Checkpoint #3 Obstacle Avoidance

Purpose:

The purpose of this checkpoint is to make sure you can control your robot to move in the arena. The mobile robot needs to detect an obstacle in front of it and take action to avoid the obstacle in order to continue its motion.

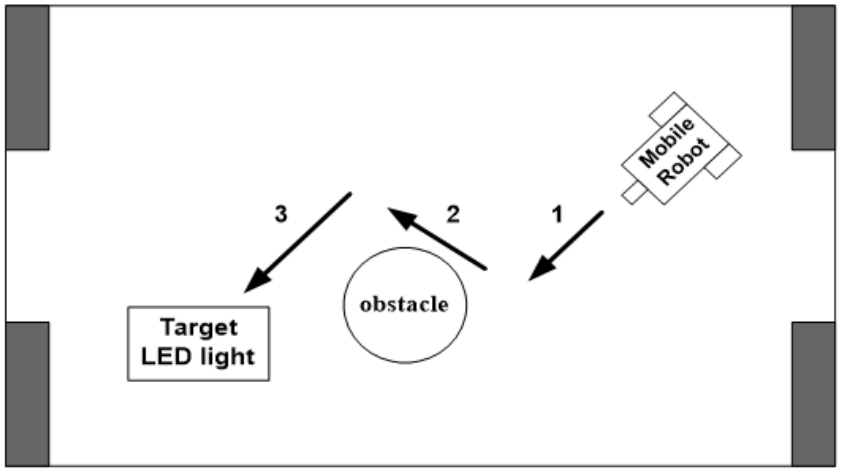
Finally, your robot can find the assigned target. In this checkpoint, the target is a ring of LED lights.

Tasks:

Please demonstrate your robot performing the following actions:

1. Please start to arrange the space configuration of your robot, make sure every and each component such as circuit boards and sensors is settled firmly and stable on the chassis and all the robot functions will not be affected by wires. **(15%)**
2. Make sure that your robot can move freely. It means that you do not need to use keyboard to control it anymore. **(20%)**
3. Integrate a light sensor and two touch sensors to the robot and program your robot to find and move toward the LED light. **(30%)**
4. The time to find the LED light. (in 90 sec). **(35%)**

Arena:



- **Materials list:**

	Material	Number
1	Photo resistor light sensor	1
2	Touch sensors	3
3	Resistances	3
4	Breadboard	1

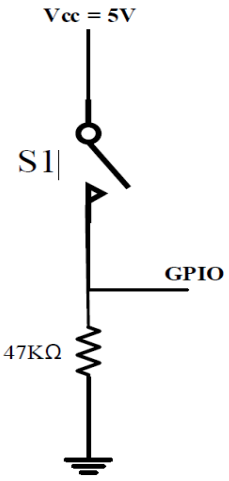
- **Photo resistor light sensor:**

Use Photo resistor sensor to detect the LED light.

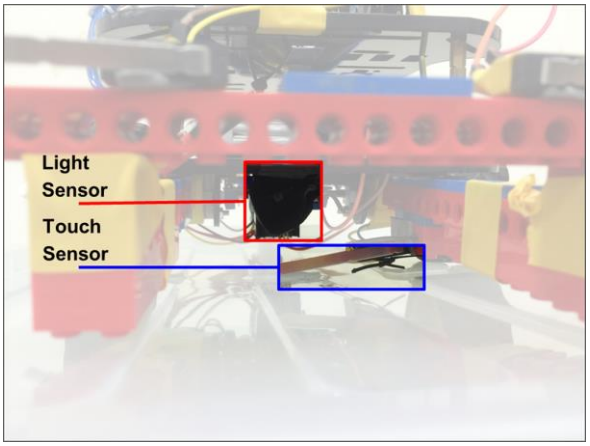
1. V_{cc} connect to Pi3's 5V.
2. GND connect to Pi3's GND.
3. D_0 connect to GPIO Pin.
4. You can change the Variable Resistor to increase the sensitivity of the sensor. If the brightness is bright enough, D_0 will be 0

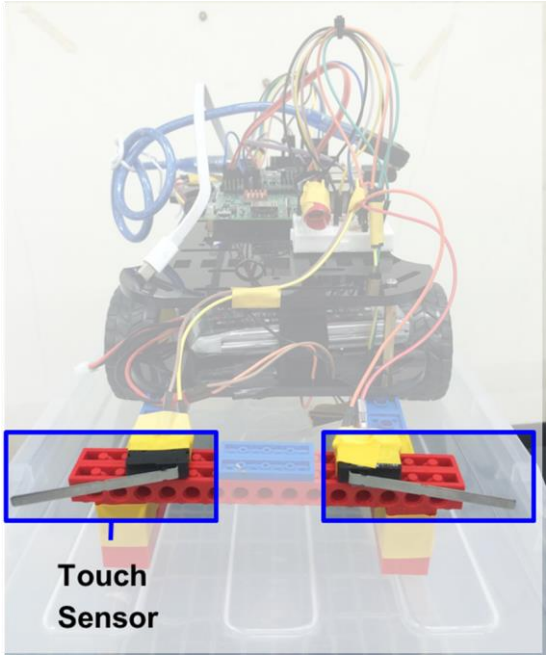
- **Touch sensor:**

Integrate touch sensors to avoid an obstacle and walls of the area.



● Example Hardware Configuration





● GPIO pin

Reference：https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/pinmappings/pinmappingsrpi

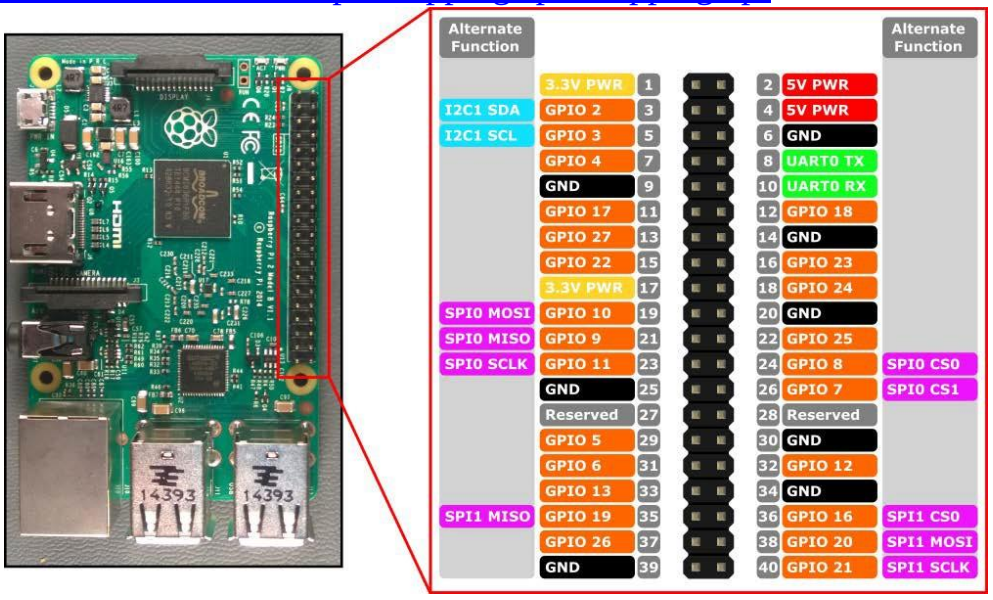


Figure 1 Hardware interfaces for the Raspberry Pi 2 and Raspberry Pi 3

● Wiring Pi

WiringPi is an GPIO interface library for the Raspberry Pi and it attempt to bring Arduino-wiring-like simplicity to the Raspberry Pi.The goal is to have a single common platform and set of functions for accessing the Raspberry Pi GPIO across multiple languages.

1. Test wiringPi's installation

- **Run the gpio command to check the installation.**

```
$ gpio -v
```

Ps. If it's not working, you have to reinstall.

Installation steps Reference：http://wiringpi.com/download-and- install/

- **To install**

```
$ gpio -v
```

If you get something, then you have it already installed. You will need to purge the package first.

• C version:

```
$sudo apt-get purge wiringpi
```

• Python version:

```
$sudo pip install wiringpi2
```

If you do not have GIT installed, you can install it with：

```
$sudo apt-get update
$sudo apt-get install git
```

To obtain WiringPi using GIT:

```
$ cd
$ git clone git://git.drogon.net/wiringPi
$ cd ~/wiringPi
$ git pull origin
```

The new build script will compile and install it all for you.

```
$ ./build
```

Check the WiringPi have already installed.

```
$ gpio -v
```

2. Check WiringPi's Pin number

Prints a table of WiringPi's Pin number on terminal.

```
$ gpio readall
```

P1: The Main GPIO connector							
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin
		3.3v	1	2	5v		
8	Rv1:0 - Rv2:2	SDA	3	4	5v		
9	Rv1:1 - Rv2:3	SCL	5	6	0v		
7	4	GPIO7	7	8	TxD	14	15
		0v	9	10	RxD	15	16
0	17	GPIO0	11	12	GPIO1	18	1
2	Rv1:21 - Rv2:27	GPIO2	13	14	0v		
3	22	GPIO3	15	16	GPIO4	23	4
		3.3v	17	18	GPIO5	24	5
12	10	MOSI	19	20	0v		
13	9	MISO	21	22	GPIO6	25	6
14	11	SCLK	23	24	CE0	8	10
		0v	25	26	CE1	7	11
WiringPi Pin	BCM GPIO	Name	Header		Name	BCM GPIO	WiringPi Pin

Figure 2 WiringPi's pin

3. WiringPi with ROS

If you can run this example code successfully, your program will display 1 when your light sensor to detect the light.

- **Example cpp code**

```
1  #include "ros/ros.h"
2  #include <wiringPi.h>
3  #include <iostream>
4  #include <std_msgs/Int16.h>
5
6  //light receive pin 3
7  const short int lightpin = 3;
8
9  ros::Time previous_time; ros::Time current_time;
10
11 int main (int argc, char **argv){
12     ros::init(argc, argv, "light_receive_data");
13     ros::NodeHandle n;
14     ros::Publisher light_pub =
15         n.advertise<std_msgs::Int16>("light_data", 1);
16
17     unsigned short int light_rev = 0;
18     std_msgs::Int16 light_data;
19
20     //use this command whithout sudo
21     setenv("WIRINGPI_GPIOMEM", "1", 1);
22
23     //library setup function wiringPiSetup ();
24     pinMode (lightpin, INPUT);
25
26     //10hz
27     ros::Rate loop_rate(10);
28     while(ros::ok()){
29         light_rev = digitalRead(lightpin) ;
30         light_data.data = light_rev;
31         ROS_INFO("light_receive : %d ", light_rev);
32         light_pub.publish(light_data);
33         ros::spinOnce();
34         loop_rate.sleep();
35     }
       return 0 ;
   }
```

- **Example cmake code**

```
cmake_minimum_required(VERSION 2.8.3)
project(light_receive_data)

find_package(catkin REQUIRED COMPONENTS roscpp
rospy std_msgs
)
FIND_LIBRARY(WIRINGPI_LIBRARY wiringPi
/usr/local/include)


catkin_package(
CATKIN_DEPENDS roscpp rospy std_msgs
)

include_directories(
${catkin_INCLUDE_DIRS}
)

add_executable(lightreceivedata src/light_receive_data.cpp)
target_link_libraries(lightreceivedata ${catkin_LIBRARIES}
${WIRINGPI_LIBRARY})
```

- **Example launch code**

```
<launch>

<!--connect arduino-->
<!--node name="connect_arduino" pkg="rosserial_python"
type="serial_node.py">
<param name="~baud" type="int" value="57600" />
<param name="~port" type="string" value="/dev/ttyACM0" />
</node-->

<!--node name="name" pkg="your package" type="Executive file"/-
->

<node name="light_receive_data" pkg="light_receive_data" type =
"lightreceivedata"></node>
</launch>
```