Part 01

Hack Lego Boost with Raspberry Pi

The Lego Boost robotics kit is designed to make learning to build and program robots fun and easy. Let's hack it with some Python

MAKER

Mike Cook

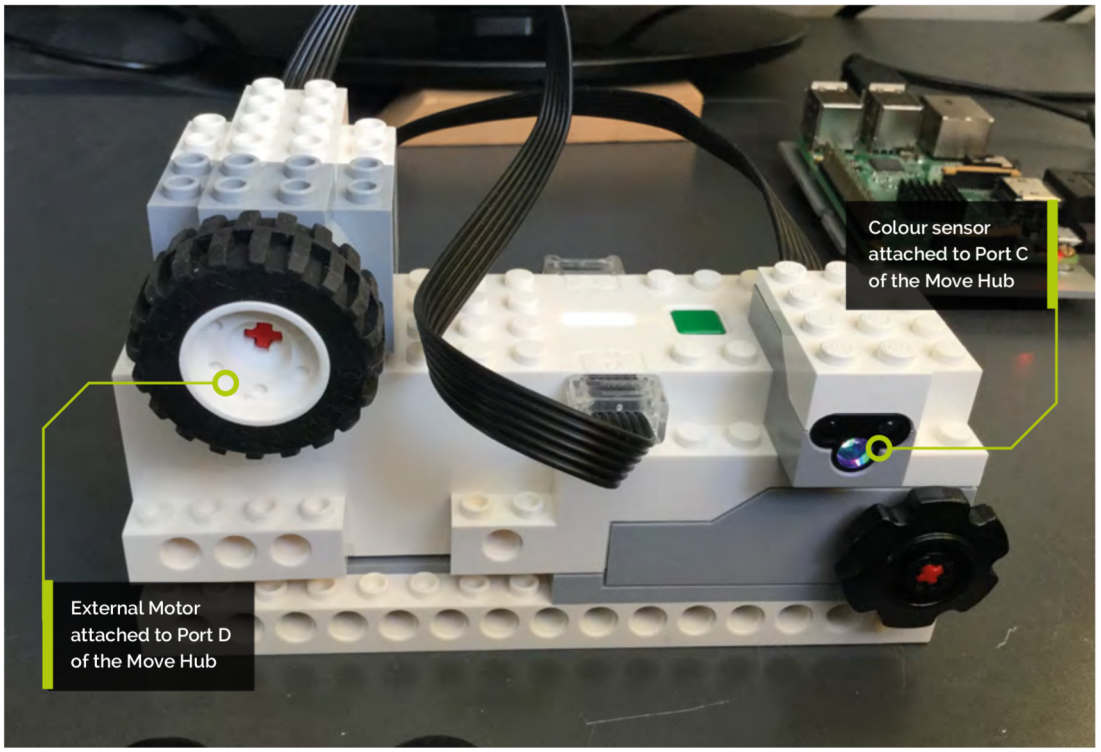
Veteran magazine author from the old days, writer of the Body Build series, plus co-author of *Raspberry Pi for Dummies*, *Raspberry Pi Projects*, and *Raspberry Pi Projects for Dummies*.

magpi.cc/TPaUfT

The Lego Boost is designed to be run from an app on an Android or iPad tablet, using a graphics programming language not unlike Scratch. It makes a good job of this and is easy for kids to pick up, but these sorts of languages have their limitations. They can be inflexible and difficult to read, especially as the code gets bigger. By using Python, much more complex programs can created, many that are not possible with graphics-based code. So, give your Boost a boost by letting Python do the controlling.

01 What is Lego's Boost?

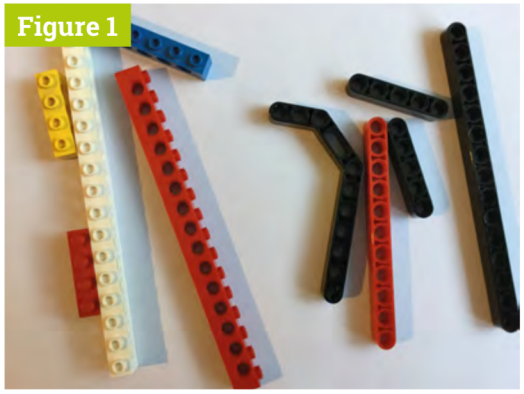
Unlike the previous Mindstorms robotics systems from Lego, the Boost system has no controlling brick to run code – instead, instructions representing the program are sent one at a time directly, over Bluetooth, in real-time to the Move Hub. This has built-in motors, LEDs, tilt sensor, and push-button. You can also plug into the Hub a smaller motor and a distance sensor or colour sensor. Also, Mindstorms uses Lego Technic beams for most constructions, known as a studless system, whereas the Boost uses the more conventional, studful brick form of construction. See **Figure 1**.



You'll Need

- Lego 17101 Boost Creative Toolbox **magpi.cc/JtUiDe**
- Raspberry Pi 3/3B+/3A+/Zero W

Figure 1



▲ **Figure 1** Studful bricks on the left, studless beams on the right

02 Lego's software

The standard Lego software provides a good structured learning system where the user builds a bit, programs it, and then moves on to building more. The big project is Vernie the robot and we'd recommend you do this first because the parts are already bagged up to make construction of this project easy. Vernie is built up in stages and the code you are asked to run at each stage gets increasingly complex. You can't progress to the next level without trying, or pretending to try, the current phase. We recommend you try this first, not least for the firmware updates it can offer to your Hub – **see Figure 2**.

03 Libraries

There are a few libraries for connecting Boost to Python – see a summary at **magpi.cc/iNfUQg** from Jorge Pe, author of the original library. He hacked the system by using a Bluetooth protocol sniffer. These days, Lego has released the protocol documentation at **magpi.cc/YfiAbL**, but it's a heavy read. We tried a few libraries and it seems the most developed and actively supported one is pylgbst

Figure 2

▲ **Figure 2** The Lego Boost program environment running on an iPad or Android

mike's_demo.py

DOWNLOAD THE FULL CODE: magpi.cc/dhaAam

Language: **Python**

```
001.  # #!/usr/bin/env python3
002.  # coding=utf-8
003.  # Mike's Demo - put the LEGO Boost trough its paces
004.  # By Andrey Pokhilko & Mike Cook Feb 2019
005.
006.  from time import sleep
007.  import time
008.  from pylgbst import *
009.  from pylgbst.movehub import MoveHub, COLORS
010.  from pylgbst.peripherals import EncodedMotor, TiltSensor,
      Amperage, Voltage
011.
012.  def main():
013.    print("Mike's Demo - put the LEGO Boost trough its paces")
014.    conn=get_connection_auto()
015.    try:
016.      movehub = MoveHub(conn)
017.      demo_voltage(movehub)
018.      demo_button(movehub)
019.      demo_led_colors(movehub)
020.      demo_motors_timed(movehub)
021.      demo_motors_angled(movehub)
022.      demo_port_cd_motor(movehub)
023.      demo_tilt_sensor_simple(movehub)
024.      demo_tilt_sensor_precise(movehub)
025.      demo_color_sensor(movehub)
026.      demo_motor_sensors(movehub)
027.      sleep(1)
028.      print("That's all folks")
029.    finally:
030.      conn.disconnect()
031.
032.  def demo_voltage(movehub):
033.      print("Reading voltage & current for a short time")
034.      def callback1(value):
035.          print("Amperage: %.3f" % value)
036.
037.      def callback2(value):
038.          print("Voltage: %.3f" % value)
039.
040.      movehub.amperage.subscribe(callback1, mode=Amperage.MODE1,
      granularity=0)
041.      movehub.amperage.subscribe(callback1, mode=Amperage.MODE1,
      granularity=1)
042.
043.      movehub.voltage.subscribe(callback2, mode=Voltage.MODE1,
      granularity=0)
044.      movehub.voltage.subscribe(callback2, mode=Voltage.MODE1,
      granularity=1)
045.      sleep(5)
046.      movehub.amperage.unsubscribe(callback1)
047.      movehub.voltage.unsubscribe(callback2)
048.
049.  def demo_button(movehub):
050.      global notPressed
051.      print("Please press the green button")
```

Top Tip

Colour sensor testing

The colour sensor only works when the object is between about 3 and 5 mm. Use tiles like part 4566179 and 4560181 to test the colours.

from Andrey Pokhilko. In fact, during the course of writing this, a new version was released. It is still not finished but, with a little tweaking, it is good.

04 Installing the pylgbst library

First off, we need to install some dependences. Open a Terminal window and enter:

```
sudo apt-get install python3-pip
sudo pip3 install pexpect
sudo pip3 install pygatt
```

Next, get the library:

```
wget https://github.com/undera/pylgbst/archive/master.zip
unzip master.zip
cd pylgbst-master
sudo python3 setup.py install
```

05 Fixing the colours

There seems to be an error in one of the files defining the colours for the RGB LED. Fix this by opening a Terminal and entering:

```
sudo nano /usr/local/lib/python3.5/dist-packages/pylgbst/constants.py
```

Scroll down using the cursor keys until you see:

```
COLOR_ORANGE = 0x09
```

...and change it to:

```
COLOR_ORANGE = 0x08
```

To save the changes, press **CTRL**+**X**, then **Y**, and **ENTER**.

06 Documentation

The bulk of the documentation is in the **README.md** file and we found that not all of it works with the Raspberry Pi. But, it does contain some short single-function examples. Basically, to get the Move Hub to do things, you send it a command, as you might expect; however, to get information back from the Move Hub, you have to subscribe to the appropriate stream. When you do, you specify the name of a function that will be called when there is new data from whatever sensor you subscribed from. You must unsubscribe from the sensor before your Python program finishes.

07 Demo example

In the library's **examples** folder is a **demo.py** program; unfortunately, this will not run without error as is – it seems to be more designed for fault finding on the Bluetooth connections than showing things working. We have rewritten this to add extra functions and correct the syntax errors – see the **mike's_demo.py** listing. While it will run with any Move Hub, incorporated into a model or not, we found it best to use a basic setup that didn't run away off the desk when the motors moved. Also, adding wheels to the motors made it easy to turn them when testing the motor read angle function.

08 Preparing the Hub

Refer to **Figures 3** to **5** with these instructions. Get the Move Hub and fix on top of it the sensor (part 6182145) and the motor (part 6181852) and plug them into ports C and D. It doesn't matter which port you use, the software will find them. Next, put a red axle stub (part 4142865) in each of the two Hub motors, and the external motor. Fit a wheel (part 4662228) on each of the Move Hub's axles, and a wheel and tyre (parts 6092256 and 4619323) onto the external motor. Finally, add two long beams (part 4508661) under the hub to stop the wheels from moving it.

Figure 3 The top of the minimum demo build

Figure 4 The front of the minimum demo build

Figure 5 The end of the minimum demo build

Figure 3

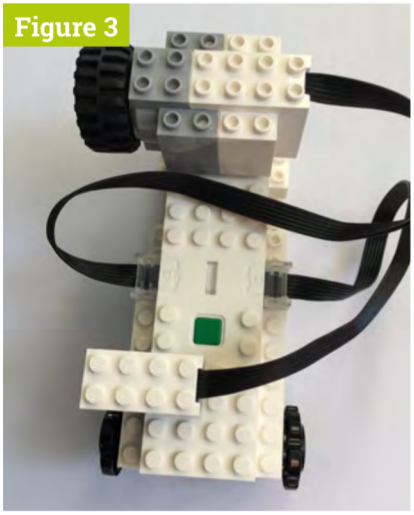


Figure 4

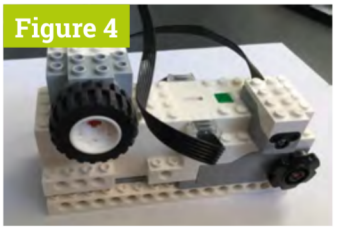
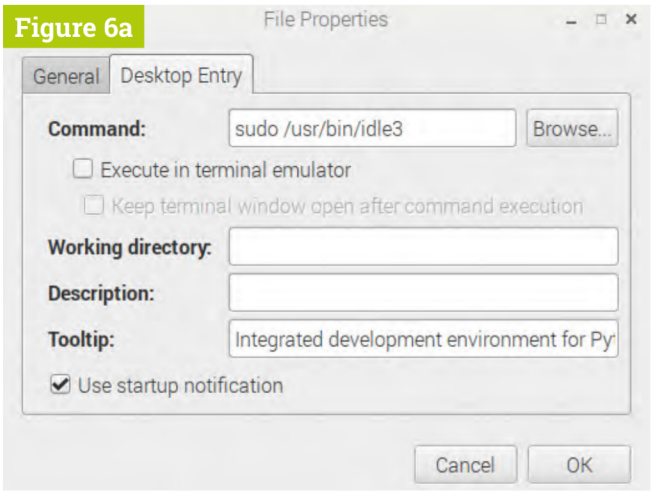
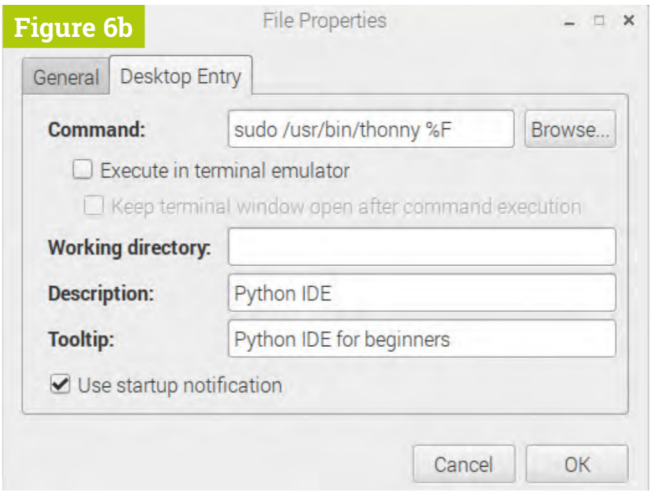


Figure 5

mike's_demo.py (continued)

```
052.      notPressed = True
053.
054.      def call_button(is_pressed):
055.          global notPressed
056.          if is_pressed :
057.             print("Thank you button pressed")
058.          else:
059.            print("Now it is released")
060.            notPressed = False
061.
062.      movehub.button.subscribe(call_button)
063.      while notPressed:
064.          sleep(0.4)
065.      sleep(0.4)
066.      movehub.button.unsubscribe(call_button)
067.
068.  def demo_led_colors(movehub):
069.      # LED colors demo
070.      print("LED colours demo")
071.      for colour in range(1,11):
072.          print("Setting LED colour to: %s" %
      COLORS[colour])
073.          movehub.led.set_color(colour)
074.          sleep(1)
075.
076.  def demo_motors_timed(movehub):
077.      print("Motors movement demo: timed")
078.      for level in range(0, 101, 10):
079.          levels = level / 100.0
080.          print(" Speed level: %s" %  levels)
081.          movehub.motor_A.timed(0.2, levels)
082.          movehub.motor_B.timed(0.2, -levels)
083.      print("now moveing both motors with one command")
084.      movehub.motor_AB.timed(1.5, -0.2, 0.2)
085.      movehub.motor_AB.timed(0.5, 1)
086.      movehub.motor_AB.timed(0.5, -1)
087.
088.
089.  def demo_motors_angled(movehub):
090.      print("Motors movement demo: angled")
091.      for angle in range(0, 361, 90):
092.          print("Angle: %s" % angle)
093.          movehub.motor_B.angled(angle, 1)
094.          sleep(1)
095.          movehub.motor_B.angled(angle, -1)
096.          sleep(1)
097.
098.      movehub.motor_AB.angled(360, 1, -1)
099.      sleep(1)
100.      movehub.motor_AB.angled(360, -1, 1)
101.      sleep(1)
102.
103.
104.  def demo_port_cd_motor(movehub): # Move motor on port
      C or D
105.      print("Move external motor on Port C or D 45
      degrees left & right")
106.      motor = None
107.      if isinstance(movehub.port_D, EncodedMotor):
108.          print("Rotation motor is on port D")
109.          motor = movehub.port_D
110.      elif isinstance(movehub.port_C, EncodedMotor):
111.          print("Rotation motor is on port C")
112.          motor = movehub.port_C
113.      else:
114.          print("Motor not found on ports C or D")
115.      if motor:
116.          print("Left")
117.          motor.angled(45, 0.3)
118.          sleep(3)
119.          motor.angled(45, -0.3)
120.          sleep(1)
121.
122.          print("Right")
123.          motor.angled(45, -0.1)
124.          sleep(2)
125.          motor.angled(45, 0.1)
126.          sleep(1)
127.
128.  def demo_tilt_sensor_simple(movehub):
129.      print("Tilt sensor simple test. Turn Hub in
      different ways.")
130.      demo_tilt_sensor_simple.cnt = 0
131.      limit = 10 # number of times to take a reading
132.
133.      def callback(state):
134.          demo_tilt_sensor_simple.cnt += 1
135.          print("Tilt # %s of %s: %s=%s" % (
      demo_tilt_sensor_simple.cnt, limit, TiltSensor.TRI_
      STATES[state], state))
136.
137.      movehub.tilt_sensor.subscribe(callback,
      mode=TiltSensor.MODE_3AXIS_SIMPLE)
138.      while demo_tilt_sensor_simple.cnt < limit:
139.          sleep(1)
140.      movehub.tilt_sensor.unsubscribe(callback)
141.
142.  def demo_tilt_sensor_precise(movehub):
143.      print("Tilt sensor precise test. Turn device in
      different ways.")
144.      demo_tilt_sensor_simple.cnt = 0
145.      limit = 50
146.
147.      def callbackTilt(roll, pitch, yaw):
148.          demo_tilt_sensor_simple.cnt += 1
149.          print(
      "Tilt #%s of %s: roll:%s pitch:%s yaw:%s" % (demo_
      tilt_sensor_simple.cnt, limit, roll,pitch,yaw))
150.
151.      # granularity = 3 - only fire callback function
      when results change by 3 or more
152.      movehub.tilt_sensor.subscribe(callbackTilt,
```





▲ **Figure 6** Changing the command so that IDLE or Thonny will run in supervisor mode

09 Running Mike's demo

Make sure Bluetooth is turned on by clicking on the Bluetooth icon on the top menu bar, run the Python code, and immediately push the green button on the Move Hub. The LED next to this will start flashing to indicate that the Bluetooth Hub is looking for something to pair with. This will turn a steady blue when connected and the demo will then start, printing out instructions on the screen. After the battery's voltage and current have been read out, the program will ask you to press the green button; you need to do this for the program to advance.

10 More demo instructions

After the green button is pressed, the distance sensor is tested; this repeats for 100 measurements. You can change the distance sensor reading by waving your hand in front of it: it returns a value in inches, to the nearest inch, for distances above one inch; for smaller distances, it returns finer values. The program converts this into millimetres for display. The tilt sensor tests require you to pick up and turn the Hub, and the motor angle read requires you to turn the motor wheels by hand. When no movement of the wheels has been detected over the last five seconds, the demo ends.

11 Further examples

The library's **examples** folder also contains other examples. Unfortunately, they don't work on the Raspberry Pi due to the method used to include packages. The `from . import *` line, which means from the current directory import everything, returns an error message saying the parent directory is missing. This might have something to do with the author not using a Raspberry Pi in his testing. Note, there is another problem but this is down to Lego, as it also happens on the tablets running the official software. Colours set or reported as cyan are actually green, so we have to cope with this in our code.

12 Move Hub connection problems

Here at the Bakery we have two Raspberry Pi boards: one with the current Raspbian release, and the other the previous one. Oddly enough, we found that running normally was fine with the older release, but with the newer one the code needed to be run in supervisor mode in order for it to connect. This can be done by typing `sudo IDLE3` in a Terminal window. If you want to permanently run IDLE in the supervisor mode from the desktop, go to the IDLE 3 menu entry, and right-click. Then choose the Properties menu, and select the Desktop Entry tab. Now edit the command box to put 'sudo ' in front of what is there. The same applies for Thonny – see **Figure 6**.

In conclusion

Now we've got the basics going, we're all ready to do some interesting stuff. Next month, we'll make a game showing how we can use our Lego Boost with the Pygame framework.

Top Tip

Lego part numbers

On the back of the cardboard play mat is a picture and part number for all parts in the Boost set.

Disclaimer

mike's_demo.py (continued)

```
    mode=TiltSensor.MODE_3AXIS_FULL, granularity=3)
153.     while demo_tilt_sensor_simple.cnt < limit:
154.         sleep(1)
155.     movehub.tilt_sensor.unsubscribe(callbackTilt)
156.
157. def callback_color(color, distance=None): # returns
    to nearest for distances > 1 inch
158.     global limit
159.     demo_color_sensor.cnt += 1
160.     correctColor = color
161.     if color == 5: # to correct for error in sensor
162.         correctColor = 6
163.     if distance != None:
164.        metric = distance * 2.45  # distance in mm
165.     else:
166.       metric = 0
167.     print('#%s/%s: Colour %s, distance %.2fmm' % (
    demo_color_sensor.cnt, limit, COLORS[correctColor],
    metric))
168.
169. def demo_color_sensor(movehub):
170.     global limit
171.     print("Colour & distance sensor test: wave your
    hand in front of it")
172.     demo_color_sensor.cnt = 0
173.     limit = 100 # number of times to take a reading
174.     try:
175.         movehub.color_distance_sensor.
    subscribe(callback_color)
176.     except:
177.        print("No colour & distance sensor found")
178.        sleep(2)
179.        return
180.     while demo_color_sensor.cnt < limit:
181.         sleep(0.5)
182.     # sometimes gives a warning - not sure why
183.     movehub.color_distance_sensor.unsubscribe(
    callback_color)
184.
185. def demo_motor_sensors(movehub):
186.     global resetTimeout, posA, posB, posE
187.     testTime = 5
188.     print("Motor rotation sensors test, move by hand
    any motor")
189.     print("Test ends after %s seconds with no change"
    % testTime)
190.     print()
191.     demo_motor_sensors.states =
    {movehub.motor_A: None, movehub.motor_B: None}
192.     resetTimeout = False ; external = False
193.     posA = 0 ; posB = 0 ; posE = 0
194.
195.     def callback_a(param1):
196.         global resetTimeout,posA
197.         last_a = demo_motor_sensors.states[movehub.motor_A]
198.         if last_a != param1:
199.             resetTimeout = True
200.         demo_motor_sensors.states[movehub.motor_A] = param1
201.         posA = param1
202.
203.     def callback_b(param1):
204.         global resetTimeout,posB
205.         last_b = demo_motor_sensors.states[
    movehub.motor_B]
206.         if last_b != param1:
207.             resetTimeout = True
208.         demo_motor_sensors.states[movehub.motor_B] = param1
209.         posB = param1
210.
211.     def callback_e(param1):
212.         global resetTimeout,posE
213.         last_e = demo_motor_sensors.states[
    movehub.motor_external]
214.         if last_e != param1:
215.             resetTimeout = True
216.         demo_motor_sensors.states[
    movehub.motor_external] = param1
217.         posE = param1
218.
219.     movehub.motor_A.subscribe(callback_a)
220.     movehub.motor_B.subscribe(callback_b)
221.     external = False
222.
223.     if movehub.motor_external is not None:
224.         demo_motor_sensors.states[
    movehub.motor_external] = None
225.         external = True
226.         movehub.motor_external.subscribe(callback_e)
227.
228.     timeOut = time.time() + testTime
229.     while timeOut > time.time():
230.       if resetTimeout :
231.          if external :
232.             print(
    "Motor A position %s \t Motor B position %s \t
    External Motor position %s" % (posA, posB, posE))
233.          else:
234.             print("Motor A position %s \t Motor B
    position %s " % (posA, posB))
235.          timeOut = time.time() + testTime
236.          resetTimeout = False
237.
238.     movehub.motor_A.unsubscribe(callback_a)
239.     movehub.motor_B.unsubscribe(callback_b)
240.
241.     if movehub.motor_external is not None:
242.         demo_motor_sensors.states[
    movehub.motor_external] = None
243.         movehub.motor_external.unsubscribe(callback_e)
244.
245. if __name__ == '__main__':
246.     main()
```