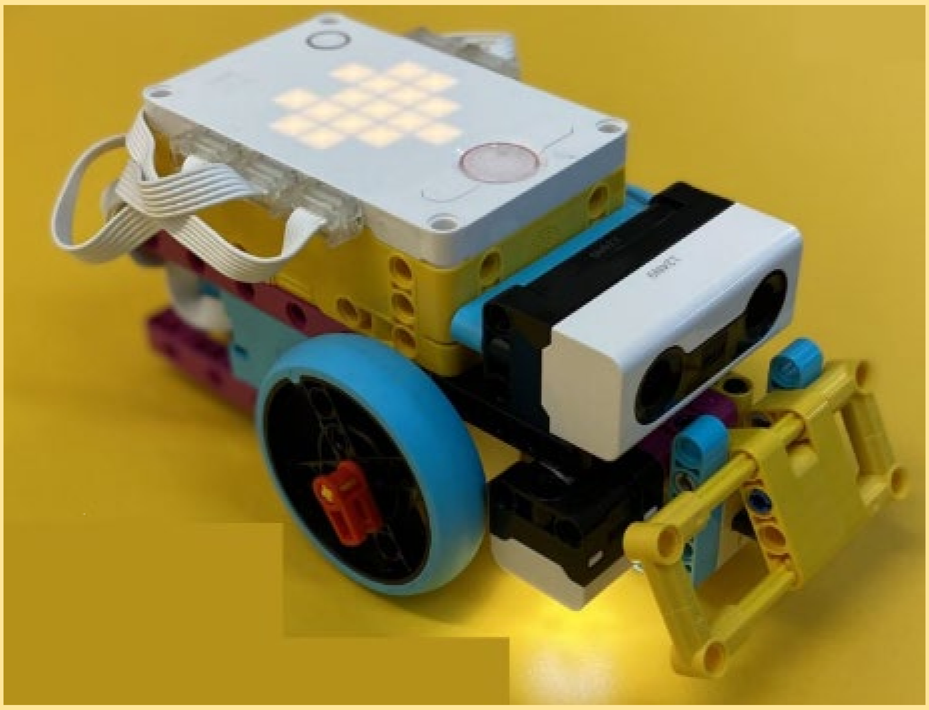
Digital Technology

Lego Spike

Python Coding



Version 2
2021

Barry Butler
bbutl58@eq.edu.au

ROBOCOAST
Sunshine Coast Robotics
www.robocoast.tech

Coolum State High School
Care • Respect • Excellence

Content

Section	Content
A	Introduction
B	Code Required for All Projects
C	Run the Motors in Set Patterns
D	Obstacle Avoidance with the Ultrasonic Sensor
E	Line Following with the Color Sensor
F	SumoBot

Documentation

https://hubmodule.readthedocs.io/en/latest/

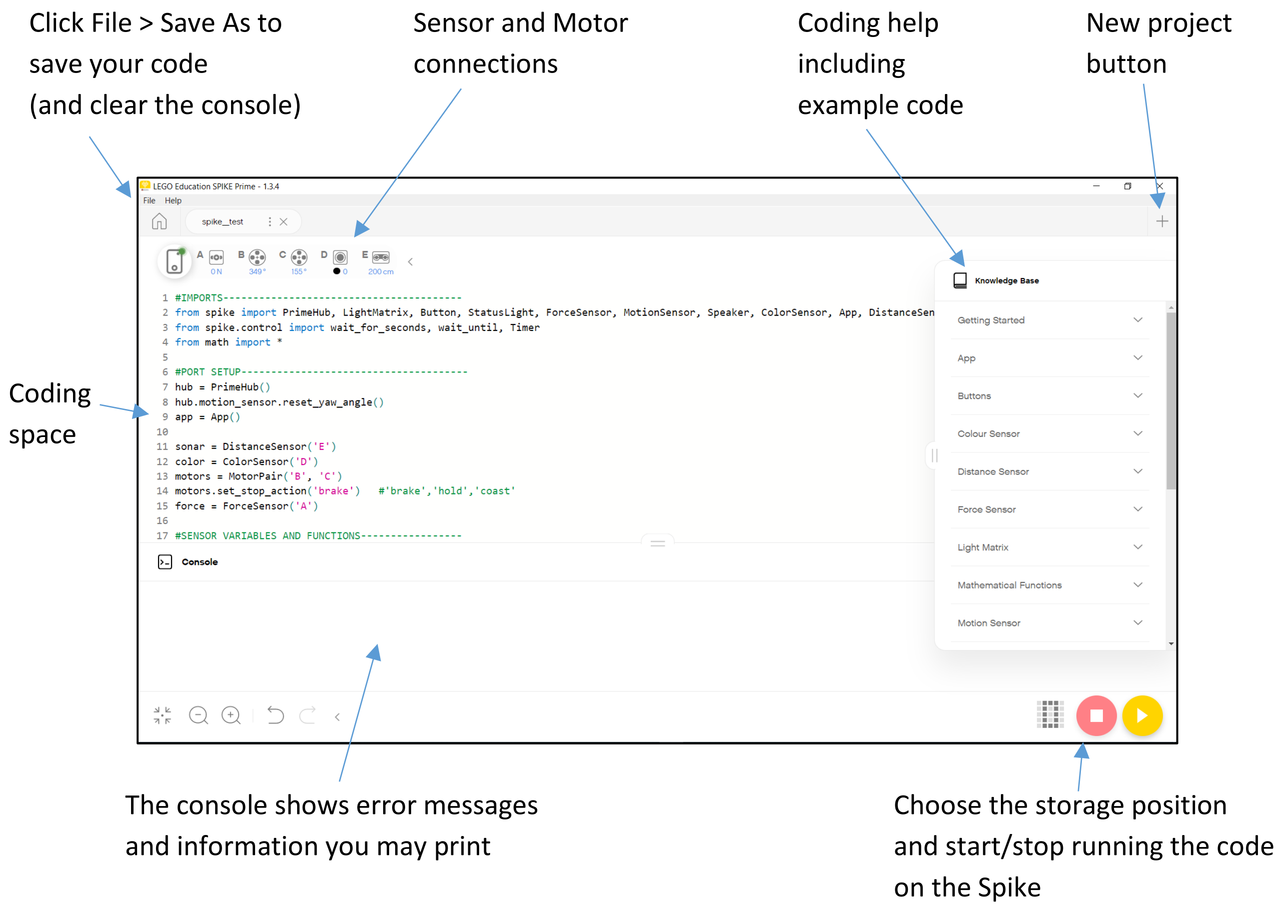
A. Introduction

To get started coding the Lego Spike:

1. Download the *LEGO Education SPIKE Prime* software from https://education.lego.com/en-us/downloads/spike-prime/software .

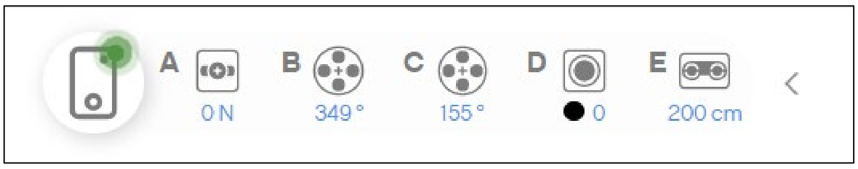
2. Run the software and connect the Spike using the USB cable. The Spike should be automatically recognised.

   The main software window has six main sections:


Click File > Save As to save your code (and clear the console)
Sensor and Motor connections
Coding help including example code
New project button
Coding space
The console shows error messages and information you may print
Choose the storage position and start/stop running the code on the Spike
LEGO Education SPIKE Prime - 1.3.4
File
Help
spike_test
A 0 N
B 349°
C 155°
D 0
E 200 cm
Knowledge Base
Getting Started
App
Buttons
Colour Sensor
Distance Sensor
Force Sensor
Light Matrix
Mathematical Functions
Motion Sensor
Console

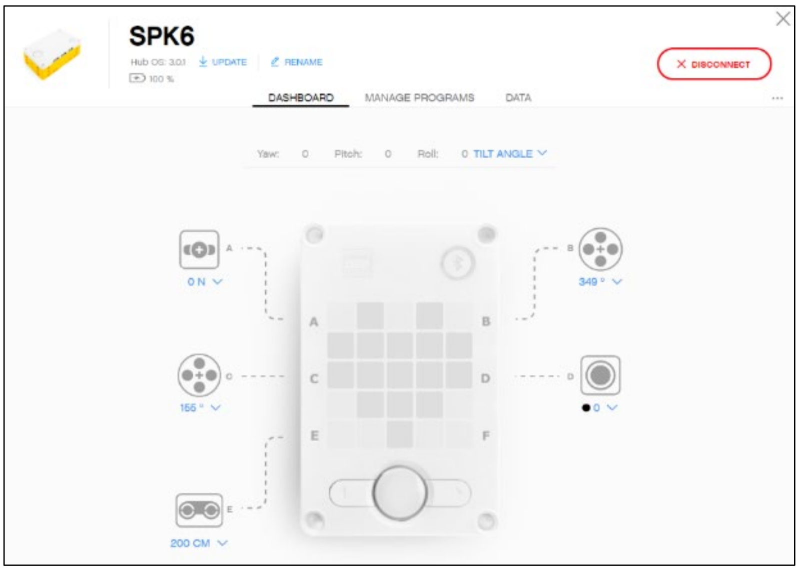
```
#IMPORTS---------------------------------------
from spike import PrimeHub, LightMatrix, Button, StatusLight, ForceSensor, MotionSensor, Speaker, ColorSensor, App, DistanceSen
from spike.control import wait_for_seconds, wait_until, Timer
from math import *

#PORT SETUP-------------------------------------
hub = PrimeHub()
hub.motion_sensor.reset_yaw_angle()
app = App()

sonar = DistanceSensor('E')
color = ColorSensor('D')
motors = MotorPair('B', 'C')
motors.set_stop_action('brake')   #'brake','hold','coast'
force = ForceSensor('A')

#SENSOR VARIABLES AND FUNCTIONS----------------
```

3. Check the connection of all the sensors and motors



4. Click the left button to change the sensor and motor settings.

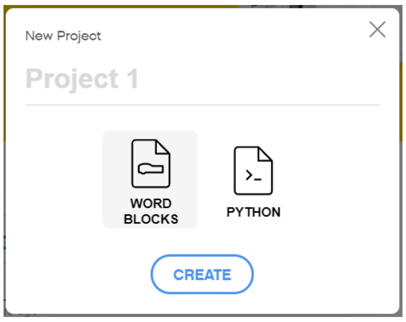
Click on each sensor or motor icon to change the setting of the sensor or motor.



5. Create a new project. Click **Home** and select **New Project**, or click the **+** button on the top right of the window.

Type in the name of the project and select **Python**.

Then click **Create.**



B. Code Required for All Projects

There is a set of instructions that is required for all projects. The code below should be modified for each project. If certain lines of code are not required (you may not be using a particular sensor or motor), just put a hash (#) in front of the line.

1. Import the Spike Libraries

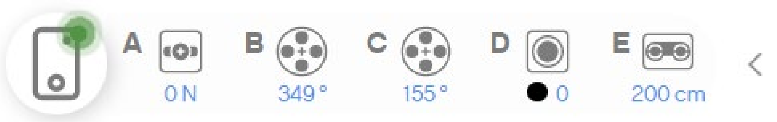
It is easiest with Spike to import all libraries, whether or not they will be used.

```
#IMPORTS---------------------------------------
from spike import PrimeHub, LightMatrix, Button, StatusLight, ForceSensor,
            MotionSensor, Speaker, ColorSensor, DistanceSensor, Motor, MotorPair
from spike.control import wait_for_seconds, wait_until, Timer
from spike.operator import *
from math import *
```

Don't import the App library unless you never want to run the program disconnected from the computer. An error will occur if you use the App library and don't have a connection.

2. Set up the Connection Ports

The connection to the Spike hub and all sensors and motors must be set up, using the correct ports. Make sure your code matches the list on the top of the screen.



Make sure you reset the yaw (rotation) angle and set the motor stop action here as well.

```
#CONNECTIONS---------------------------------------
hub = PrimeHub()
hub.motion_sensor.reset_yaw_angle()

sonar_sensor = DistanceSensor('E')
color_sensor = ColorSensor('D')
motors = MotorPair('B', 'C')
motors.set_stop_action('brake')   #'brake','hold','coast'
force_sensor = ForceSensor('A')
```

3. Get the Sensor Readings

It is easiest in most projects to get all sensor readings at once. This code creates global variables for each sensor value. The function ***get_all_values()*** can be called to get the current sensor values.

The line ***def get_sonar():*** is the start of a function. See how the line has a **colon** (:) at the end of it and the lines underneath are **indented**. Always use the **tab key** to create the indents for one line or many selected lines)

```
#SENSOR VARIABLES AND CONSTANTS-----------------
reflected = 0
distance = 999
switch = False
yaw = 0
pitch = 0

#SENSOR FUNCTIONS------------------------------
def get_sonar():
    d = 999
    value = sonar_sensor.get_distance_cm()
    if not value == None: d = value
    return d

def get_all_values(output):
    global reflected, distance, switch, yaw, pitch

    reflected = color_sensor.get_reflected_light()
    distance = get_sonar()
    switch = force_sensor.is_pressed()
    yaw = hub.motion_sensor.get_yaw_angle()
    pitch = hub.motion_sensor.get_pitch_angle()

    if output: print(reflected, distance, switch, yaw, pitch)
```

4. Functions to Turn Motors to an Angle

Spike Python has no built-in function to turn the motors until a yaw angle is reached. The functions in this section fill that gap.

```
#MOTOR FUNCTIONS--------------------------------
def wait_for_yaw(angle=90):
    yaw = 0
    if angle > 0:
        while yaw <= angle: yaw = hub.motion_sensor.get_yaw_angle()
    elif angle < 0:
        while yaw >= angle: yaw = hub.motion_sensor.get_yaw_angle()

def angle_turn(steer=100, speed=50, angle=90, stop=False):
    hub.motion_sensor.reset_yaw_angle()
    motors.start(steering=steer, speed=speed)
    wait_for_yaw(angle=angle)
    if stop: motors.stop()

def angle_turn_tank(left_speed=0, right_speed=50, angle=90, stop=False):
    hub.motion_sensor.reset_yaw_angle()
    motors.start_tank(left_speed, right_speed)
    wait_for_yaw(angle=angle)
    if stop: motors.stop()
```

5. Delay the Start of the Code

For most projects involving motors, you don't want the code to start running as soon as it is sent to the Spike. Change the value of the the ***wait_for_seconds()*** command, if you also want a longer delay before it starts running.

```
#WAIT TO START----------------------------------
print('waiting to start')
hub.status_light.on('blue')               #blue indicates program loaded and ready
hub.left_button.wait_until_pressed()      #press left button to start program
wait_for_seconds(0.5)
hub.light_matrix.show_image('HAPPY')      #program starting to run
hub.status_light.on('green')
```

Note: remove *hub.left_button.wait_until_pressed()* if you want the program to start immediately when you press the middle button. However, remember that there will be a delay after the middle button is pressed for the code to be loaded and ready to execute.

6. Main Loop

Most projects will have an endless loop. Most of your code will be written inside this loop.

```
#MAIN LOOP--------------------------------------
print('main loop')

while True:
    get_all_values(True)
```

7. Save the Code and Run the Code on the Spike

- Click the **File > Save As** menu item to save your code to a file (this will also clear and reset the console)
- Select the storage position
- Download and run the code on the Spike.
- Stop the code running whenever you like.

C. Run the Motors in Set Patterns

The two driving motors are linked as a motor pair. There are two ways to run the motors:

1. Steering – where the direction of travel is controlled by a steering value (zero straight ahead, negative to the left, positive to the right). A single power setting is given to control the speed. The commands are:

   To start and stop the motors using steering

   ```
   motors.start(steering=0, speed=50)
   motors.stop()
   ```

   To move the motors by a specific value ('cm', 'rotations', 'degrees', 'seconds')

   ```
   motors.move(2, 'seconds', steering=0, speed=50)
   ```

2. Tank-Drive – where the direction of travel is controlled by the amount of power given to each motor

   To start and stop the motors using tank-drive:

   ```
   motors.start_tank(100, -100)
   motors.stop()
   ```

   To tank-drive the motors by a specific value ('cm', 'rotations', 'degrees', 'seconds')

   ```
   motors.move_tank(10, 'cm', left_speed=25, right_speed=75)
   ```

Where to use start(), start_tank() and stop()

The start() and stop() commands are used where you start the motors, then **wait until a certain condition is met** before stopping the motors. For example:

- Start the motors, then drive forward until Spike is 10cm from a wall before stopping the motors
- Start the motors, then drive forward until the color sensor detects the color ‘red’

Where to use move() and move_tank()

These commands are used when you want to drive or turn for a specific **distance** (in cm, wheel rotations or degrees of axle turn) or **time** (seconds).

Let’s start by getting the Spike moving in some basic patterns:

1. Forwards and Backwards (for a number of rotations or a distance)
2. Turn Left or Right
3. Turn until an Angle is Reached
4. Drive in a square

C1. Run Motors Forwards and Backwards

We will first use the move() command to do this, as we want to drive for a specific distance or time.

a. Drive forward for 30cm, then back for 30cm, at 50% speed

```
#MAIN LOOP---------------------------------------
while True:
    motors.move(30, 'cm', steering=0, speed=50)
    motors.move(30, 'cm', steering=0, speed=-50)
```

b. Drive forward for 2 wheel-rotations, then back for 2 wheel-rotations, at 50% speed

```
#MAIN LOOP---------------------------------------
while True:
    motors.move(2, 'rotations', steering=0, speed=50)
    motors.move(2, 'rotations', steering=0, speed=-50)
```

c. Drive forward for 2 seconds, then back for 2 seconds, at 50% speed

```
#MAIN LOOP---------------------------------------
while True:
    motors.move(2, 'seconds', steering=0, speed=50)
    motors.move(2, 'seconds', steering=0, speed=-50)
```

d. Drive forward for half a wheel turn (180^o), then back for half a wheel turn, at 50% speed

```
#MAIN LOOP---------------------------------------
while True:
    motors.move(180, 'degrees', steering=0, speed=50)
    motors.move(180, 'degrees', steering=0, speed=-50)
```

We can do the same thing using the move_tank() command.

```
#MAIN LOOP---------------------------------------
while True:
    motors.move_tank(10, 'cm', left_speed=50, right_speed=50)
    motors.move_tank(10, 'cm', left_speed=-50, right_speed=-50)
```

Try moving by rotations and time yourself.

C2. Turn to the Left or Right

We can use either steering and tank-drive to turn.

In this example we use steering control to first steer to the right (positive value), then steer to the left (negative value).

```
#MAIN LOOP--------------------------------------
while True:
    motors.move(30, 'cm', steering=100, speed=50)
    motors.move(30, 'cm', steering=-100, speed=50)
```

We could use the tank-drive to do this as well.

- If the left_speed is slower than the right_speed, turn to the left
- If the right_speed is slower than the left_speed, turn to the right

```
#MAIN LOOP--------------------------------------
while True:
    motors.move_tank(20, 'cm', left_speed=20, right_speed=70)
    motors.move_tank(20, 'cm', left_speed=70, right_speed=20)
```

Work out the distance for a particular speed that turns 90^o using either steering or tank-drive. Which is best - using 'cm', 'rotations', or 'degrees' ?

C3. Turn until an Angle is Reached

With steering, we can use the ***angle_turn()*** function to turn until an angle is reached. To turn a right angle:

```
#MAIN LOOP--------------------------------------
while True:
    get_all_values(False)
    angle_turn(steer=100, speed=50, angle=90, stop=True)
    wait_for_seconds(3)
```

With tank-drive, we can use the ***angle_turn_tank()*** function.

```
#MAIN LOOP--------------------------------------
while True:
    angle_turn_tank(left_speed=50, right_speed=0, angle=90, stop=True)
    wait_for_seconds(3)
```

C4. Drive in a square

We repeat a loop four times, to drive forward and turn to the right.

```
#MAIN LOOP--------------------------------------
while True:
    for i in range(4):
        motors.move(40, 'cm', steering=0, speed=50)             #straight
        angle_turn(steer=100, speed=50, angle=90, stop=True)      #turn
```

D. Obstacle Avoidance with the Ultrasonic Sensor

The sequence of events we will repeat is:

- Drive forward
- If an obstacle is found within 10cm then stop
- Move backward 5cm
- Turn (any angle will do)

To do this we can use the wait_for_distance() command **or** get and test the distance ourselves

Use the wait_for_distance() Command

```
#MAIN LOOP----------------------------------------
while True:
    motors.start(steering=0, speed=50)                  #forward
    sonar.wait_for_distance_closer_than(10, 'cm')       #collision test
    motors.stop()                                       #stop
    motors.move(5, 'cm', steering=0, speed=-50)         #move back
    motors.move(10, 'cm', steering=100, speed=50)       #turn
```

Get and Test the Distance

```
#MAIN LOOP----------------------------------------
while True:
    get_all_values(True)
    if distance < 10:                                       #collision test
        motors.stop()                                       #stop
        motors.move(5, 'cm', steering=0, speed=-50)         #move back
        motors.move(10, 'cm', steering=100, speed=50)       #turn
    else:
        motors.start(steering=0, speed=50)                  #forward
```

Slow Down when Close to a Collision

```
#MAIN LOOP----------------------------------------
print('main loop')
while True:
    get_all_values(False)

    if distance < 10:                                       #collision test
        motors.stop()                                       #stop

        motors.move(5, 'cm', steering=0, speed=-50)         #move back
        motors.move(10, 'cm', steering=100, speed=50)       #turn
        motors.start(steering=0, speed=50)                  #forward

    elif distance < 20:
        new_speed = round(50 * (distance - 10)/10)          #ratio of speed required
        motors.start(steering=0, speed=new_speed)
    else:
        motors.start(steering=0, speed=50)                  #forward
```

E. Line Following with the Color Sensor

We have previously created a global variable to get the reflected light value from the color sensor. We can use that value to test whether the color sensor is on or off a black line.

- On a line will give a low reflectance value or off a line will give a high value.
- Assume for a start that if the reflected light value is less than 60% if we are on or near a black line.
- Place the Spike with the color sensor on the left of the black line
- If the sensor is on white – turn right
- If the sensor is on black – turn left

```
#MAIN LOOP--------------------------------------
motors.start(steering=0, speed=50)                        #forward

while True:
    get_all_values(True)
    if reflected > 60:
        motors.move(steering=30, speed=50)                #turn right
    else:
        motors.move(steering=-30, speed=50)               #turn left
```

F. SumoBot

SumoBots use the ultrasonic sensor to seek and destroy another robot vehicle in the Sumo ring, while using the color sensor to sense the white border and avoid falling off the edge.

F1. Basic Sumo Code

The basic actions of a SumoBot are:

- A three second wait before doing anything
- Move forward from the edge 20cm
- Rotate until the ultrasonic sensor locates the other vehicle (less than 80cm away)
- Drive full speed toward the other vehicle
- If the white edge is detected (high reflectance value) then stop, back up and rotate to locate the other vehicle

```
#MAIN LOOP--------------------------------------
print('main loop')
motors.move_tank(20, 'cm', left_speed=50, right_speed=50)

while True:
    get_all_values(False)

    if distance < 80:                           #other bot detected
        motors.stop()
        motors.start_tank(100, -100)            #forward full speed
    else:
        motors.start_tank(50, -50)              #rotate to locate

    if reflected > 60:                          #white line detected
        motors.stop()                           #stop, back and rotate
        motors.move_tank(20, 'cm', left_speed=-50, right_speed=-50)
        motors.start_tank(50, -50)
```

F2. Enhancements

- Don’t waste time moving forward at the start before starting to find the other vehicle
- Only scan left and right up to 90 degrees the first time
- Stop every 10 degrees when scanning to make sure scan detects vehicle (moving too fast doesn’t work) by using the movement sensor yaw and the angle-turn() function
- Use movement sensor to detect a collision or the bot lifted off the ground (pitch or roll)
- If motion is stopped for x seconds, use a series of rapid wheel movements (e.g. back and forth) to try and get free
- Use a different strategy:
    - Follow white line around the outside
    - Drive to a random place
    - Drive forward until white line and turn and go somewhere else until white line
- Use two ultrasonic sensors at different angles