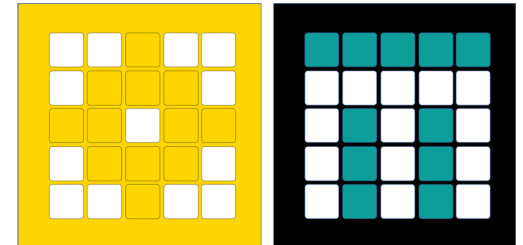


PRIME LESSONS

By the Makers of EV3Lessons



TURNING WITH THE GYRO

BY SANJAY AND ARVIND SESHAN

This lesson uses SPIKE 3 software

LESSON OBJECTIVES

Learn how to turn using the built-in motion sensor (gyro)

METHODS YOU NEED IN THIS LESSON

Motion Sensor methods – Used to read and reset the values of the gyro sensor

```
motion_sensor.tilt_angles()
```

This method returns a tuple containing 3 values. You can read more about Python Tuples in the Lists and Tuples lesson or at w3Schools [Python Tuples](#)

Each value in the tuple is in decidegrees (tenths of degrees). So to check for > 90 degrees you have to check a value of > 900.

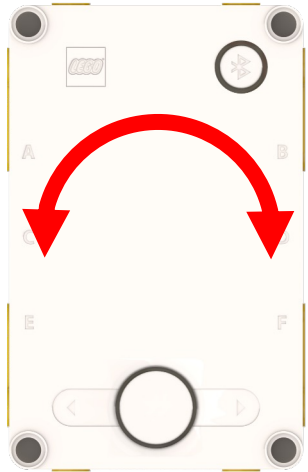
```
motion_sensor.reset_yaw()
```

```
motion_sensor.stable()
```

This method returns true when the sensor is resting flat

ROBOT ORIENTATION: YAW, PITCH AND ROLL

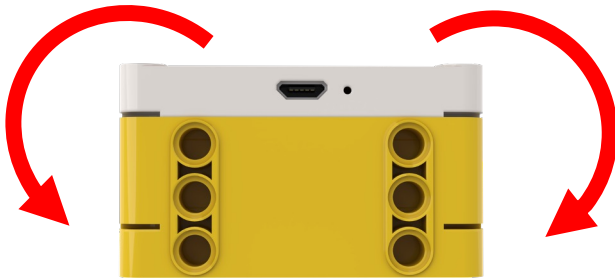
Yaw is turning the Hub to right or left



Pitch is turning the Hub up and down

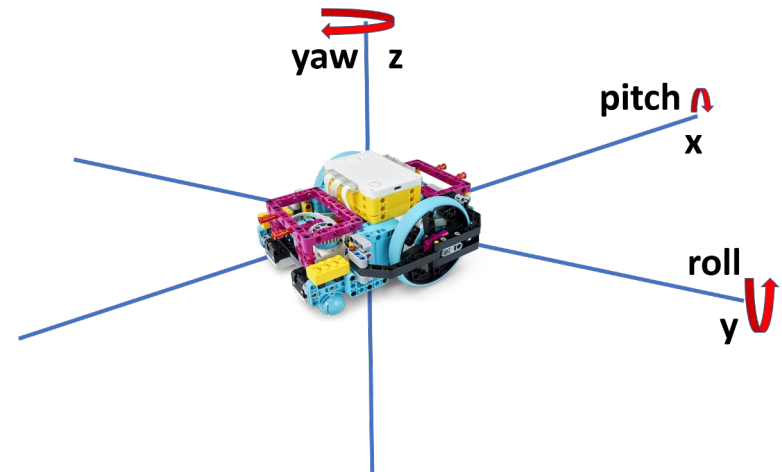


Roll is turning the Hub to side-to-side



Just like x, y and z coordinates are used to describe a robot's position, yaw, pitch and roll are terms used to describe a robot's orientation. Yaw is rotation around the z-axis. Pitch is rotation around y-axis. Roll is rotation around the x-axis.

The built-in Gyro Sensor can measure the robot's orientation



USING THE MOTION SENSOR (GYRO) TO TURN

The gyro sensor can be programmed to measure the hub's yaw, pitch and roll

These values can be used to sense if the robot has turned around x, y, or z axes

In this lesson, we will focus on yaw which can be used to determine if a robot has turned left or right

For pitch and roll, the robot uses gravity to determine what is a zero reading. Flat on the ground is 0 pitch and 0 roll.

For yaw, the robot doesn't have a compass to tell it what is north or south. Therefore, you need to tell the robot what it should consider zero. This is done with the `reset_yaw()` method.

```
motion_sensor.tilt_angles()  
motion_sensor.reset_yaw()
```

YAW ANGLE MEASUREMENTS

The Yaw value as seen in the app:

The yaw angle counts up from 0 (positive values) if the robot is turning right, and down from 0 (negative values) as the robot turns left

At the 180 degree mark, the signs change and there is a switch! The yaw reading as shown in the app will go from 179 to -180 if turning clockwise, and -180 to 179 if turning counterclockwise.

If you want to turn more than 180 degrees in a particular direction, you have to do some extra checks, explained later in the lesson.

The yaw value as read by the tilt_angles tuple:

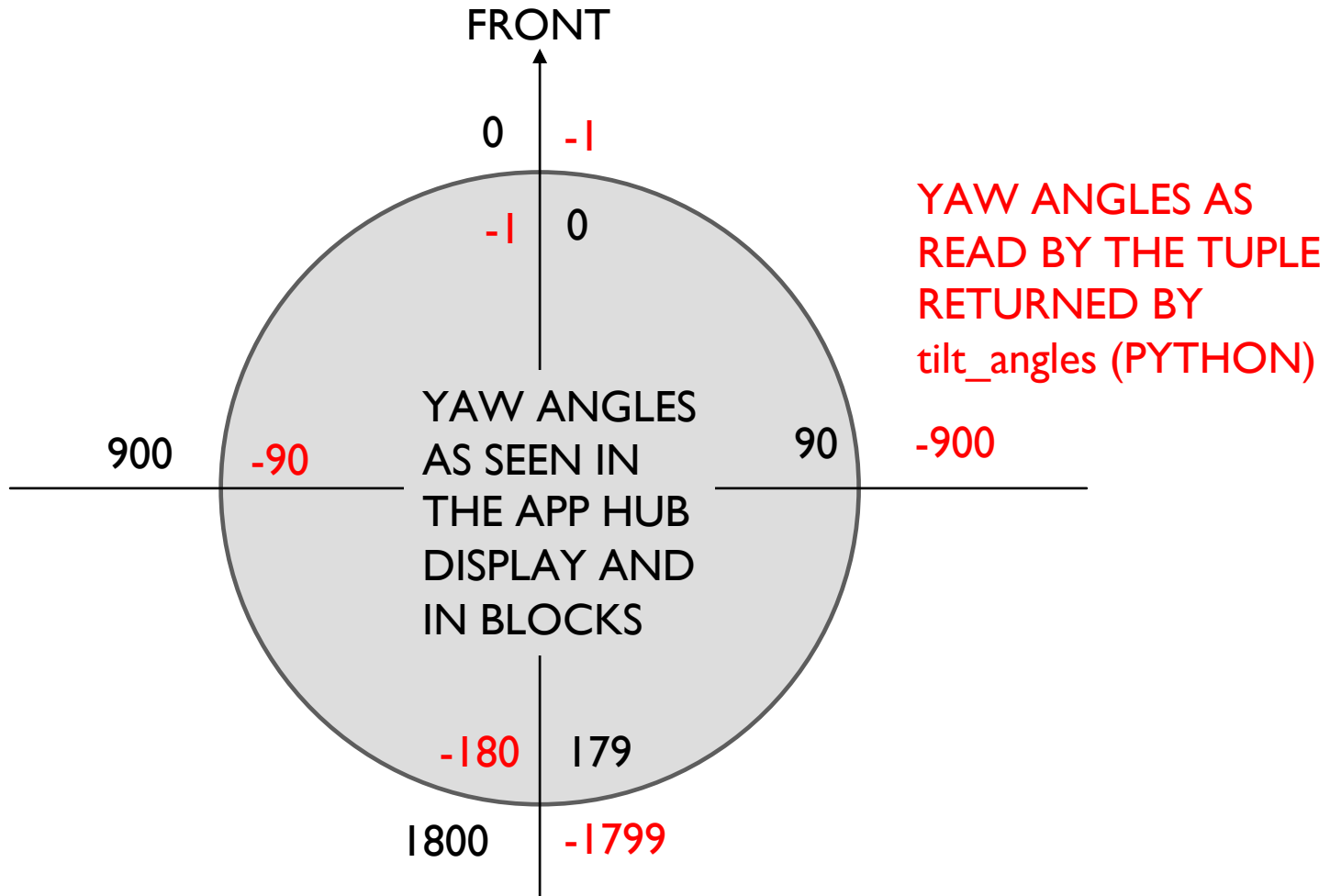
Motion sensor tuple readings are the **opposite** sign as the values shown as the yaw reading in the app and in Blocks, and in decidegrees (tenths of degrees)

When moving clockwise from 0, the readings go from 0 to -1799, then to 1800 and down to 0

Multiplying the tuple reading by -0.1 converts it into the same value as in the App/Blocks.

If your turns are limited to less than 180 degrees, use absolute values to avoid bugs.

YAW ANGLE MEASUREMENTS (GRAPHIC)



WAITING FOR THE GYRO TO REACH AN ANGLE

There are two options to measure if the robot has reached the desired angle

Option I: LEGO-specific API

Use the `runloop.until` function. It will wait until the function returns true.

The function cannot have any parameters. Use global variables as needed instead of parameters.

```
import runloop
await runloop.until(<function that returns true or false based on conditions/sensor readings>)
```

This option is easier to use and details can be found in the Spike Knowledge Base

Option II: General Python API

Use while loops

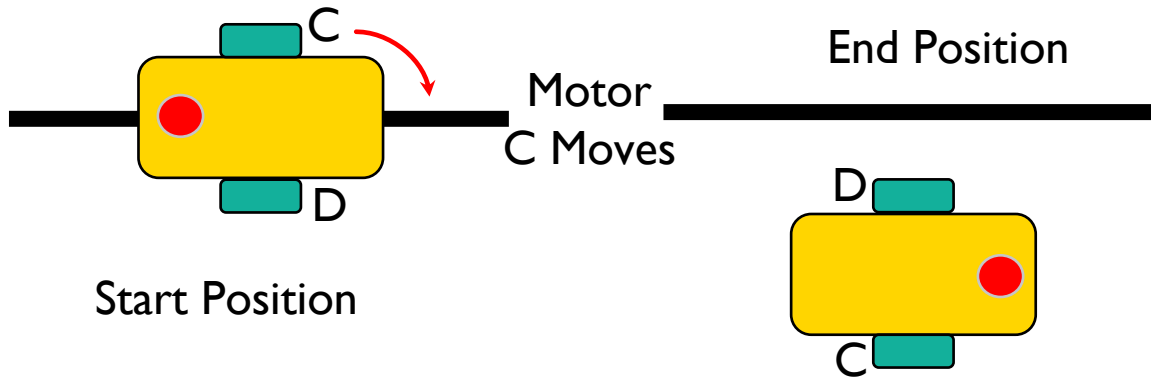
```
start moving....
while (motion_sensor.get_yaw_angle() < ANGLE):
    <code>
stop moving....
```

Easier for running code while waiting. You could also use a user defined `operator_function` in `wait_until()` – but a while loop makes the code clearer.

If you do not want to run code, you can place `pass` in place of `<code>` to skip the iteration of the loop

THERE ARE TWO TYPES OF TURNS YOU CAN DO

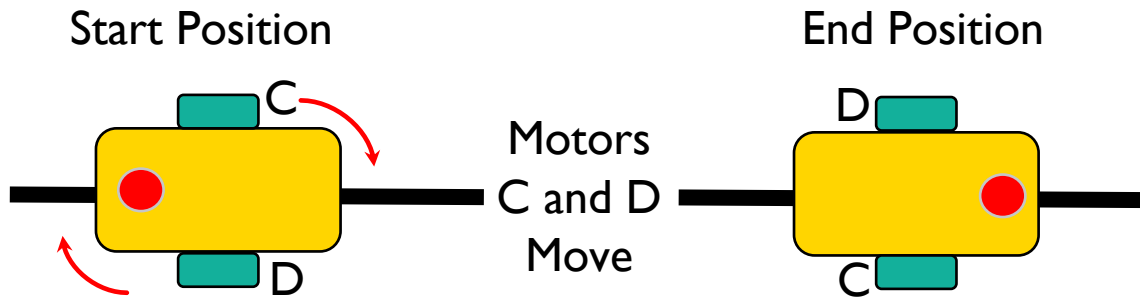
180 Degree Pivot Turn



Notice where the robot ends in both pictures after a 180 degree turn.

In the Spin Turn, the robot moves a lot less and that makes Spin Turns are great for tight positions. Spin turns tend to be a bit faster but also a little less accurate.

180 Degree Spin Turn



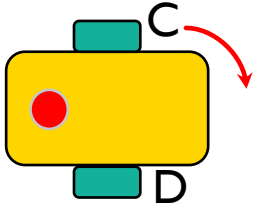
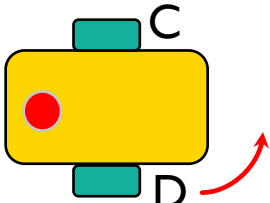
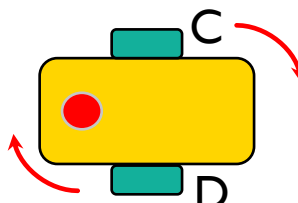
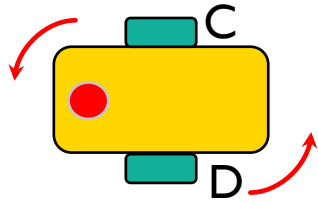
So when you need to make turns, you should decide which turn is best for you!

HOW TO MAKE PIVOT AND SPIN TURNS - I

Using move:

Change steering value here. A value of 0 moves straight

```
motor_pair.move(pair, steering)
```

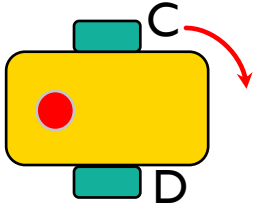
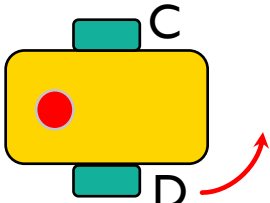
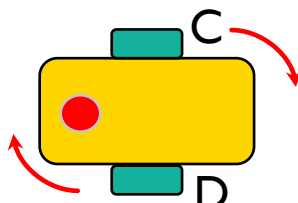
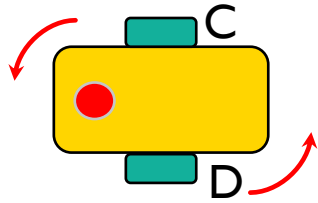
Steering Values			
50	-50	100	-100
			
Pivot Turn Right	Pivot Turn Left	Spin Turn Right	Spin Turn Left

HOW TO MAKE PIVOT AND SPIN TURNS - II

Using move tank:

Change velocity values here.
Same velocity values moves
straight

```
motor_pair.move_tank(pair, left_velocity, right_velocity)
```

Tank Velocity Values			
200, 0	0, 200	200, -200	-200, 200
			
Pivot Turn Right	Pivot Turn Left	Spin Turn Right	Spin Turn Left

CHALLENGE I

Write a program that turns 90 degrees to the right (clockwise) using a pivot turn

Basic Steps:

Define the motor pair

Reset yaw

Wait until motion sensor is stable

Start the motor pair using `move` or `move_tank` as you like.

Wait until the yaw has changed 90 degrees

Stop

NOTE: Always use `<` and `>` operators to compare sensor values! Never use `==` because the sensor may not be read at that exact value, and your loop will never stop.

CHALLENGE I SOLUTION

```
from hub import port, motion_sensor
import runloop, motor_pair, sys
```

```
# Function that returns true when the absolute yaw angle is 90 degrees
```

```
def turn_done():
```

```
    # convert tuple decidegree into same format as in app and blocks
```

```
    return abs(motion_sensor.tilt_angles()[0] * -0.1) > 90
```

```
async def main():
```

```
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)
```

```
    motion_sensor.reset_yaw(0)
```

```
    await runloop.until(motion_sensor.stable)
```

```
    # move with a steering of 50
```

```
    motor_pair.move(motor_pair.PAIR_1, 50, velocity=200)
```

```
    # Alternate using tank:
```

```
    # motor_pair.move_tank(motor_pair.PAIR_1, 200, 0)
```

```
    await runloop.until(turn_done)
```

```
    motor_pair.stop(motor_pair.PAIR_1)
```

```
    sys.exit("Done")
```

```
runloop.run(main())
```

CHALLENGE II (ADVANCED)

Write a program that turns up to 355 degrees, **clockwise or counterclockwise**, using a spin turn

If you are turning counterclockwise, the yaw angle tuple reading will count up from 0 until 1800, but then it becomes negative and counts down! The reverse is true if you are turning clockwise. See the graphic.

Important steps:

- Set the steering to positive if turning clockwise, and negative if turning counterclockwise

- Determine what the yaw reading must be when the robot has to stop, using the standard (not tuple) convention as seen in the app:

 - If turning clockwise check that the yaw has become negative, AND is GREATER than the stop angle (see the graphic and try to figure out why this math works)

 - If turning counterclockwise check that the yaw has become positive, AND is LESS than the stop angle (see the graphic and try to figure out why this math works)

- Convert the tuple reading into the standard yaw format (degrees, clockwise being positive and counterclockwise being negative) by multiplying it by -0.1 before using in any checks.

CHALLENGE II (ADVANCED) SOLUTION PAGE 1 OF 5

```
from hub import port, motion_sensor
import runloop, motor_pair, sys
```

```
# GLOBALS
```

```
# Set from -355 to 355. Positive numbers are clockwise.
```

```
degrees_to_turn = 0
```

```
# Yaw angle reading that indicates the robot needs to stop
```

```
stop_angle = 0
```

Set up a couple of globals to use in the program:

1. `degrees_to_turn`: degrees that the robot needs to turn. Positive is clockwise, negative is counterclockwise.
2. `stop_angle`: the yaw angle reading we are going to check against.

CHALLENGE II (ADVANCED) SOLUTION PAGE 2 OF 5

```
# Function that returns true when the yaw has turned past stop angle
def turn_done():
    global degrees_to_turn, stop_angle
    # convert tuple decidegree into the same format as in app and blocks
    yaw_angle = motion_sensor.tilt_angles()[0] * -0.1
    # if we need to turn less than 180 degrees, check the absolute values
    if (abs(degrees_to_turn) < 180):
        return abs(yaw_angle) > stop_angle

    # If we need to turn more than 180 degrees, compute the yaw angle we need to stop at.
    if degrees_to_turn >= 0: # moving clockwise
        # The adjusted yaw angle is positive until we cross 180.
        # Then, we are negative numbers counting up.
        return yaw_angle < 0 and yaw_angle > stop_angle
    else:
        # The adjusted yaw angle is negative until we cross 180
        # Then, we are positive numbers counting down.
        return yaw_angle > 0 and yaw_angle < stop_angle
```

Add a function that lets the runloop know when the wait condition is reached.

The function has different checks for:

1. Angles whose absolute values are less than 180 i.e. they don't cross the transition point
2. Angles that are more than 180 clockwise
3. Angles that are more than 180 counterclockwise

CHALLENGE II (ADVANCED) SOLUTION PAGE 3 OF 5

```
async def setupMotors():  
    motor_pair.pair(motor_pair.PAIR_1, port.C, port.D)  
    motion_sensor.reset_yaw(0)  
    await runloop.until(motion_sensor.stable)
```

Add a function that :

1. Sets up your motor pair
2. Resets the yaw angle to 0
3. Waits for the sensor to be stable

CHALLENGE II (ADVANCED) SOLUTION PAGE 4 OF 5

```
async def spinTurn(degrees):
    if abs(degrees) > 355:
        print("Out of range")
        return
    await setupMotors()
    global degrees_to_turn, stop_angle
    degrees_to_turn = degrees # set the global to use in the turn_done function
    # set the stop_angle global to use in the turn_done function
    if (abs(degrees) < 180):
        stop_angle = abs(degrees_to_turn)
    else:
        stop_angle = (360 - abs(degrees)) if degrees < 0 else (abs(degrees) - 360)
    # set the steering laue based on turn direction
    steering_val = 100 if degrees >= 0 else -100
    motor_pair.move(motor_pair.PAIR_1, steering_val, velocity=200)
    await runloop.until(turn_done)
    motor_pair.stop(motor_pair.PAIR_1)
```

Add a spinTurn that :

1. Inputs the degrees to turn and does some error checking for range
2. Sets up the motors and calculates the global variable values
3. Starts the motor pair moving with correct steering
4. Waits for the stop condition to be reached and then stops the motors

CHALLENGE II (ADVANCED) SOLUTION PAGE 5 OF 5

```
async def main():  
    await spinTurn(270)  
    sys.exit("Done")  
  
runloop.run(main())
```

Put it all together:

1. Main function calls spin turn with desired degrees (-355 to 355), waits for completion and exits
2. Runloop runs the main function

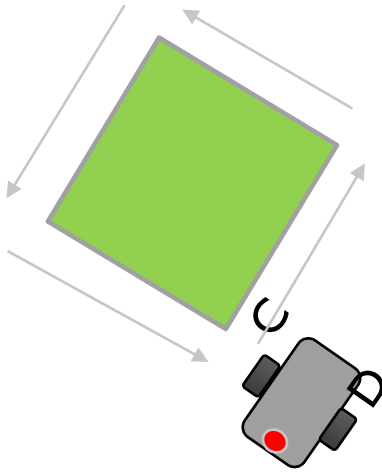
You can extend these concepts to write a pivot function that can pivot the robot -355 to 355. These are good functions to add to a library.

Note that yaw readings are not accurate at fast speeds. Slower turns give better results. If you need to, you can adjust the turn angle up if it is stopping too soon.

TURNING CHALLENGES

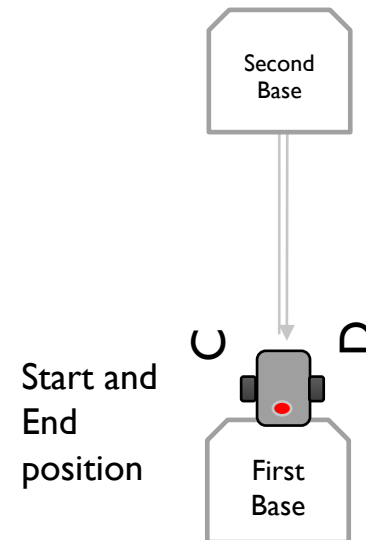
Challenge 1

- Your robot is a baseball player who has to run to all the bases and go back to home plate.
- Can you program your robot to move forward and then turn left?
- Use a square box or tape



Challenge 2

- Your robot baseball player must run to second base, **turn around** and come back to first.
- Go straight. Turn 180 degrees and return to the same spot.



CHALLENGE SOLUTIONS

Challenge 1

You probably used a combination of the `move()` method to go straight and do **pivot turns** to go around the box.

Challenge 2

You probably used a spin turn because it is better for tighter turns and gets you closer to the starting point!

CREDITS

This lesson was created by Sanjay and Arvind Seshan for Prime Lessons

Additional contributions by FLL Share & Learn community members.

More lessons are available at www.primelessons.org



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).