Buggy and Controller

This project consists of two programs and requires two BBC micro:bits and a Kitronic Servo-Lite control board.

1. One to control two servo driven wheels on a buggy based on commands received via radio
2. One to send the commands to the buggy controller to instruct it to move.

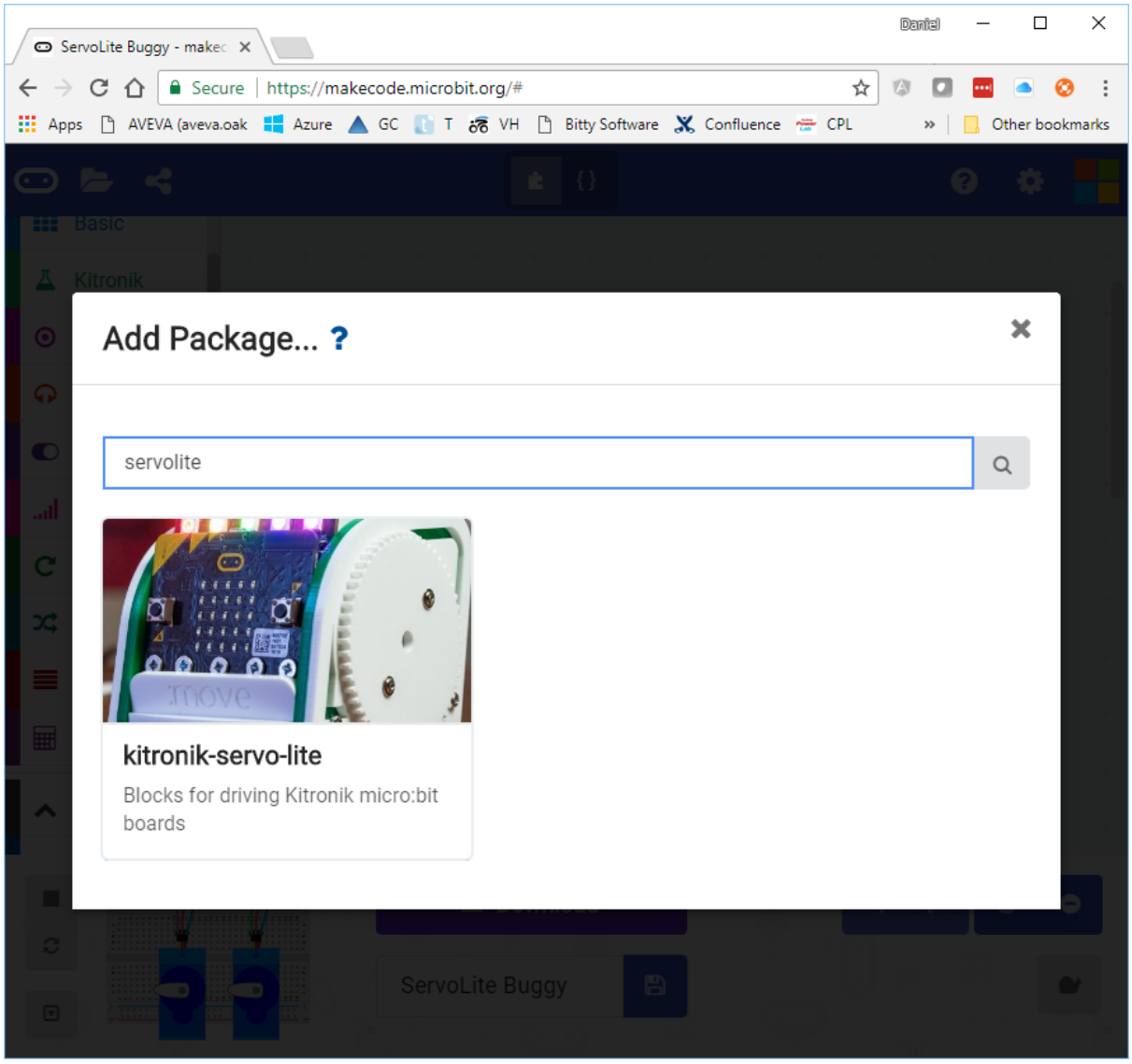
Buggy

The buggy is the simplest program but requires two continuous rotation servos connected to wheels. The BBC micro:bit can only drive one servo so it is recommended to use a Kitronic servo-lite .

Step 1 – Add the Kitronic servo blocks

These blocks make it easier to write code to control the servos.

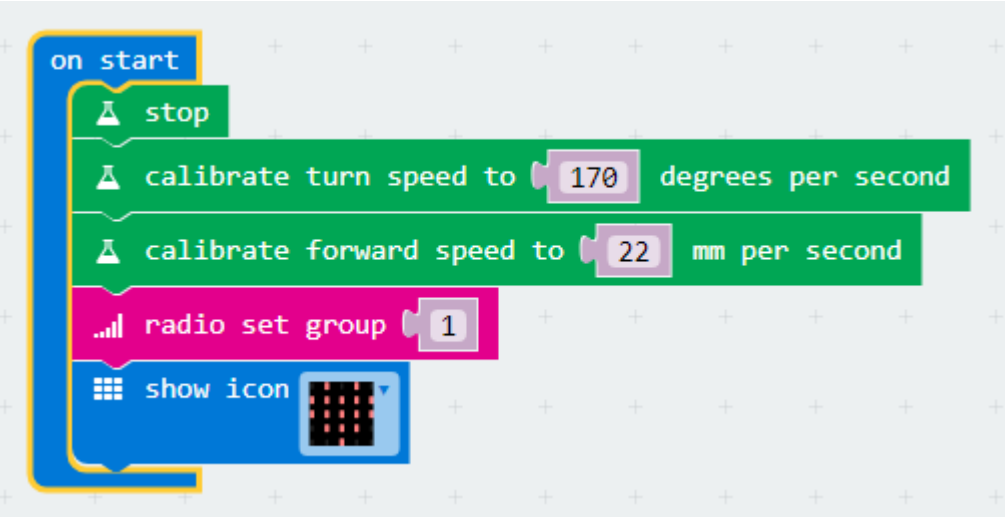
- Select “Add Package” from the Advanced section of the Blocks tool.
- Type in servolite and click the search button.
- Select the kitronik-servo-lite package.



Step 2 – Calibrate the servos and set the radio group

Setting the calibration values allows the user to balance the servos so that they move by the desired distance and turn by the desired speed when commanded to do so.

We also want to set the radio group to a value that the controller program will transmit to. If there are multiple buggies, each will need to be set to a different radio group.

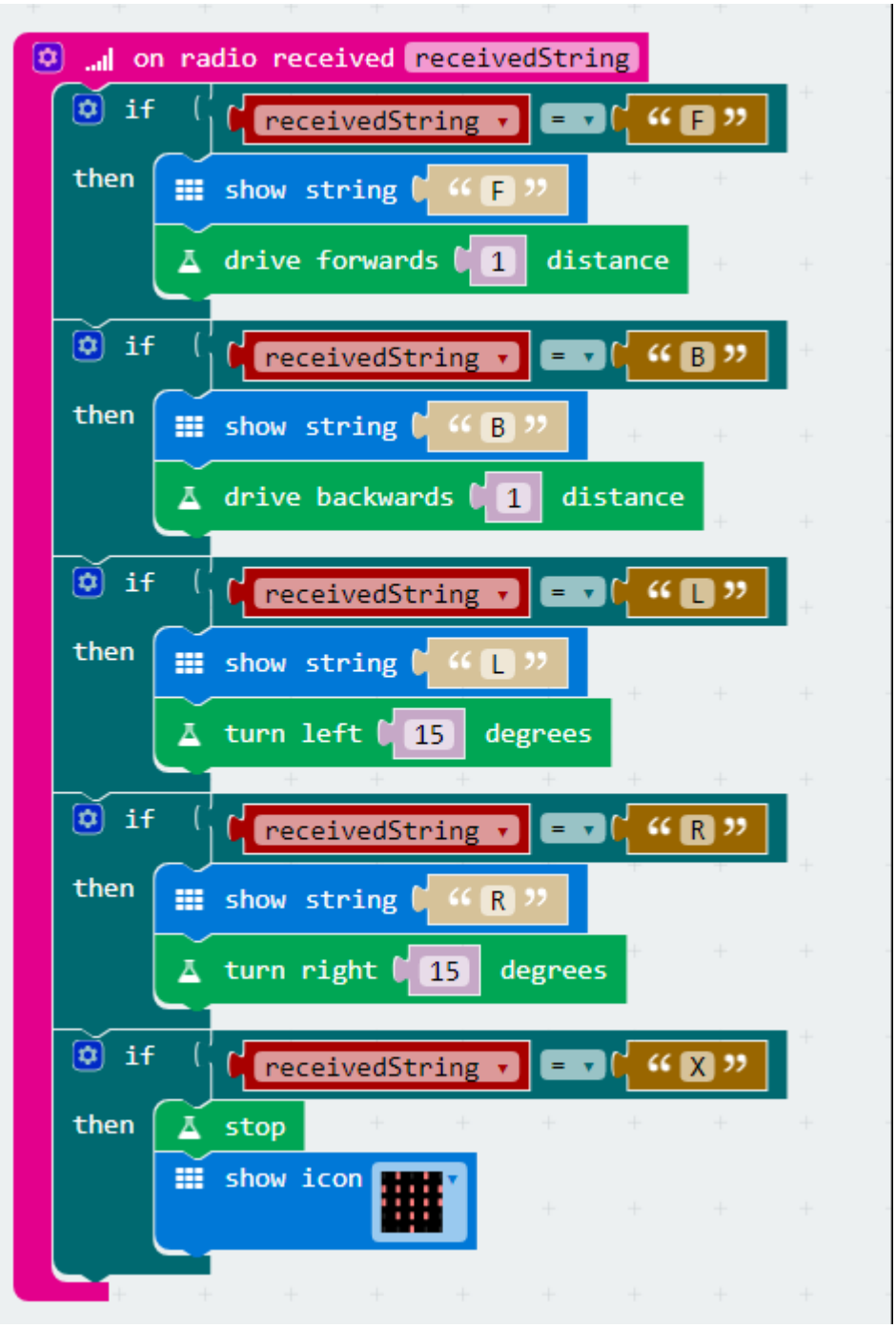


Step 3 – Receiving and processing the commands

The buggy will respond to the following string commands over the radio:

- F – move forward
- B – move backwards
- L – rotate left
- R – rotate right
- X – stop

The code to drive the buggy is given below. Can you extend this program so that the buggy can respond to more commands? How about a command to make the buggy do a wiggle or dance?


on radio received receivedString
if receivedString = "F"
then
show string "F"
drive forwards 1 distance
if receivedString = "B"
then
show string "B"
drive backwards 1 distance
if receivedString = "L"
then
show string "L"
turn left 15 degrees
if receivedString = "R"
then
show string "R"
turn right 15 degrees
if receivedString = "X"
then
stop
show icon

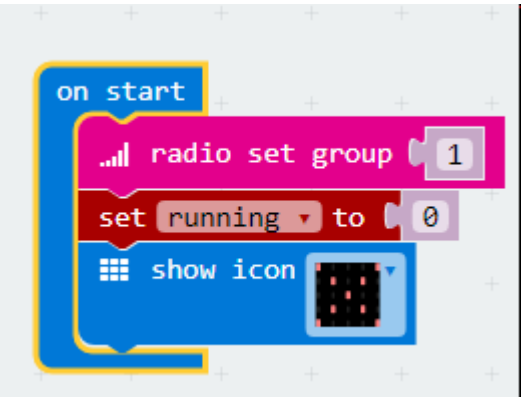
Controller

The controller program will send commands to the buggy based on the orientation of the BBC micro:bit. This program makes use of the accelerometer and compass sensors in the micro:bit to send commands to the buggy based on how much it is tilting.

Step 1 – Set the radio group and initialise the program

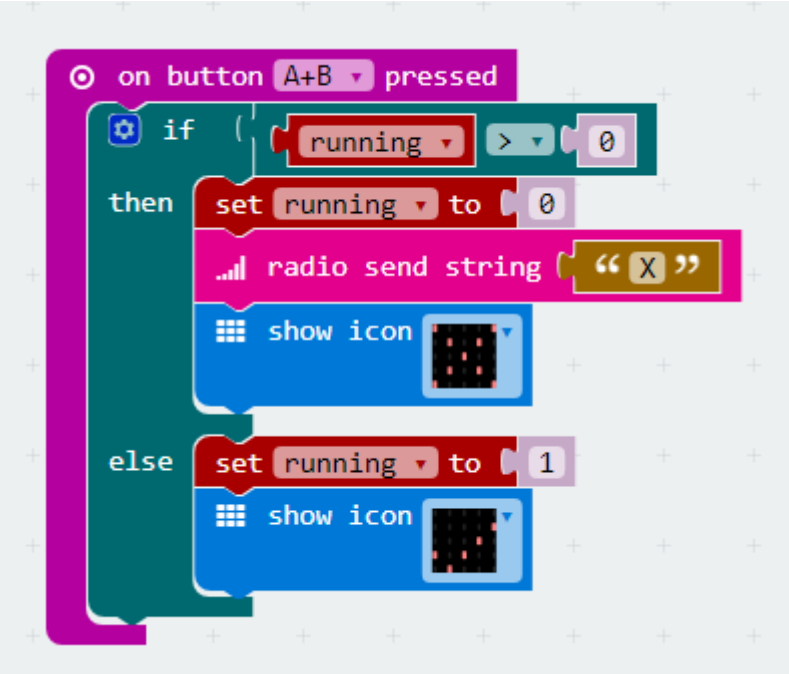
The radio group selected needs to match the radio group that the buggy is using.

A new running variable will also need to be created.



Step 2 – Connect and Disconnect from the buggy

Pressing the A and B buttons together will be used to connect and disconnect from the buggy. When disconnecting, we transmit the stop command (an X) to the buggy.

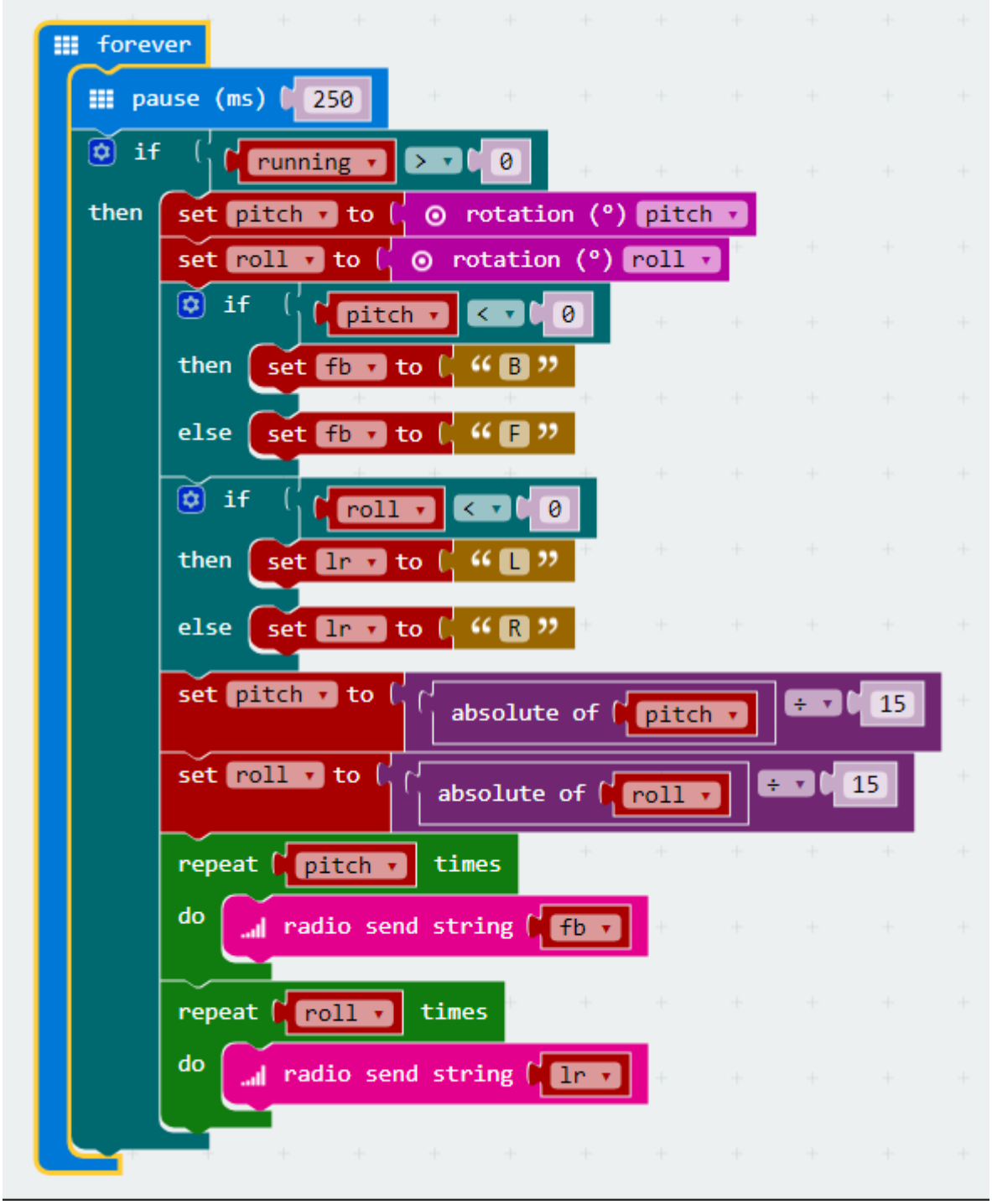


Step 3 – Sending the control signals

The following variables will need to be created.

- Pitch
- Roll
- fb
- lr

fb is short for Front or Back and “lr” is short for Left or Right.



The speed with which the commands are sent can be controlled by adjusting the length of the pause. Reduce the pause to speed up the rate commands are sent, increase the pause slow down the rate the commands are sent.

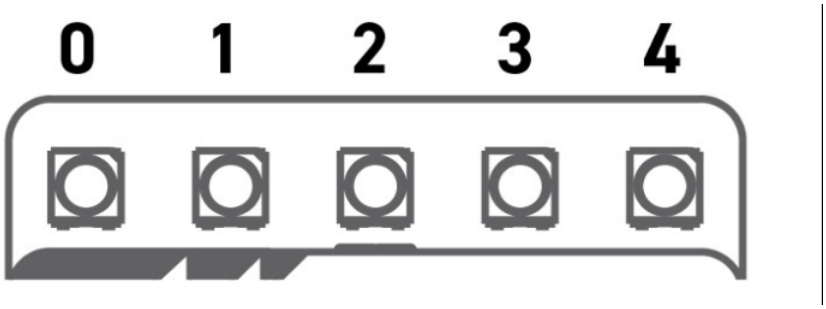
The code determines whether to send commands to go forward or backwards based on whether the pitch is a positive or negative number. The same method is used for roll to determine left or right.

The values of pitch and roll are divided by 15 to reduce the sensitivity of the pitch and roll. Try adjusting these values to see the effect it has.

Depending on how the servos are controlled on the buggy, it might be necessary to swap left and right or forward and backward to get the buggy to move in the desired direction.

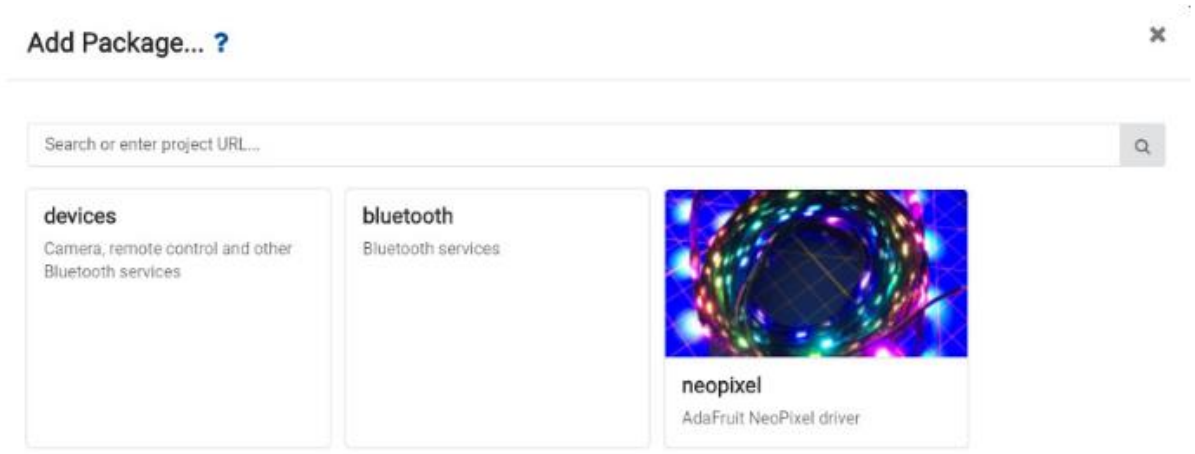
Adding Lights to the Buggy

The servo-lite board used on the Buggy has 5 NeoPixel lights across the top. Each NeoPixel has a number from zero to four as shown on the diagram below.



These LEDs can be controlled using the NeoPixel package which adds blocks to control them and can be added by following these steps.

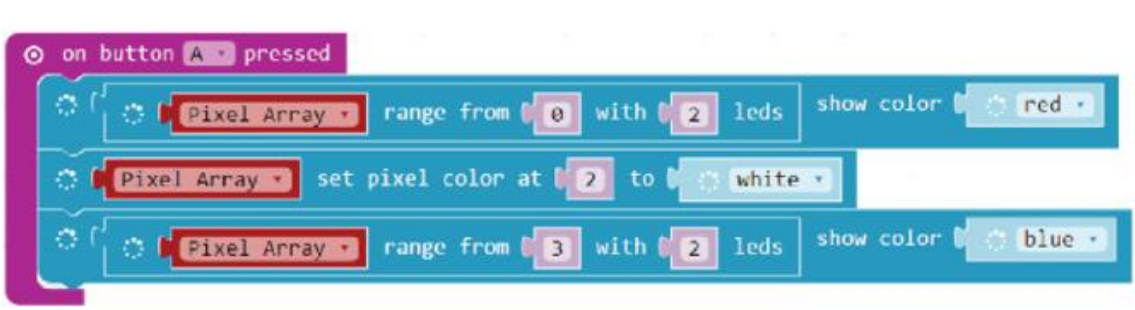
- Select “Add Package” from the Advanced section of the Blocks tool.
- Type in neopixel and click the search button.
- Select the AdaFruit NeoPixel package.



A new variable will need to be added to setup and control the NeoPixels. This variable should be called PixelArray and needs configuring as below.

To turn all of the NeoPixels on, the A button can be configured with the code as below.

It is also possible to address NeoPixels individually or as groups as the code below shows.



Write code so that the NeoPixels do the following when the buggy receives these commands.

- F – Set the middle 3 NeoPixels to green
- B – Set the middle 3 NeoPixels to Orange
- L – Set the first NeoPixel to Orange
- R – Set the last NeoPixel to Orange
- X – Set the middle 3 NeoPixels to Red

Further Experimentation

- Consider using the A button press events or the B button press events to send new commands to the buggy; such as a wiggle, dance or turn on the spot.
- Can you add different or more complex lighting effects to the buggy?
- Can you adjust the brightness of the NeoPixels on the buggy based on light effect?
- Can you create a more complex controller using the JoyStick Bit?