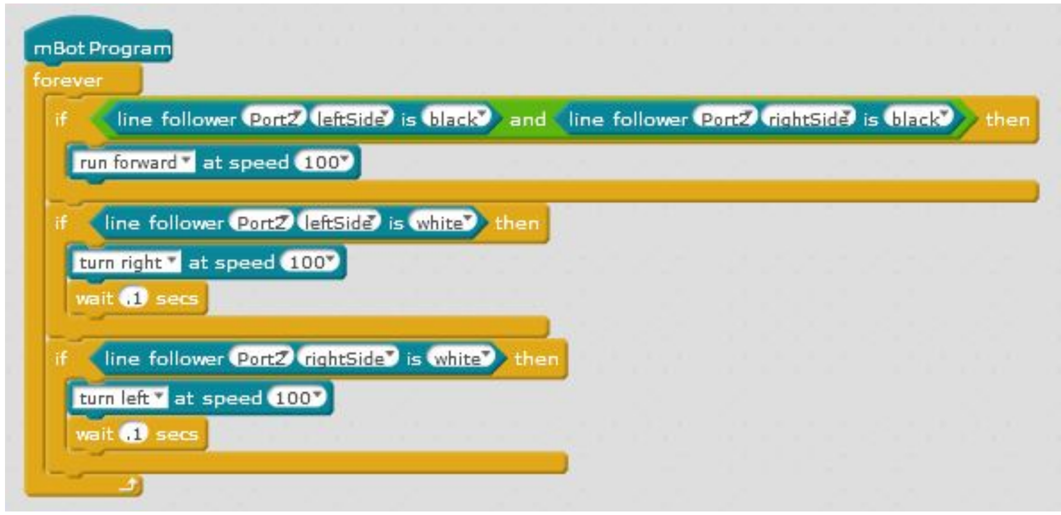
The Line Sensor is actually two sensors; one for the left side and one for the right. Either side can return a value indicating it is detecting white or black.

We need to decide what behaviour we want our mBot to perform. If both sensors pick up black, we want our mBot to drive forward. If the left side picks up white, the mBot should turn right to correct itself. When the sensor picks up white on the right side, it should turn left.



```
forever
if < > then
wait 1 secs
< and >
mBot Program
line follower Port2 rightSide is white
run forward at speed 0
```

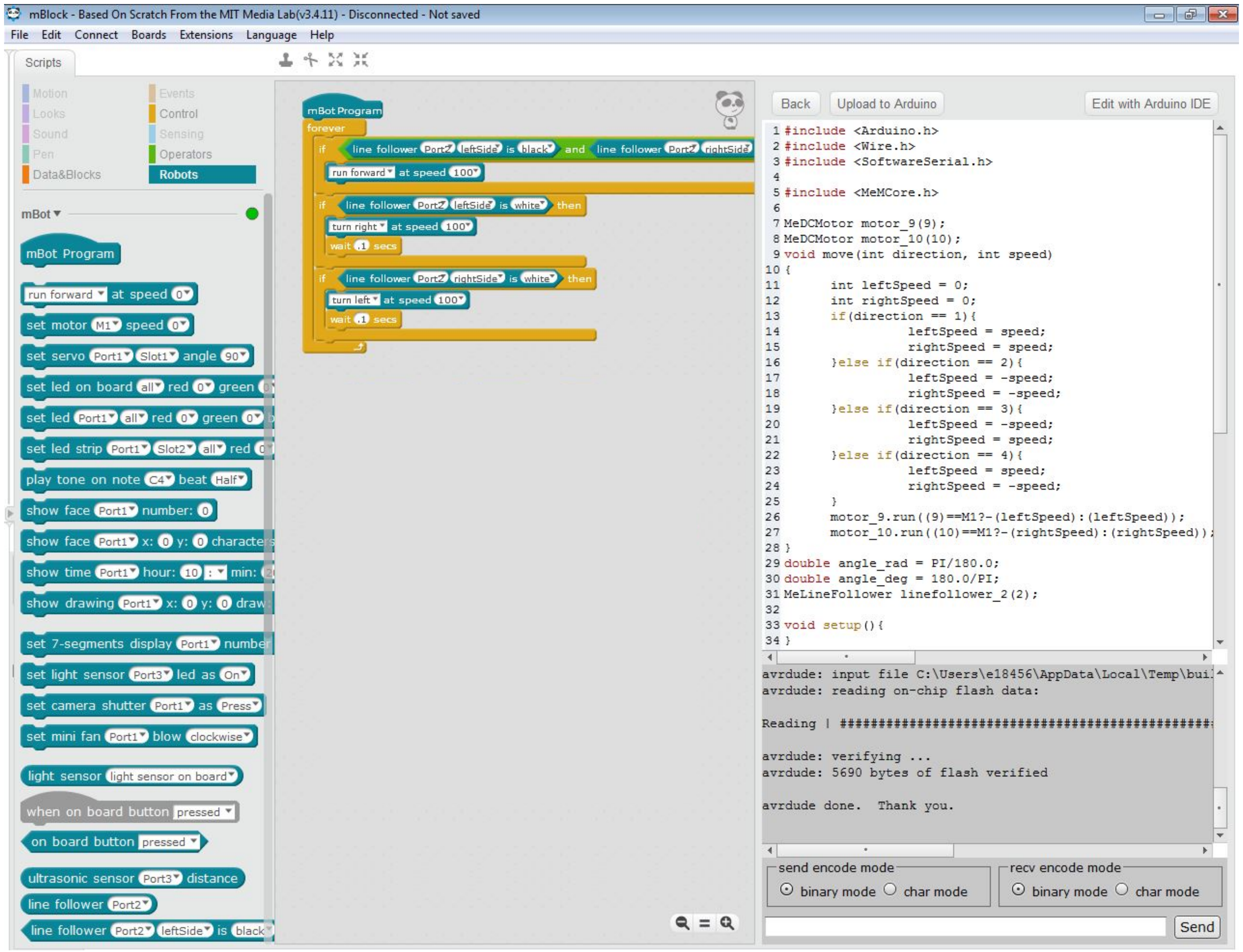
Blocks Required for Line Following



```
mBot Program
forever
  if <line follower Port2 leftSide is black> and <line follower Port2 rightSide is black> then
    run forward at speed 100
  if <line follower Port2 leftSide is white> then
    turn right at speed 100
    wait .1 secs
  if <line follower Port2 rightSide is white> then
    turn left at speed 100
    wait .1 secs
```

Sample Line Following Code

Uploading this code is a bit different than the ones in the past. To start uploading your code, make sure to connect the mBot to the laptop with the supplied USB cable.


mBlock - Based On Scratch From the MIT Media Lab(v3.4.11) - Disconnected - Not saved
File Edit Connect Boards Extensions Language Help
Scripts
Motion Looks Sound Pen Data&Blocks Events Control Sensing Operators Robots
mBot Program
forever
if line follower Port2 leftSide is black and line follower Port2 rightSide
run forward at speed 100
if line follower Port2 leftSide is white then
turn right at speed 100
wait .1 secs
if line follower Port2 rightSide is white then
turn left at speed 100
wait .1 secs
Back
Upload to Arduino
Edit with Arduino IDE
send encode mode
recv encode mode
binary mode
char mode
Send

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>

#include <MeMCore.h>

MeDCMotor motor_9(9);
MeDCMotor motor_10(10);
void move(int direction, int speed)
{
      int leftSpeed = 0;
      int rightSpeed = 0;
      if(direction == 1){
              leftSpeed = speed;
              rightSpeed = speed;
      }else if(direction == 2){
              leftSpeed = -speed;
              rightSpeed = -speed;
      }else if(direction == 3){
              leftSpeed = -speed;
              rightSpeed = speed;
      }else if(direction == 4){
              leftSpeed = speed;
              rightSpeed = -speed;
      }
      motor_9.run((9)==M1?-(leftSpeed):(leftSpeed));
      motor_10.run((10)==M1?-(rightSpeed):(rightSpeed));
}
double angle_rad = PI/180.0;
double angle_deg = 180.0/PI;
MeLineFollower linefollower_2(2);

void setup(){
}
```

```
avrdude: input file C:\Users\e18456\AppData\Local\Temp\bui
avrdude: reading on-chip flash data:

Reading | ##################################################

avrdude: verifying ...
avrdude: 5690 bytes of flash verified

avrdude done.  Thank you.
```

When uploading to the mBot directly, mBlock needs to convert our blocks into C

In the Connect menu, be sure to select the serial connection (instructions for this on page 5). The next step in uploading is to click on the "mBot program" hat.

This will shift your screen over to the left, hiding the "Stage" and "Sprites" menu. It will reveal what your code looks like written in the programming language C. To finish the upload, click the "Upload to Arduino" button. Be sure to hold your mBot so that it doesn't take off before you unplug the USB!

Take a moment and look at the code written out in C. Do you recognize any of the commands you used? Consider revisiting some of your old code and try to see what it looks like written in C!

Troubleshooting

Sometimes the programs we write do not work the way we intended them to, and sometime they don't even work at all! Instead of getting frustrated, try checking to see if any of the following things are happening:

Are the mBot batteries charged & plugged in? Is the mBot power switch in the ON position? Are all of the motors and sensor plugged in to the correct ports? (left motor should be in port M1, right motor in port M2, Ultrasonic sensor is port 3, Line Following sensor in port 2)

If the mBot isn't responding to the computer and you are using a USB Serial connection, check to make sure that you have selected the correct COM port. (see instructions on page 6)

If the mBot isn't responding and you are trying to use 2.4G Serial (wireless connection), check to make sure this is the option selected in the Connections menu.

Make sure that the wireless dongle is plugged into the laptop, and that it is labelled with the same number as the robot. If it still doesn't work, look at the mBot itself. Through the plastic on top of the mBot, beside where the battery plugs you will see a small black board. This is the what receives the wireless signal. You can re-pair the mBot to the dongle by pressing the plastic on top of this board. Make sure the mBot is next to the plugged in dongle when you do this. There is a blue light that will flash on the mBot when you press the button that will switch to a solid blue light once it is paired.

Once you've made sure the mBot is properly connected, we need to start looking at the code. Remember, robots only know what we tell them and do what we ask them. A common mistake is forgetting to put commands into the correct loop. If you want the mBot to do something continuously, make sure that it is inside a "forever" loop, or else it will only perform the task once! This can make it seem like the robot is not following your directions.

Be sure to carefully read over the code you've created. It can be helpful to write down a flowchart or list of what you want the mBot to do, and compare that to your code step by step.

The internet is full of tutorials and videos to help you solve problems too. Don't get discouraged! We learn the most when things don't work on the first try!

Extension Activities

Once you have some of the basics down, it can be fun to challenge yourself to try new things. There are many ways that mBot can be used to explore other curricular areas, with a little bit of ingenuity.

If you explore the Operators section, you'll find blocks that allow the mBot to perform various Math operations. The Ultrasonic sensor can be used to measure distances in centimetres, and the mBot can be programmed use this number to convert the distance into different units.

Students can use mBlock to create interactive stories and games about whatever topic you (or they) choose. They can make their stories interact with the mBot, so it can react to story events.

If your class isn't quite ready for lots of programming, the class can use the driver control function to play balloon jousting by attaching a balloon and bamboo skewer. A World Cup of mBot Soccer can be held.

Consider making your mBot a part of a Rube Goldberg machine, or designing a specific task for your students to complete using their mBot!

A class could theorize what kind of environment the mBot would most likely live in and create a model of an mBot community. The mBots could be given the task of finding their way through a maze using only the ultrasonic sensor, or using just the line following sensor.

Learning to program the mBot is really just the tip of the iceberg. The only limiting factor is time; the more time spent using the mBot, the more rich and varied the learning experience will be.