Rule booklet for Maze-Runner:-

1. Participants are expected to complete solve a real time maze by building a robot of their own.
2. Then the map of this solved maze should be generated by the robot,which will be the solution of the maze.
3. The dimensions of the maze will be given many days prior to the event
4. Participants are expected to keep the dimensions of their robot within the limits of maze dimensions.
5. The choice of hardware(microcontroller,etc) is upon the participant . But raspberry-pi boards are suggested
6. In case of the tie-the participant with the most accurate map,and time taken to solve the maze will be considered as the tie breaker.

Maze dimensions:-

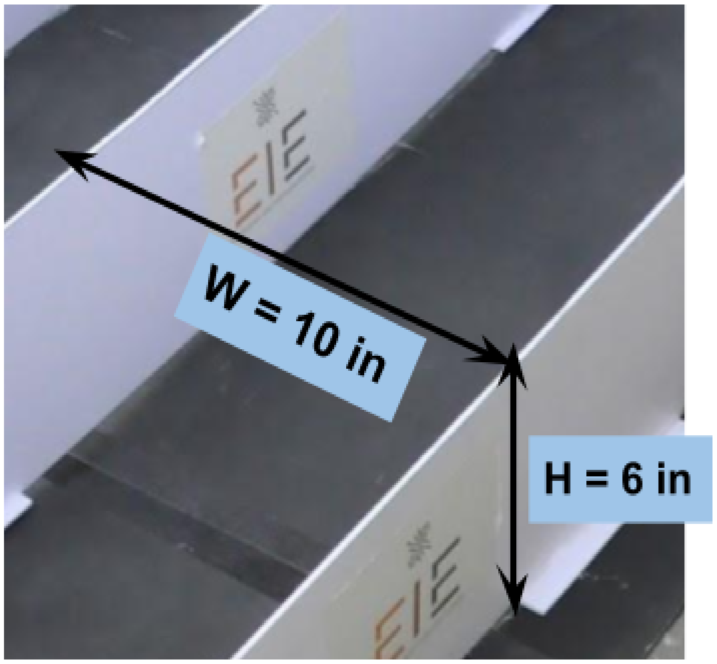


Fig-1

1. The wall-to-wall distance or **width** is **10 inches** as shown in fig-1
2. A wall **height** of **6 inches** as shown in fig-1.

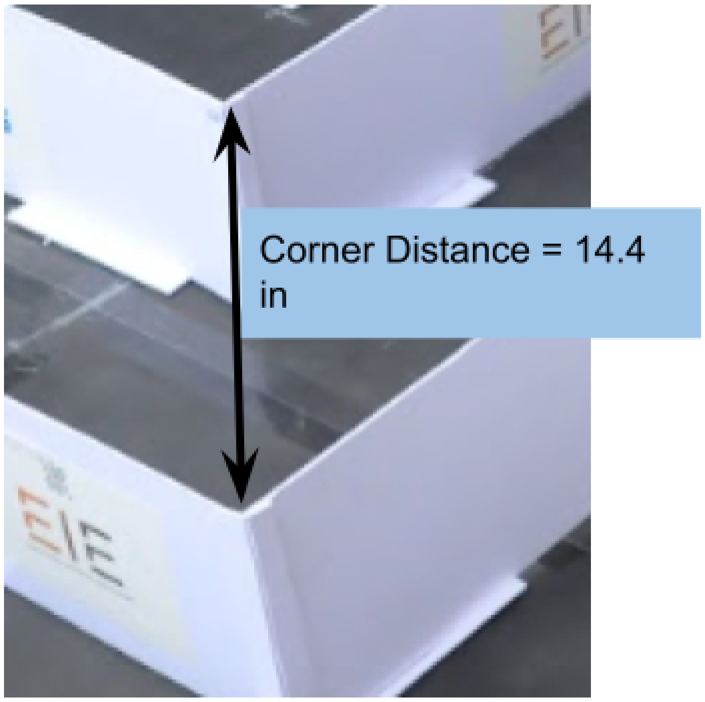


Fig-2

3. A corner to **corner** distance of **14.14 inches approx** as shown in fig-2

Sample Algorithm Requirements:-

1. Keep robot at start point.
2. Move forward (relative north). Check distances on front, right and left. record on map subroutine.
3. Move forward until the turning point i.e, arriving at a dead end in the relative north.
    - **Note:** the participants have to stop well prior to reaching a dead end. Alternative turning points which can be opening on the right or left, which provides alternate maze path and record on map subroutine.

4. Choose a path if only one path then there is no other choice. If multiple paths, create an check point and choose one path.
5. Record on map subroutine.
6. Go on a new path and map the paths chosen. If any path results in a dead end, then make u-turn and retrace to the last checkpoint. Choose alternate paths and repeat till the maze is complete.
7. Publish the complete map.
8. Publish right and wrong paths chosen by algorithm.

Important Notes:-

- Ideally maze runner robot should not repeatedly go to a path that has been identified as a dead end.
- Camera or Ultrasonic sensor can be chosen.
- Students can come with their robot with programming done and ready for demonstration.
- Students cannot preprogram a path (fluke chance or known path) and such programs will be immediately disqualified.

Sample Mapping Algorithm:-

- Note :The below code is shows a method of plotting using matplotlib module(You are free to use any methodology you find comfortable like turtlesim etc)
- Please note you need to merge this code with the code used by your robot to run the after sensor based decision making (You can reduce or increase steps to match your speed)

```
from matplotlib import pyplot as plt
#*********************GLobal Variables******************************
x=[0]
y=[0]
xref='+'
yref='+'
```

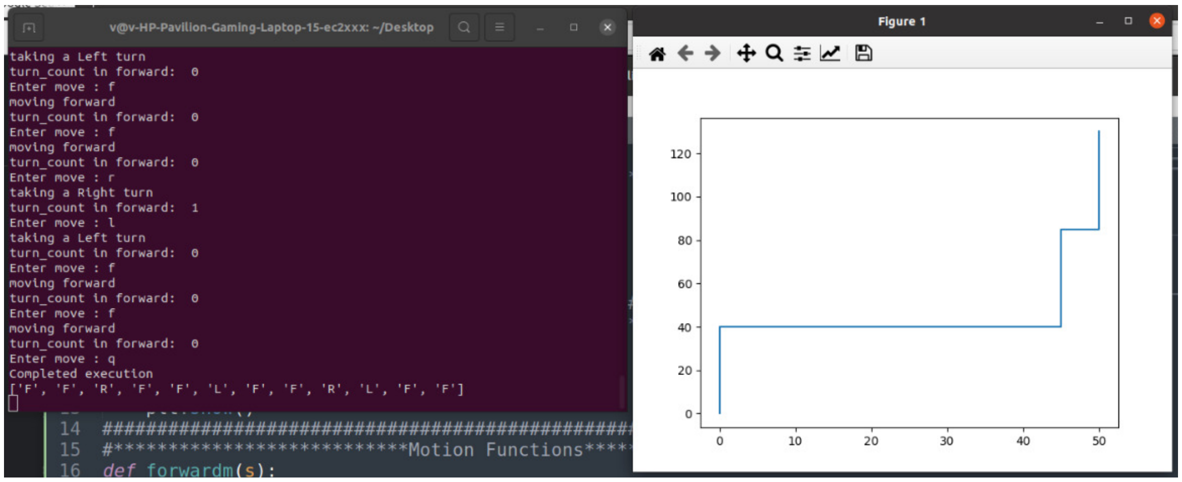
```python
i=None
###################################################################
#**************************Plotting Function**********************
def plotNshow():
    global x,y
    plt.plot(x,y)
    plt.show()
##################################################################
#***************************Motion Functions***********************
def forwardm(s):
    # s indicated the number of steps
    global x,y,xref,yref,turn_count
    if turn_count>=4:
        turn_count=turn_count-4
    elif turn_count<=-4:
        turn_count=turn_count+4
    print('turn_count in forward: ',turn_count)
    if turn_count==0:
        xref='+'
        yref='+'
    elif turn_count==-1:
        xref='-'
    elif turn_count==-2:
        yref='-'
    elif turn_count==-3:
        xref='+'
    elif turn_count==1:
        xref='+'
    elif turn_count==2:
        yref='-'
    elif turn_count==3:
        xref='-'
    if abs(turn_count)==0 or abs(turn_count)==2:
        if yref=='+':
            y.append(int(y[len(y)-1]+s))
            x.append(int(x[len(x)-1]))
        if yref=='-':
            y.append(int(y[len(y)-1]-s))
            x.append(int(x[len(x)-1]))
    if abs(turn_count)==1 or abs(turn_count)==3:
        if xref=='+':
            x.append(int(x[len(x)-1]+s))
            y.append(int(y[len(y)-1]))
        if xref=='-':
            x.append(int(x[len(x)-1]-s))
            y.append(int(y[len(y)-1]))

#*********************Main Function*******************************
if __name__=="__main__":
    turn_count=0
    moves=[]
    print('Move Key')
    print('F- Forward')
    print('R- Right')
    print('L- Left')
    print('Q- quit')
    while True:
        i=str(input('Enter move : '))
        if str.lower(i)=='f':
            print('moving forward')
            moves.append(str.upper(i))
            forwardm(20)
        elif str.lower(i)=='r':
            print('taking a Right turn')
            moves.append(str.upper(i))
            turn_count=turn_count+1
            forwardm(5)

        elif str.lower(i)=='l':
            print('taking a Left turn')
            moves.append(str.upper(i))
            turn_count=turn_count-1
            forwardm(5)

        elif str.lower(i)=='q':
            print('Completed execution')
            print(str(moves))
            plotNshow()
            exit()
        else:
            print('invalid input')

###################################################################
```



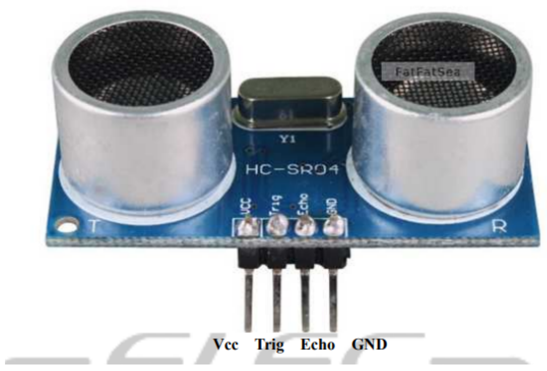
```
taking a Left turn
turn_count in forward:  0
Enter move : f
moving forward
turn_count in forward:  0
Enter move : f
moving forward
turn_count in forward:  0
Enter move : r
taking a Right turn
turn_count in forward:  1
Enter move : l
taking a Left turn
turn_count in forward:  0
Enter move : f
moving forward
turn_count in forward:  0
Enter move : f
moving forward
turn_count in forward:  0
Enter move : q
Completed execution
['F', 'F', 'R', 'F', 'F', 'L', 'F', 'F', 'R', 'L', 'F', 'F']
```

```
##################################################
#**************************Motion Functions****
def forwardm(s):
```

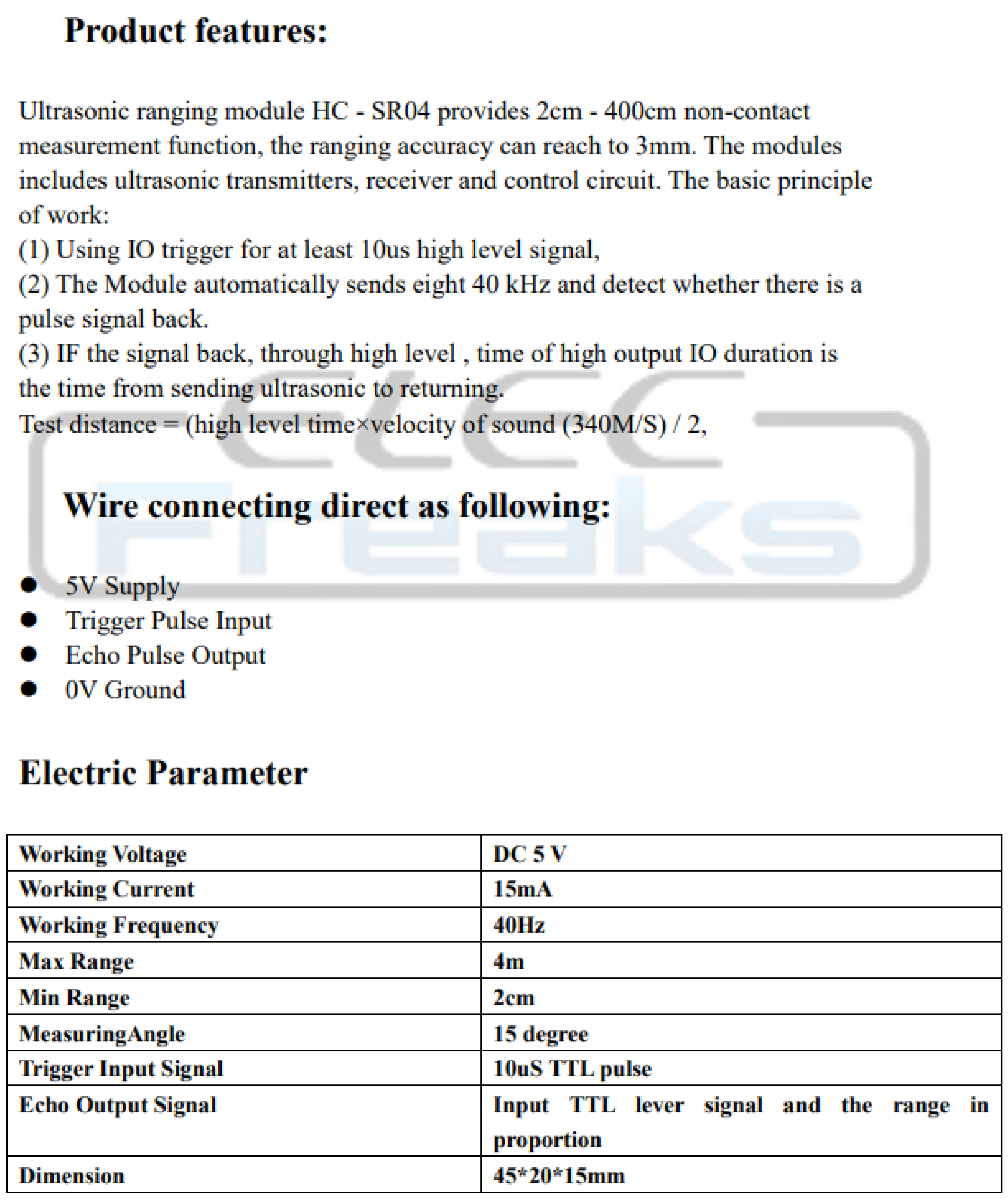
A Sample output of mapping algorithm

Sensor Requirements:-

- For this project,the preferred sensor would be a HC-SR04 ultrasonic sensor or the pi camera module. The details of these sensors are attached below:



Specification sheet:



Product features:

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit. The basic principle of work:
(1) Using IO trigger for at least 10us high level signal,
(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
(3) IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning.
Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

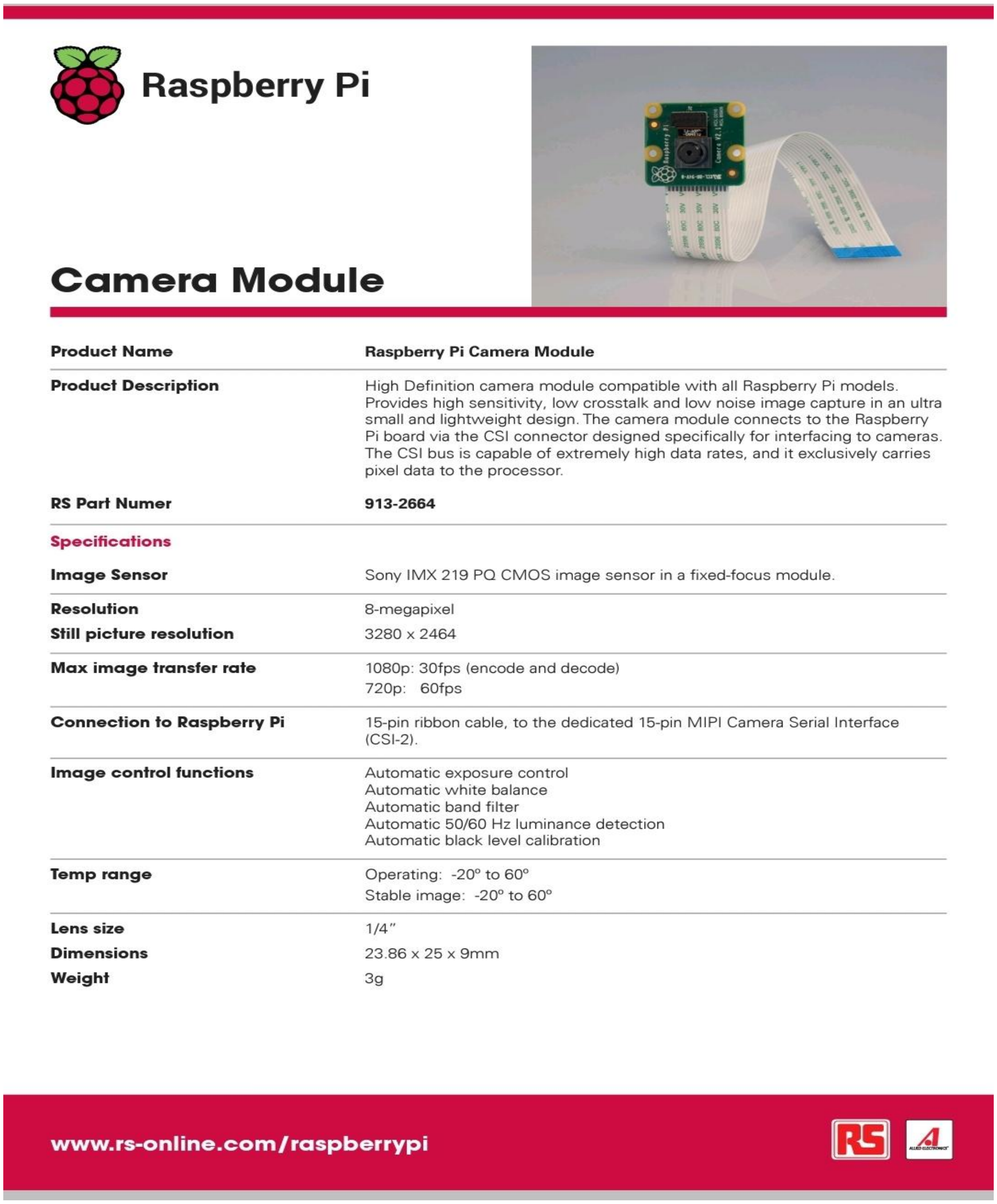
- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter

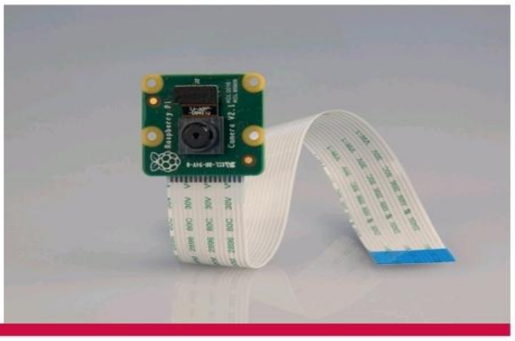
Working Voltage	**DC 5 V**
Working Current	**15mA**
Working Frequency	**40Hz**
Max Range	**4m**
Min Range	**2cm**
MeasuringAngle	**15 degree**
Trigger Input Signal	**10uS TTL pulse**
Echo Output Signal	**Input TTL lever signal and the range in proportion**
Dimension	**45*20*15mm**

*credits:https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf

2) Pi- camera module:-



Raspberry Pi



Camera Module

Product Name	**Raspberry Pi Camera Module**
Product Description	High Definition camera module compatible with all Raspberry Pi models. Provides high sensitivity, low crosstalk and low noise image capture in an ultra small and lightweight design. The camera module connects to the Raspberry Pi board via the CSI connector designed specifically for interfacing to cameras. The CSI bus is capable of extremely high data rates, and it exclusively carries pixel data to the processor.
RS Part Numer	**913-2664**
Specifications	
Image Sensor	Sony IMX 219 PQ CMOS image sensor in a fixed-focus module.
Resolution	8-megapixel
Still picture resolution	3280 x 2464
Max image transfer rate	1080p: 30fps (encode and decode) 720p: 60fps
Connection to Raspberry Pi	15-pin ribbon cable, to the dedicated 15-pin MIPI Camera Serial Interface (CSI-2).
Image control functions	Automatic exposure control Automatic white balance Automatic band filter Automatic 50/60 Hz luminance detection Automatic black level calibration
Temp range	Operating: -20° to 60° Stable image: -20° to 60°
Lens size	1/4"
Dimensions	23.86 x 25 x 9mm
Weight	3g

www.rs-online.com/raspberrypi