Worksheet 13
Harry: Junctions

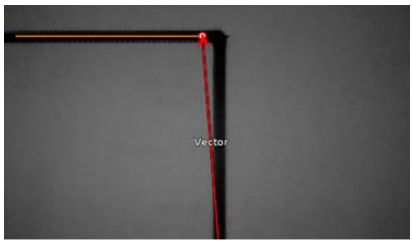
Learning Outcome 1,3

Book Chapter 1.24 – 1.26

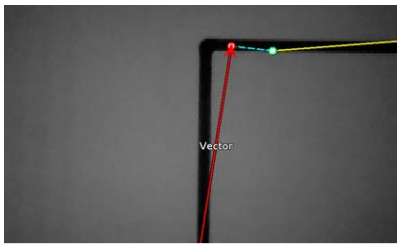
Purpose and API

Here we shall explore how to get Harry, the Stepper robot, to recognize corners and junctions between straight lines. Typical results from a left, right and tee junction are shown on the right.

Note that there is an array **Angle[]** with 4 possible elements, each for a junction line. For a left and right junction, there are two non-zero angles, and for a tee junction there are three.




```
Angle[0] = -176
Angle[1] = 90
Angle[2] = 0
Angle[3] = 0
```




```
Angle[0] = -90
Angle[1] = 177
Angle[2] = 0
Angle[3] = 0
```


```
Angle[0] = -90
Angle[1] = 90
Angle[2] = 180
Angle[3] = 0
```

---

1. Getting to know the junction API

(a) Draw on paper the four possible junction types and investigate. I suggest each line is 5cm long.

(b) Connect Pixymon to the camera, but do not connect the motor power jack.
Run the sketch **CBP_2403_R_Junction_Test_1**. The angles of the junction lines are sent to the serial monitor. You can pause the autoscroll on the monitor to freeze the display. Each vector will be assigned a unique color.

(c) Make sure you understand the angles returned. These should agree with the axes given in the previous worksheet. Since the forward direction is 0 degs, then an angle like -179 degrees shows the line you are coming from.

---

2. Reporting Junctions

(a) Open sketch **CBP_2403_R_Junction_Test_2.** This uses a library function **detectCorners(float angles);** to return whether the junction is a TEE, LEFT, or RIGHT (declared in the library files)

(b) Open up the library **sketchbook > libraries > CBPFBOStepperA.cpp** and look for this function. Make sure you understand what it does.

(c) Upload and run the sketch, but do not connect the motor power jack. Fire up the Serial Monitor to observe the reported junction type, and if LEDs are available on the board, they will also report the junction type. This will be useful in the field.

3. Detect, Advance and Pivot

Here you will need to use the following functions.

straightLine(dist, vMax, false, true);
Moves a distance 'dist' (mm) at speed vMax, does not ramp at start of move, but does ramp at end.
pivot(-90,vMax,false,false);
Pivots -90 degrees at speed vMax, does not ramp at beginning, does not ramp at end of pivot

(a) Open the sketch **CBP_2403_R_JunctionReact_1a** Look for the variable **juncAdvance**. This should be set to the distance (in mm) between the robot axle and the centre of the Pixycam. Update this variable.

(b) Add code to advance the robot when it detects a junction.

- test if there is a junction (they have ID's > 0)
- call **straightLine(…)** to move the robot **juncAdvance – advance** with no ramp on start, but ramp on the end.

Test it out on a long straight with a LEFT or RIGHT junction.

(c) Tweak the value of **juncAdvance** to get it working accurately.

(d) After the 1 second delay, add a line of code to get the robot to pivot 90 degs so it is aligned with your junction. Best to ramp up the speed at the start of the pivot.

(e) You might need to tweak the angle sent to **pivot(…)**.