



PROGRAMMING A RASPBERRY PI WITH PYTHON

التجربة 5

إعداد: م. علا جزماتي

Import modules

module

- Functions

time

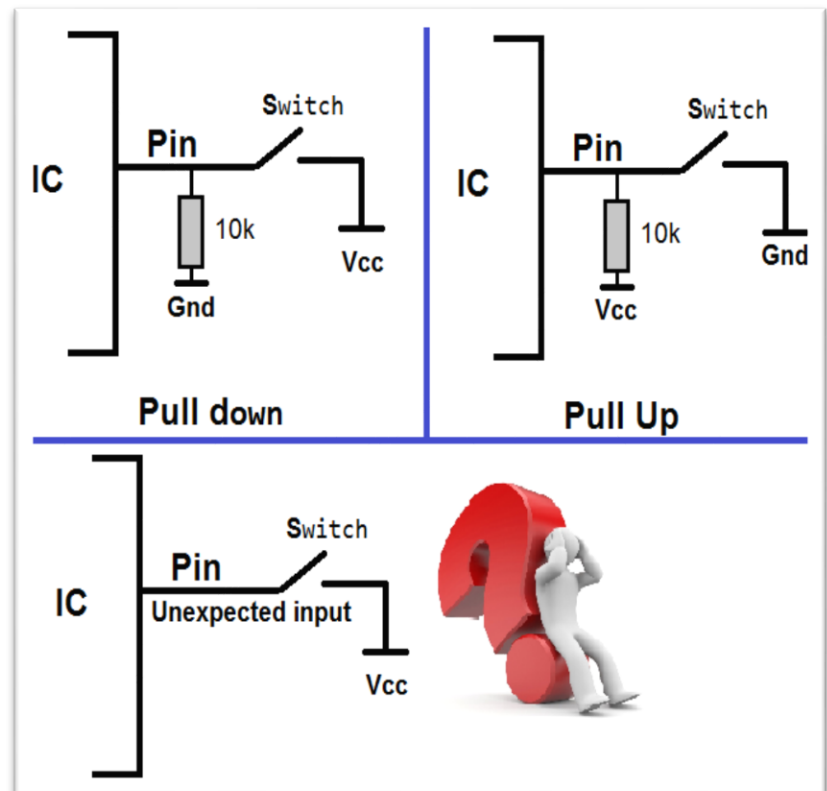
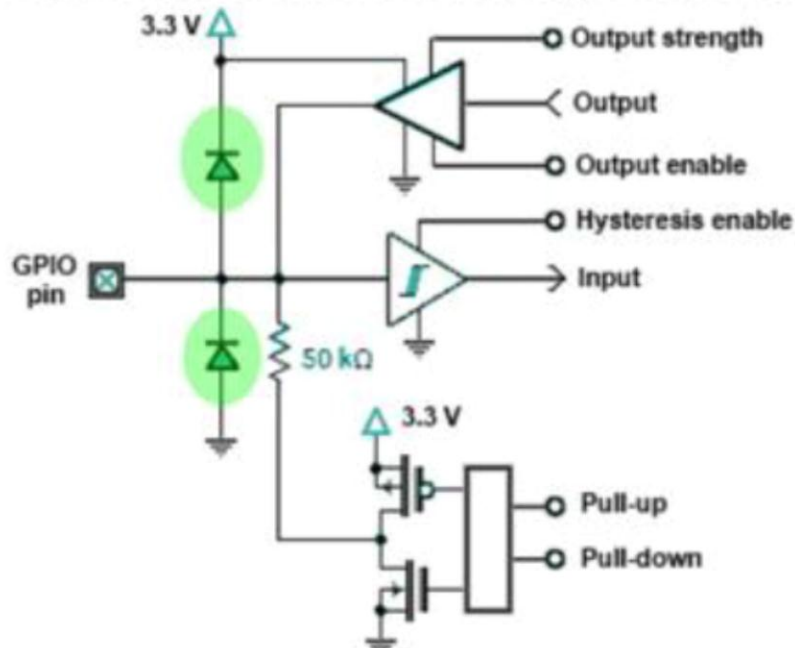
- sleep()
- time()

RPi.GPIO

- setmode()
 - setup()
- setwarnings()
 - output()
 - input()

التصريح عن أقطاب الدخل في Raspberry pi باستخدام مكتبة Rpi.GPIO

Equivalent Circuit for Raspberry Pi GPIO pins

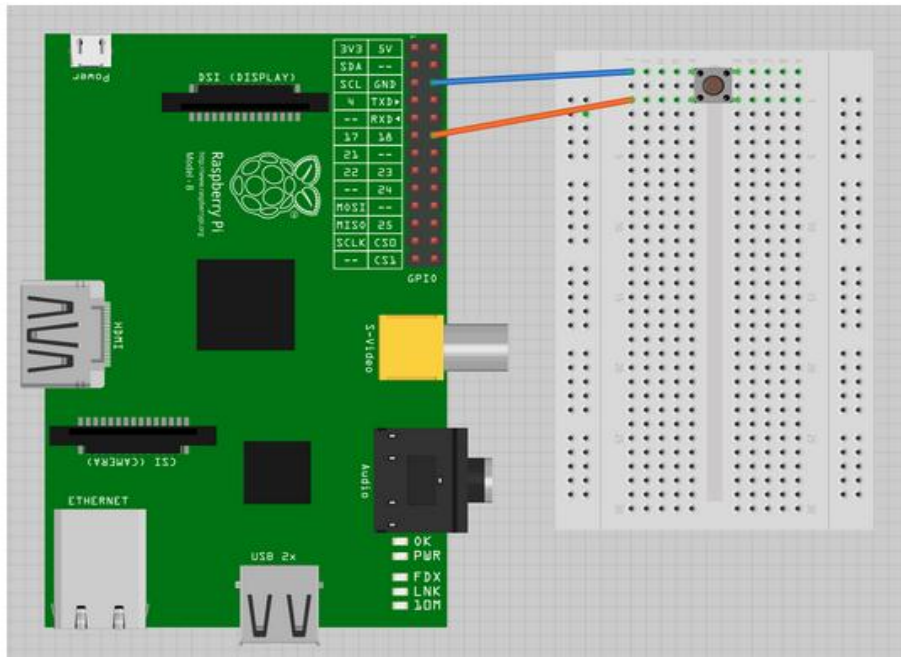


التصريح عن أقطاب الدخل في Raspberry pi باستخدام مكتبة Rpi.GPIO

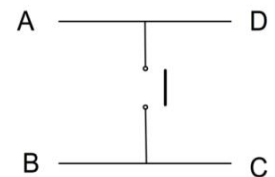
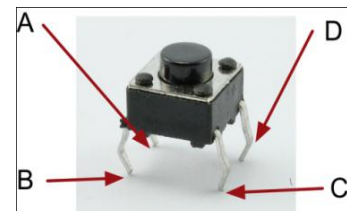
```
GPIO.setup(13, GPIO.IN)
```

```
GPIO.setup(23, GPIO.IN, pull_up_down = GPIO.PUD_DOWN)
```

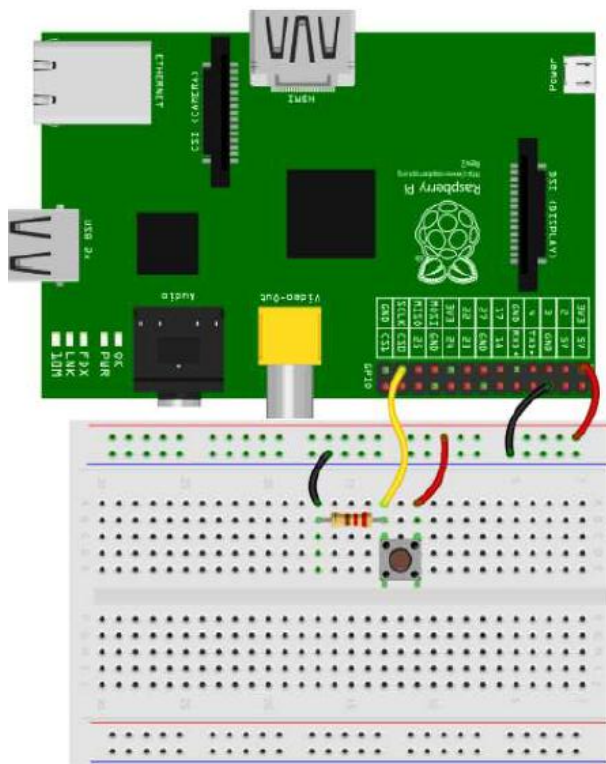
```
GPIO.setup(24, GPIO.IN, pull_up_down = GPIO.PUD_UP)
```



Let's try some examples:



Reading a Switch state



```
import time
import RPi.GPIO as GPIO
GPIO.setmode(GPIO.BOARD)
GPIO.setup(23, GPIO.IN)
while True:
    if (GPIO.input(23)==True):
        print "Input is True (3.3 volt)"
    else:
        print "Input is False (zero volt)"
    time.sleep(1)
```

قم بتعديل البرنامج التالي بإضافة متصل ضوئي يضئ عند الضغط على المفتاح اللحظي مع تعريف متحولات تغيير الحالة وحذف التأخير الزمني

شرح برنامج قراءة حالة الدخل:

استدعاء الموديلات اللازمة

```
import time  
import RPi.GPIO as GPIO
```

اختيار ترتيب الأقطاب حسب ترقيم اللوحة

```
GPIO.setmode(GPIO.BOARD)
```

تحديد طبيعة القطب 23 كقطب دخل

```
GPIO.setup(23,GPIO.IN)
```

```
While True:
```

حلقة

شرط: عندما يكون القطب 23 (on) يقوم بطباعة "Input is True"

```
    if(GPIOinput(23)==True):  
        print("Input is True")
```

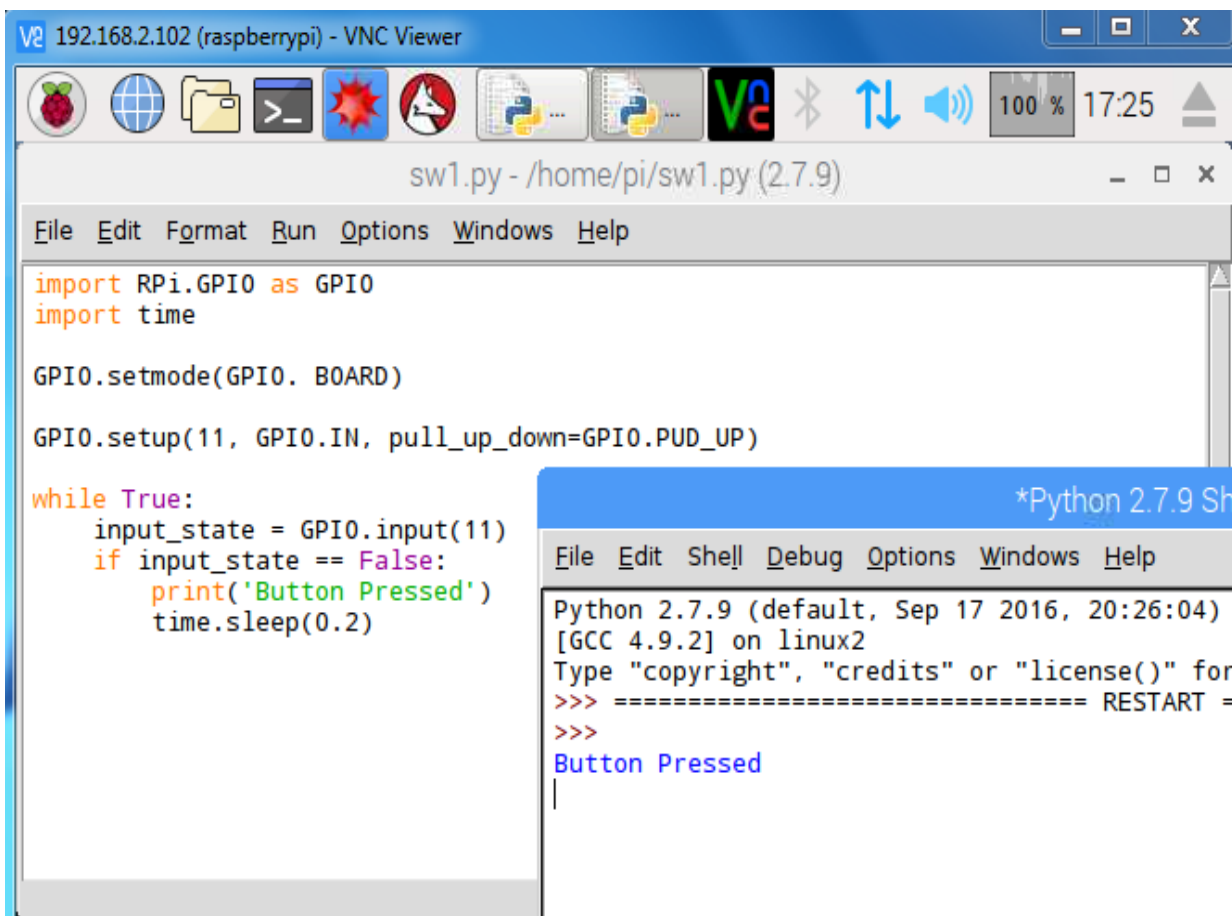
وإلا يطبع "Input is False"

```
    else:  
        print("Input is True")
```

تأخير زمني بمدة ثانية واحدة

```
time.sleep(1)
```

Reading a Switch state



```
192.168.2.102 (raspberrypi) - VNC Viewer
sw1.py - /home/pi/sw1.py (2.7.9)
File Edit Format Run Options Windows Help

import RPi.GPIO as GPIO
import time

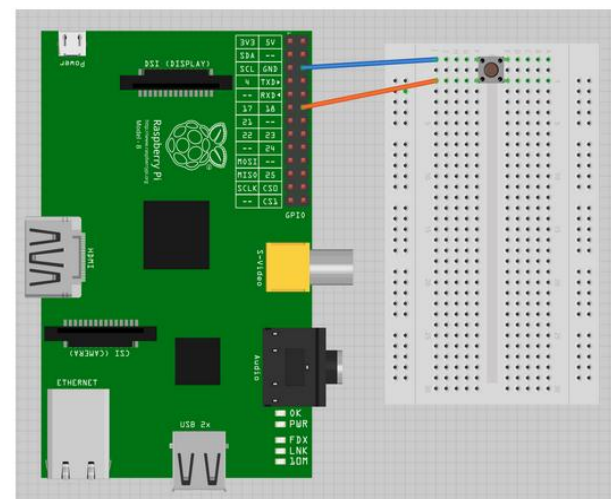
GPIO.setmode(GPIO.BOARD)

GPIO.setup(11, GPIO.IN, pull_up_down=GPIO.PUD_UP)

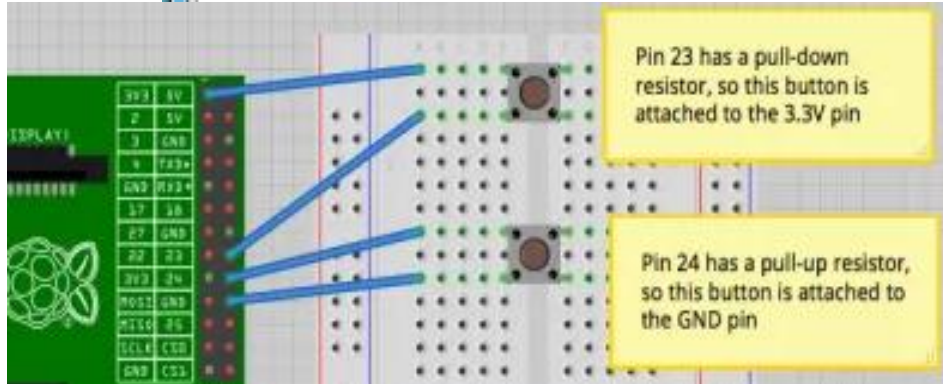
while True:
    input_state = GPIO.input(11)
    if input_state == False:
        print('Button Pressed')
        time.sleep(0.2)
```

```
*Python 2.7.9 Shell
File Edit Shell Debug Options Windows Help

Python 2.7.9 (default, Sep 17 2016, 20:26:04)
[GCC 4.9.2] on linux2
Type "copyright", "credits" or "license()" for
>>> ===== RESTART =====
>>>
Button Pressed
```



Reading a Switch state



تطبيقات التجربة

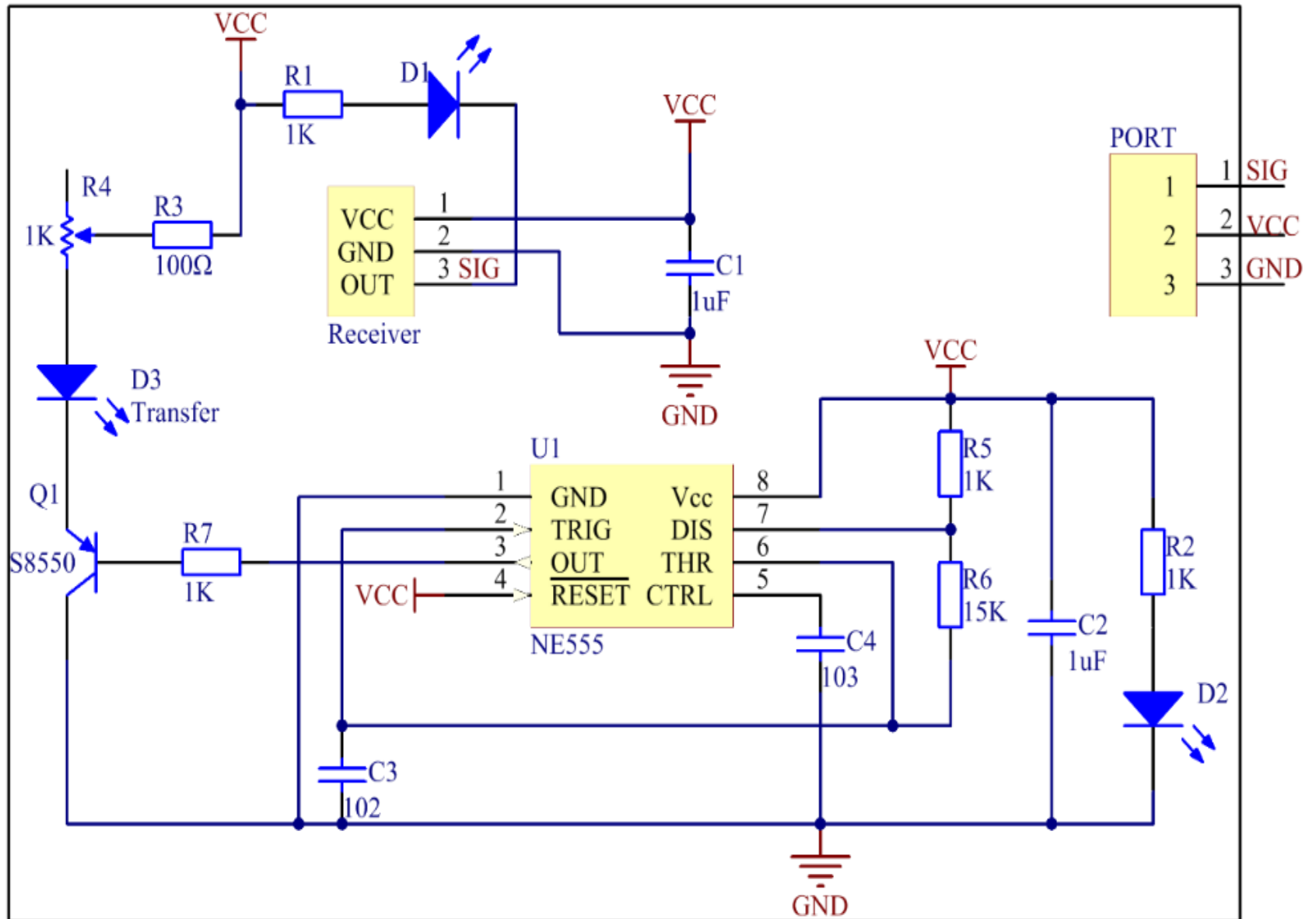
التطبيق الأول:

تصميم خوارزمية لتجنب العقبات (An obstacle avoidance) بالاعتماد على حساسات الأشعة تحت الحمراء (حساس مكتشف الحواجز), فعندما لا يكون هناك عقبة أو جسم أمام المرسل لن يتم انعكاس الأشعة إلى المستقبل, وفي حال وجود جسم سيتم الانعكاس وقيمة الإشارة تتناسب مع بعد الجسم , ويمكن معايرتها باستخدام المقاومات المتغيرة الموجودة ضمن النموذج, يغطي مسافة من 2 إلى 40 سم.



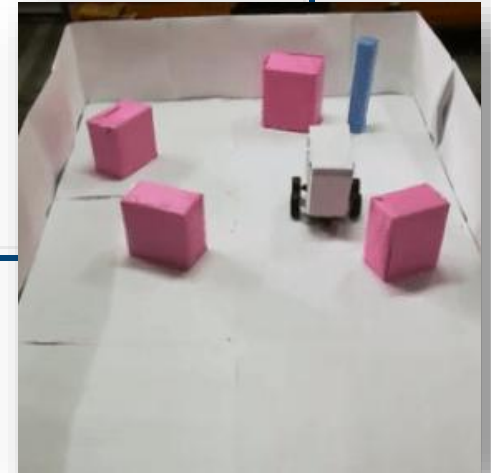
Obstacle Avoidance Sensor

Obstacle Avoidance Sensor



Python Code

```
import RPi.GPIO as GPIO
ObstaclePin = 11
def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(ObstaclePin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
def loop():
    while True:
        if (GPIO.input(ObstaclePin)== 0):
            print "Barrier is detected !"
def destroy():
    GPIO.cleanup()                      # Release resource
if __name__ == '__main__':             # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:          # 'Ctrl+C'
        destroy()
```



شرح البرنامج:

استدعاء موديول التحكم بأقطاب GPIO

```
import RPi.GPIO as GPIO
```

اسناد القطب 11 إلى المتغير Obstaclepin

```
Obstaclepin=11
```

تعريف تابع setup لتفعيل الأقطاب وترتيب المنافذ بناء على ترقيم اللوحة وتحديد طبيعة القطب Obstaclepin كقطب دخل مع تفعيل مقاومة الرفع الداخلية

```
def setup():
```

```
    GPIO.setmode(GPIO.BOARD)
```

```
    GPIO.setup(Obstaclepin,GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

تعريف تابع loop الحلقة حيث يحتوي شرط عندما يكون الدخل مساوي إلى 0 أي تم اكتشاف حاجز فنقوم بطباعة عبارة "Barrier is detected"

```
def loop():
```

```
    While True:
```

```
        If(GPIO.input(Obstaclepin)==0):
```

```
            Print("Barrier is detected")
```

تعريف تابع destroy لمسح مسجلات الأقطاب لاستقبال قيم جديدة عند تنفيذ البرنامج مرة أخرى

```
def destroy():
```

```
    GPIO.cleanup()
```

عند إقلاع البرنامج تكون الحالة الافتراضية للمتحول __name__ مساوية '__main__' وبالتالي يتم استدعاء تابع setup

```
if __name__ == '__main__':
```

```
    setup()
```

ثم قمنا باستخدام تعليمة try لتنفيذ استدعاء تابع loop من خلالها الذي يقوم بفحص قطب الدخل ويقوم بتحديد إذا كان هناك حاجز أم لا

```
    try:
```

```
        loop()
```

ويستمر تنفيذ التتابع والتعليمات داخل try حتى تتم مقاطعة لوحة المفاتيح عند الضغط على ctrl+C وعند حصول المقاطعة يتم استدعاء التابع destroy الذي يقوم بتهيئة الأقطاب

```
except KeyboardInterrupt:
```

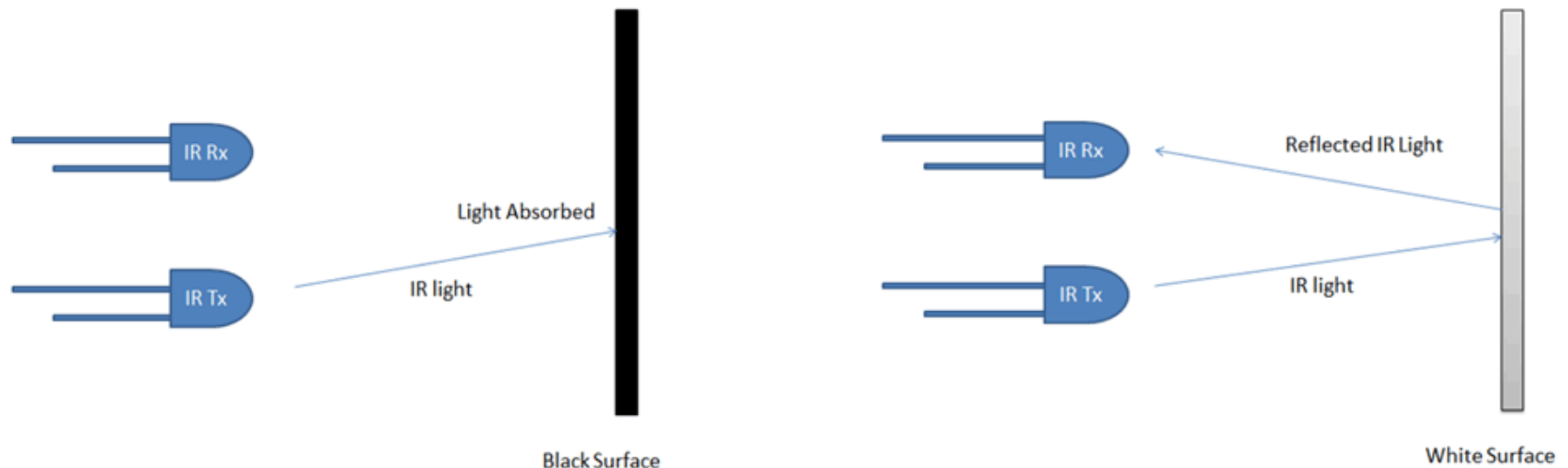
```
    destroy()
```


تصميم خوارزمية للملاحقة باستخدام (Tracking Sensor) بالاعتماد على حساسات الأشعة تحت الحمراء (مرسل - مستقبل) أيضا، ولكن مع اختلاف عن نموذج الحساس السابق أنه باستطاعة إرسال (transmitting power) منخفضة.

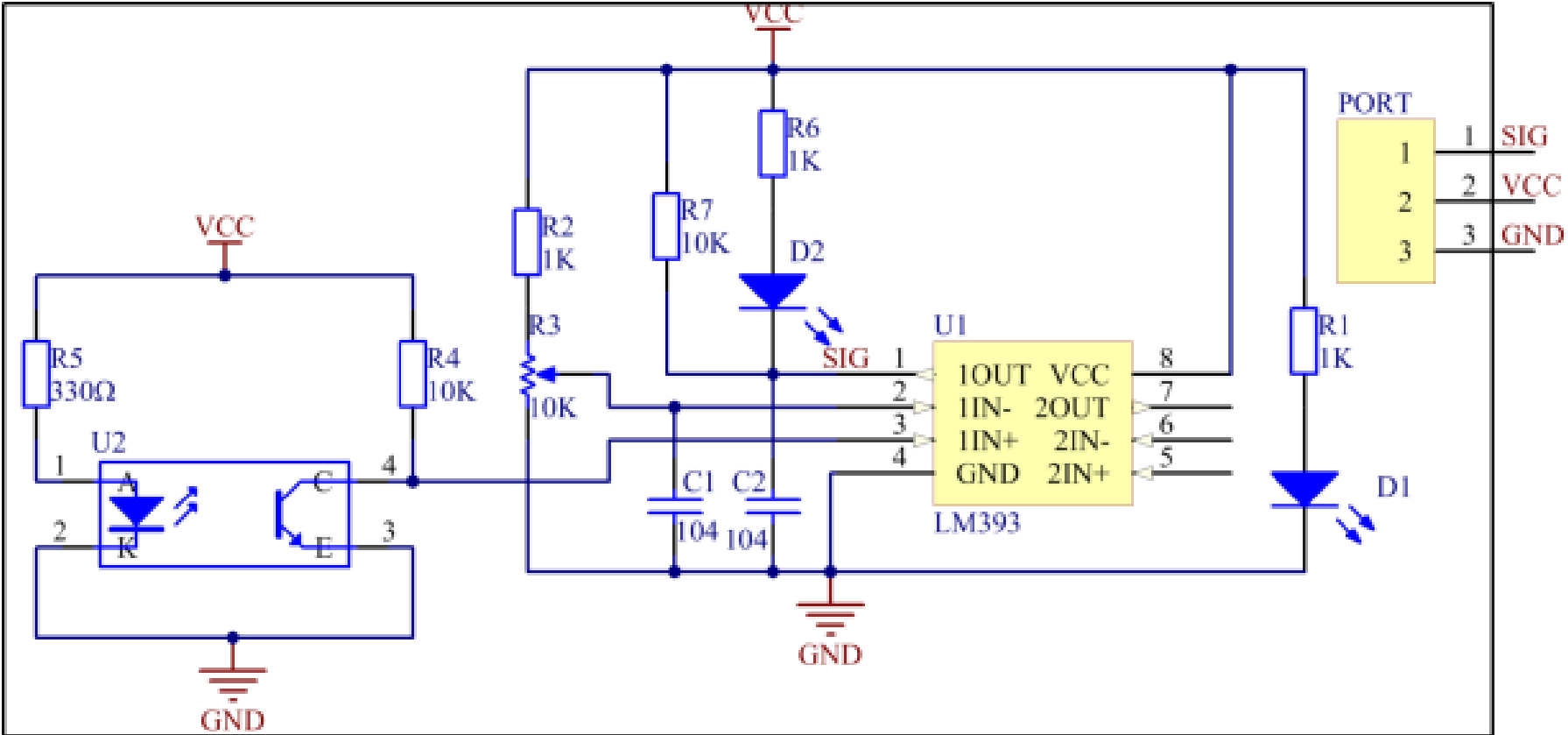


Tracking Sensor

When the infrared transmitter on the sensor emits rays to a piece of paper, if the rays shine on a white surface, they will be reflected and received by the receiver, and pin S will output low level; If the rays encounter black lines, they will be absorbed, thus the receiver gets nothing, and pin S will output high level.



Tracking Sensor



Python Code

```
import RPi.GPIO as GPIO
TrackPin = 11
LedPin    = 12
def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(LedPin, GPIO.OUT)
    GPIO.setup(TrackPin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
    GPIO.output(LedPin, GPIO.HIGH)
def loop():
    while True:
        if GPIO.input(TrackPin) == GPIO.LOW:
            print 'White line is detected'
            GPIO.output(LedPin, GPIO.LOW)    # led on
        else:
            print '...Black line is detected'
            GPIO.output(LedPin, GPIO.HIGH)    # led off
def destroy():
    GPIO.output(LedPin, GPIO.HIGH)           # led off
    GPIO.cleanup()
if __name__ == '__main__':                  # Program start from here
    setup()
    try:
        loop()
    except KeyboardInterrupt:               # When 'Ctrl+C' is pressed
        destroy()
```

```
import RPi.GPIO as GPIO
```

اسناد القطب 11 إلى المتغير Trackpin والقطب 12 للمتغير Ledpin

```
Trackpin = 11
```

```
Ledpin = 12
```

تعريف تابع setup لتفعيل الأقطاب واختيار طريقة ترقيم الأقطاب وتحديد طبيعة القطب Trackpin كقطب دخل مع تفعيل مقاومة الرفع الداخلية وتفعيل القطب Ledpin كقطب خرج وتفعيل القطب بقيمة 1 أي (Led(off)).

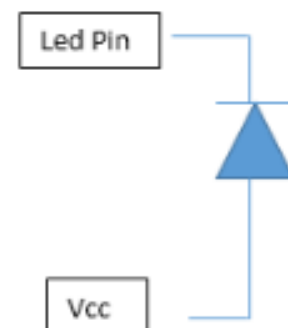
```
def setup():
```

```
    GPIO.setmode(GPIO.BOARD)
```

```
    GPIO.setup(Ledpin, GPIO.OUT)
```

```
    GPIO.setup(Trackpin, GPIO.IN, pull_up_down=GPIO.PUD_UP)
```

```
    GPIO.output(Ledpin, GPIO.HIGH)
```



تعريف تابع loop حيث يحتوي شرط :

عندما يكون الدخل Trackpin مساوياً 0 أي تم ارتداد الأشعة تحت الحمراء وتم كشف الخط الأبيض ويتم طباعة عبارة "white line is detected" وإسناد قيمة 0 للخرج 12 فيكون LED(on).

وإلا إذا كان Trackpin =1 أي تم امتصاص الأشعة تحت حمراء وتم كشف الخط الأسود وتتم طباعة العبارة "black line is detected" وإسناد 1 للخرج 12 فيكون LED(off).

def loop():

While True:

 If(GPIO.input(Trackpin)==GPIO.LOW):

 Print("white line is detected")

 GPIO.output(Ledpin,GPIO.LOW)

 else:

 Print("black line is detected")

 GPIO.output(Ledpin,GPIO.HIGH)

تعريف تابع destroy يجعل Led(off) ويعيد تهيئة الأقطاب

```
def destroy():
```

```
    GPIO.output(Ledpin,GPIO.HIGH)
```

```
    GPIO.cleanup()
```

عند إقلاع البرنامج تكون الحالة الافتراضية للمتحول __name__ مساوية '__main__' وبالتالي يتم استدعاء تابع setup

```
if __name__ == '__main__':
```

```
    setup()
```

ثم قمنا باستخدام تعليمة try لتنفيذ استدعاء تابع loop من خلالها الذي يقوم بفحص قطب الدخل ويقوم بتحديد إذا كان هناك حاجز أم لا

```
    try:
```

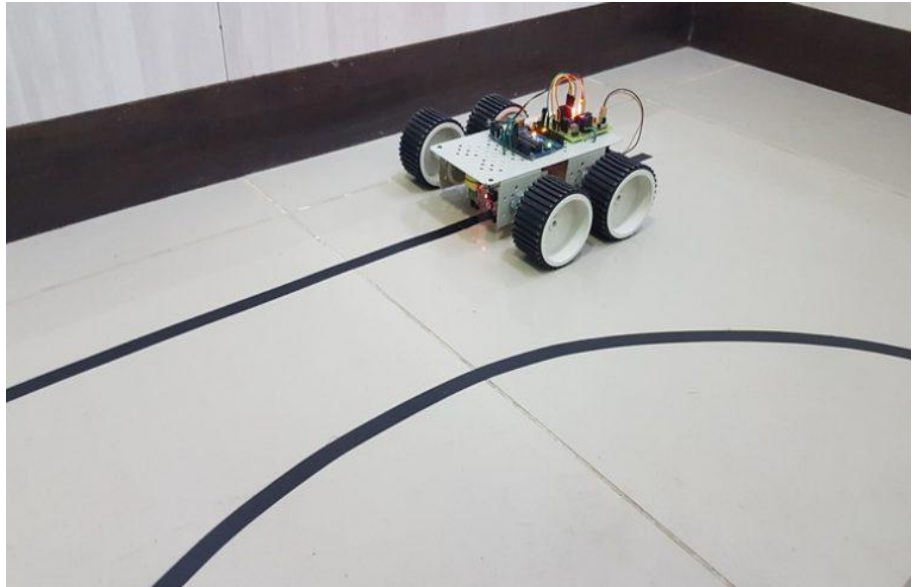
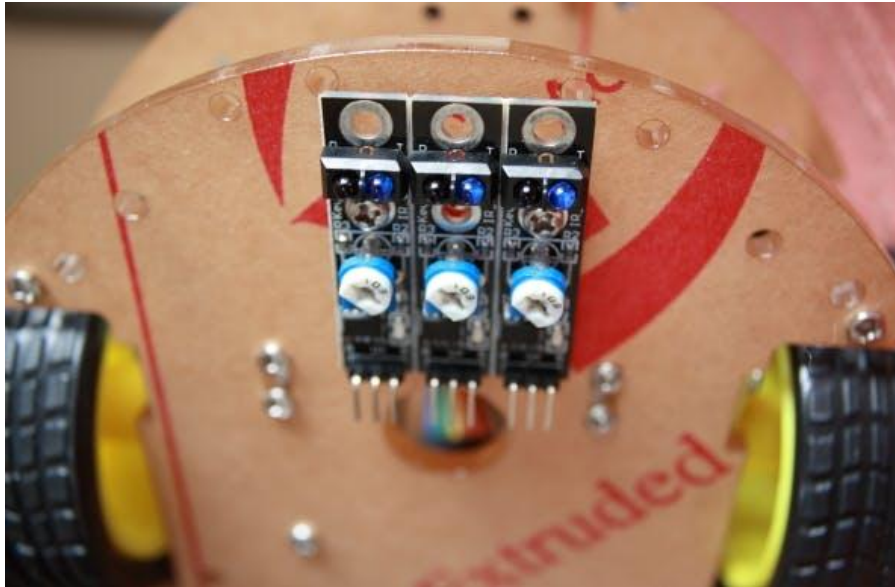
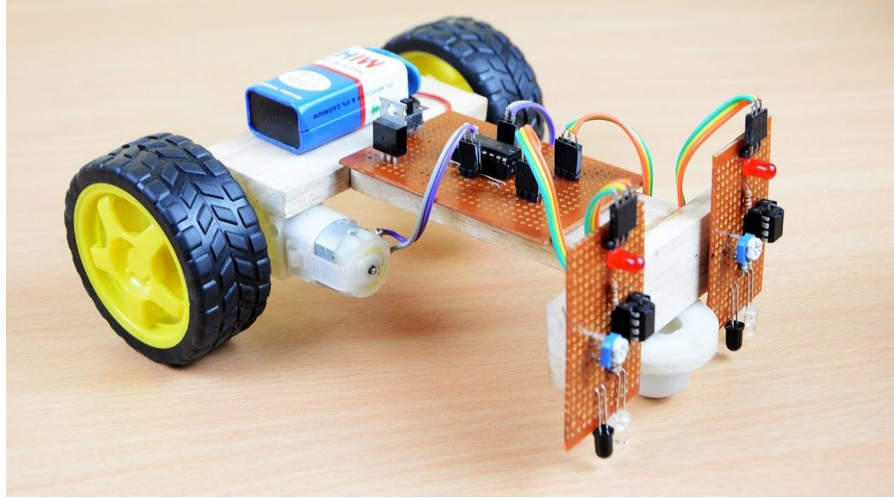
```
        loop()
```

ويستمر تنفيذ التتابع والتعليمات داخل try حتى تتم مقاطعة لوحة المفاتيح عند الضغط على ctrl+C وعند حصول المقاطعة يتم استدعاء التابع destroy الذي يقوم بتهيئة الأقطاب.

```
except KeyboardInterrupt:
```

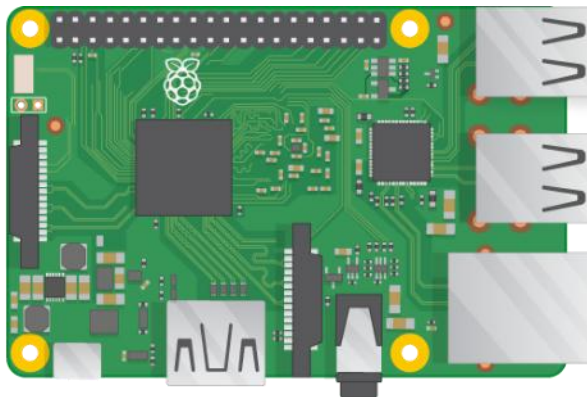
```
    destroy()
```

تطبيقات مختلفة مثل روبوتات تتبع الخط الأسود

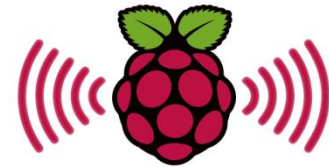
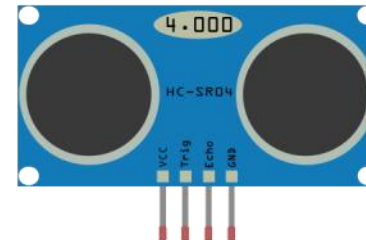




Ultrasonic Distance Measurement Using Python



+



هناك أنواع مختلفة من الموديولات التي تستخدم لقياس المسافة بعضها يعتمد على تقنية الأمواج فوق الصوتية أو الأشعة تحت الحمراء أو الليزر وغيرها ..
حاول أن تقارن بينها .. هل تعرف إصدارات أخرى منها .. ماهي مميزاتها ...



1



2



3

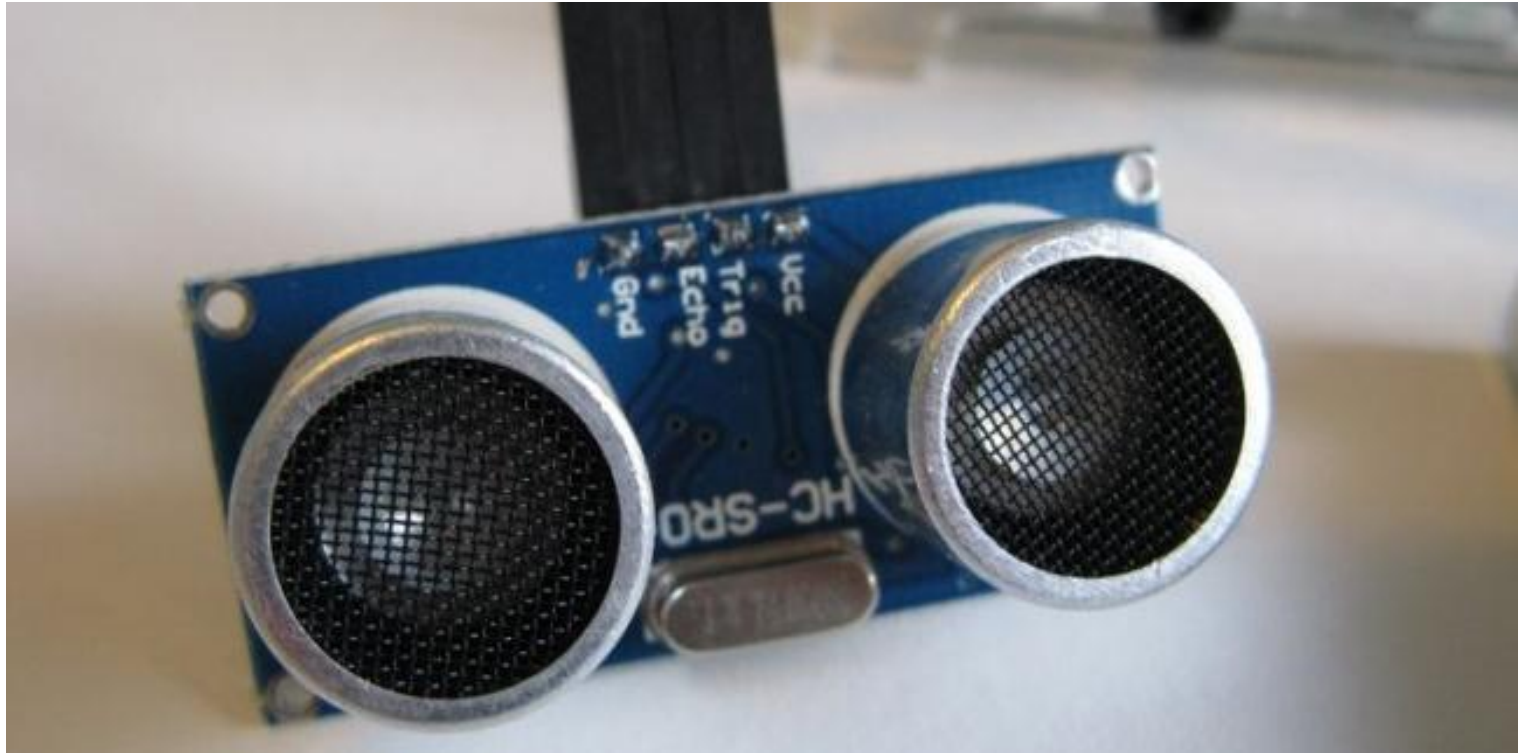


4



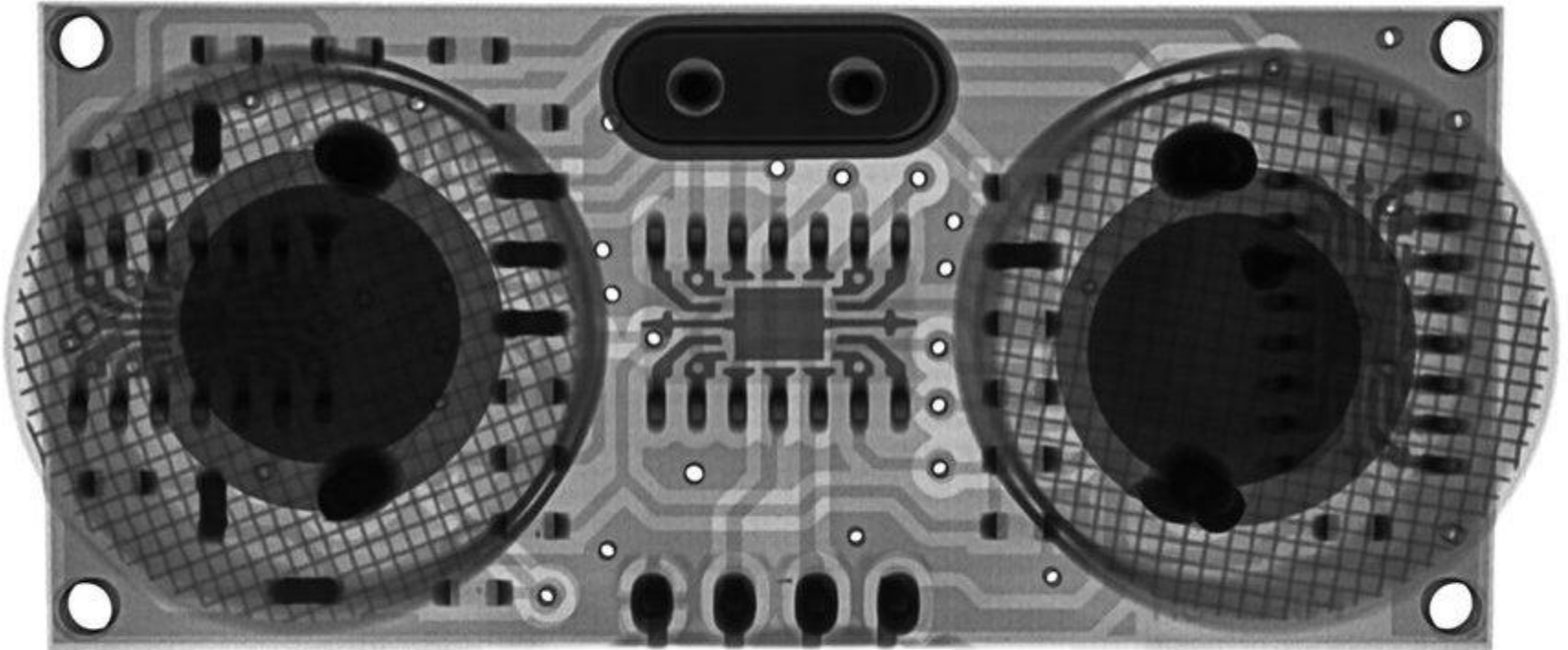
5

Ultrasonic Distance Measurement Using Python

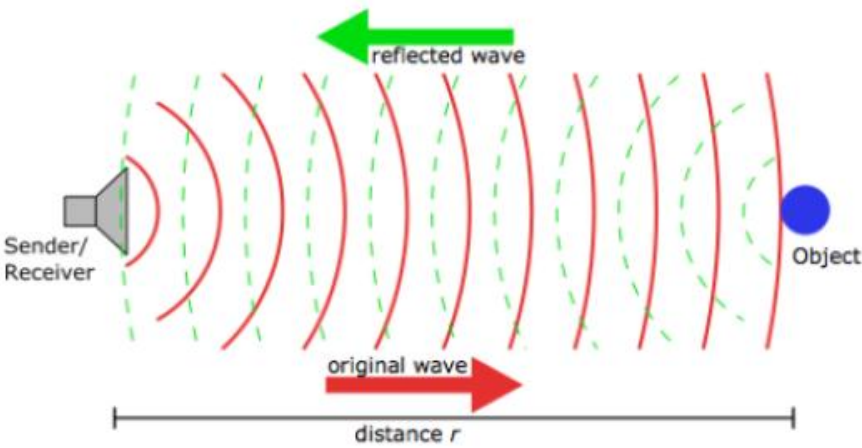


حساس المسافة UltraSonic HC-SR04

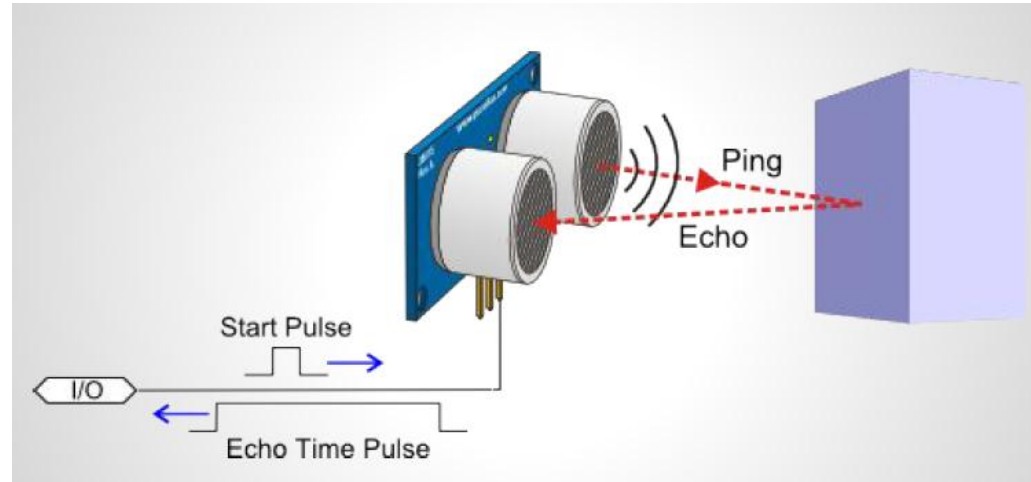
Ultrasonic Distance Measurement Using Python



Ultrasonic Distance Measurement Using Python



عملية إرسال واستقبال الأمواج فوق الصوتية ودورها في تحديد المسافة



فحساس المسافة هو جهاز نستطيع استخدامه لقياس بعد جسم ما عن الحساس بمجال يتراوح بين 3cm-

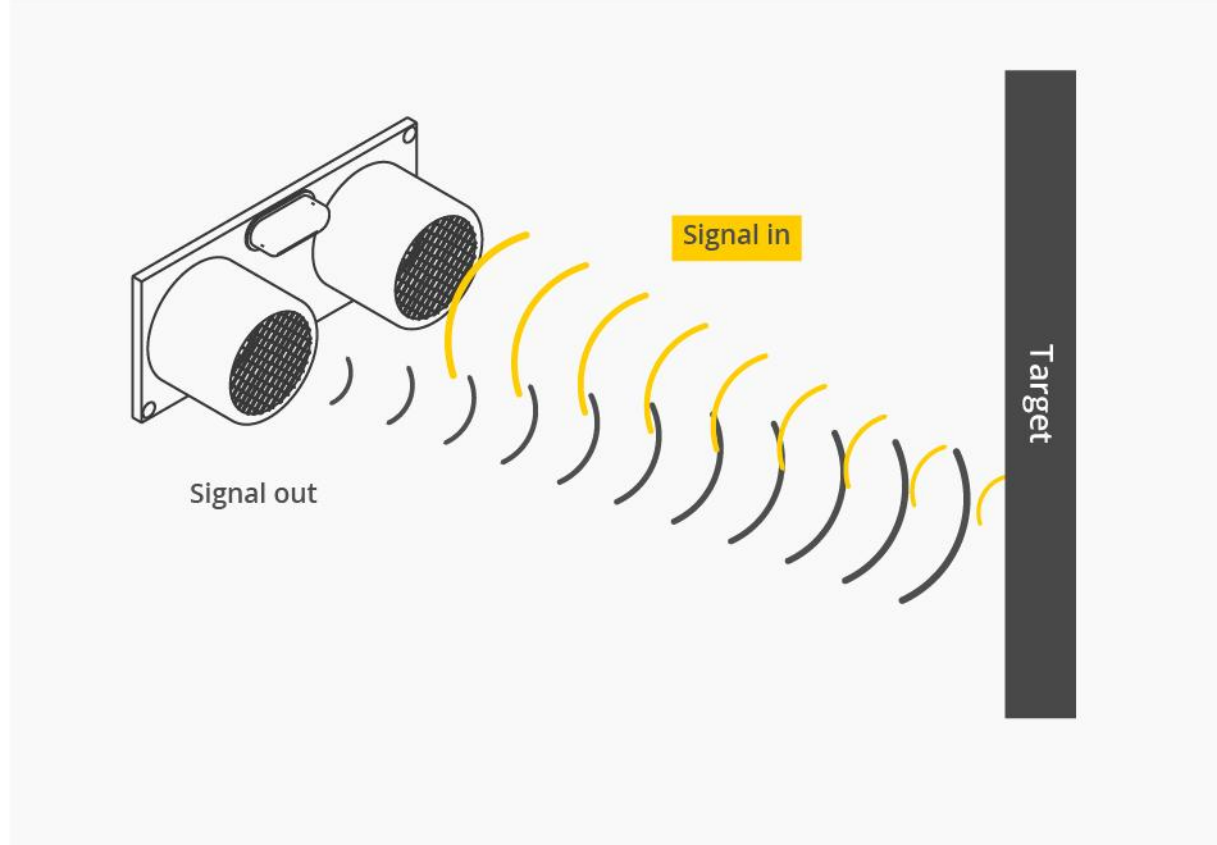
3.3m.

ويستخدم في تطبيقات الروبوت ومشاريع الأتمتة ، حيث يقيس مسافة الأجسام بدقة تصل إلى نصف سنتيمتر.

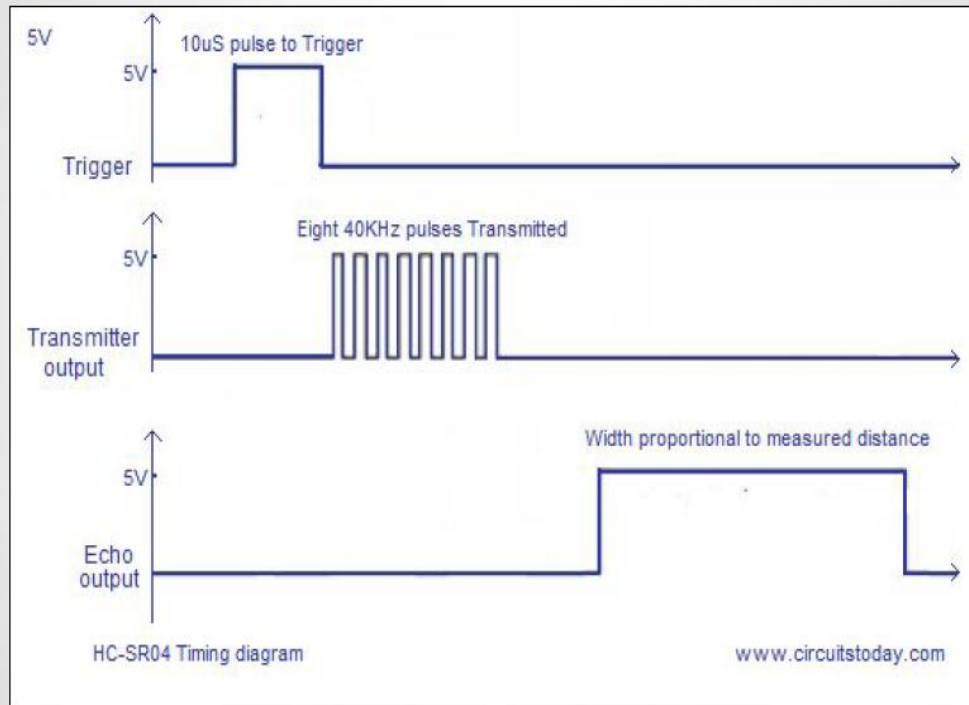
ما هو مبدأ عمل الحساس ؟



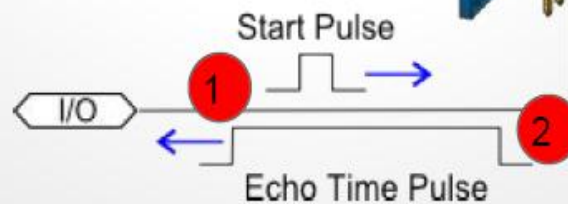
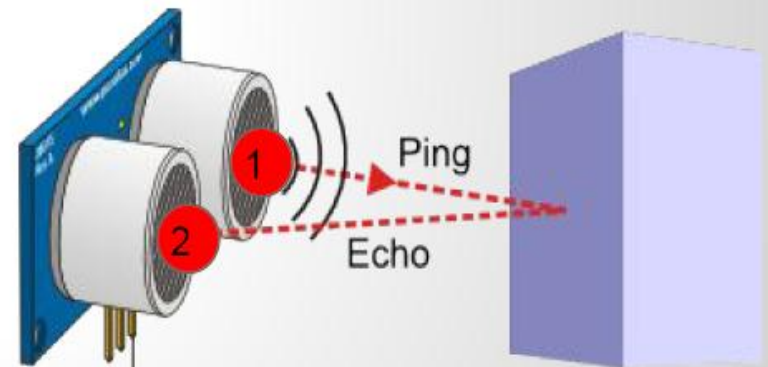
ما هو مبدأ عمل الحساس ؟



Ultrasonic Distance Measurement Using Python



Internal circuit workings



$$\frac{3.3}{5} = \frac{R2}{1000 + R2}$$

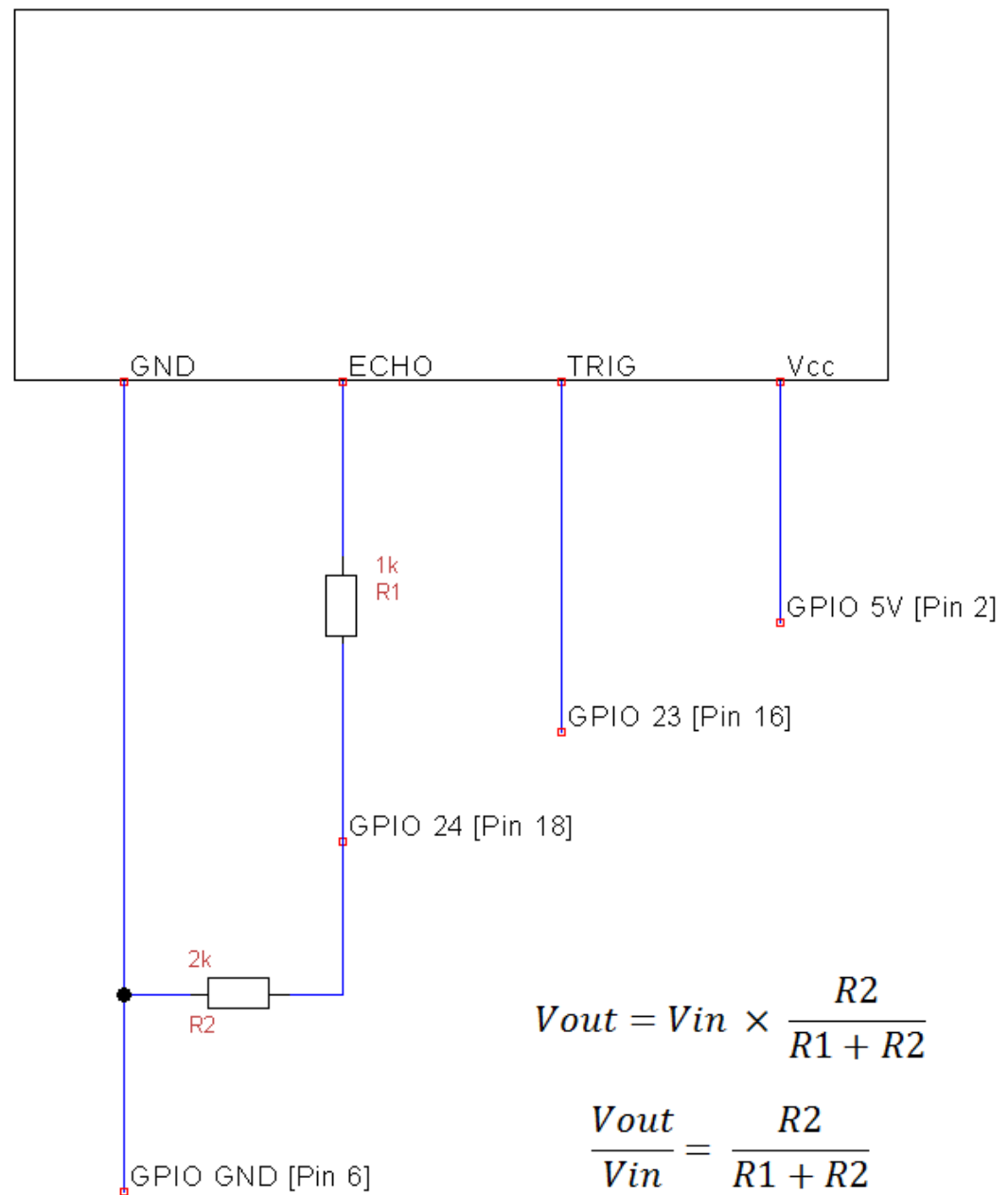
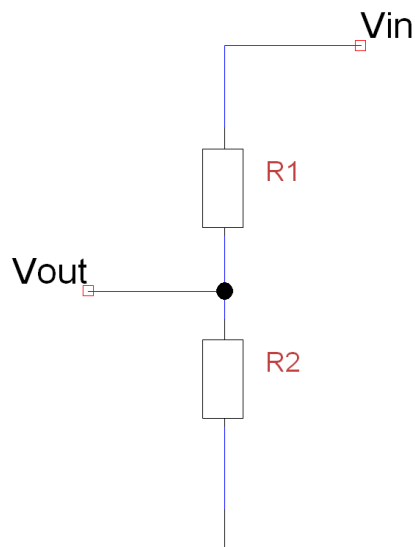
$$0.66 = \frac{R2}{1000 + R2}$$

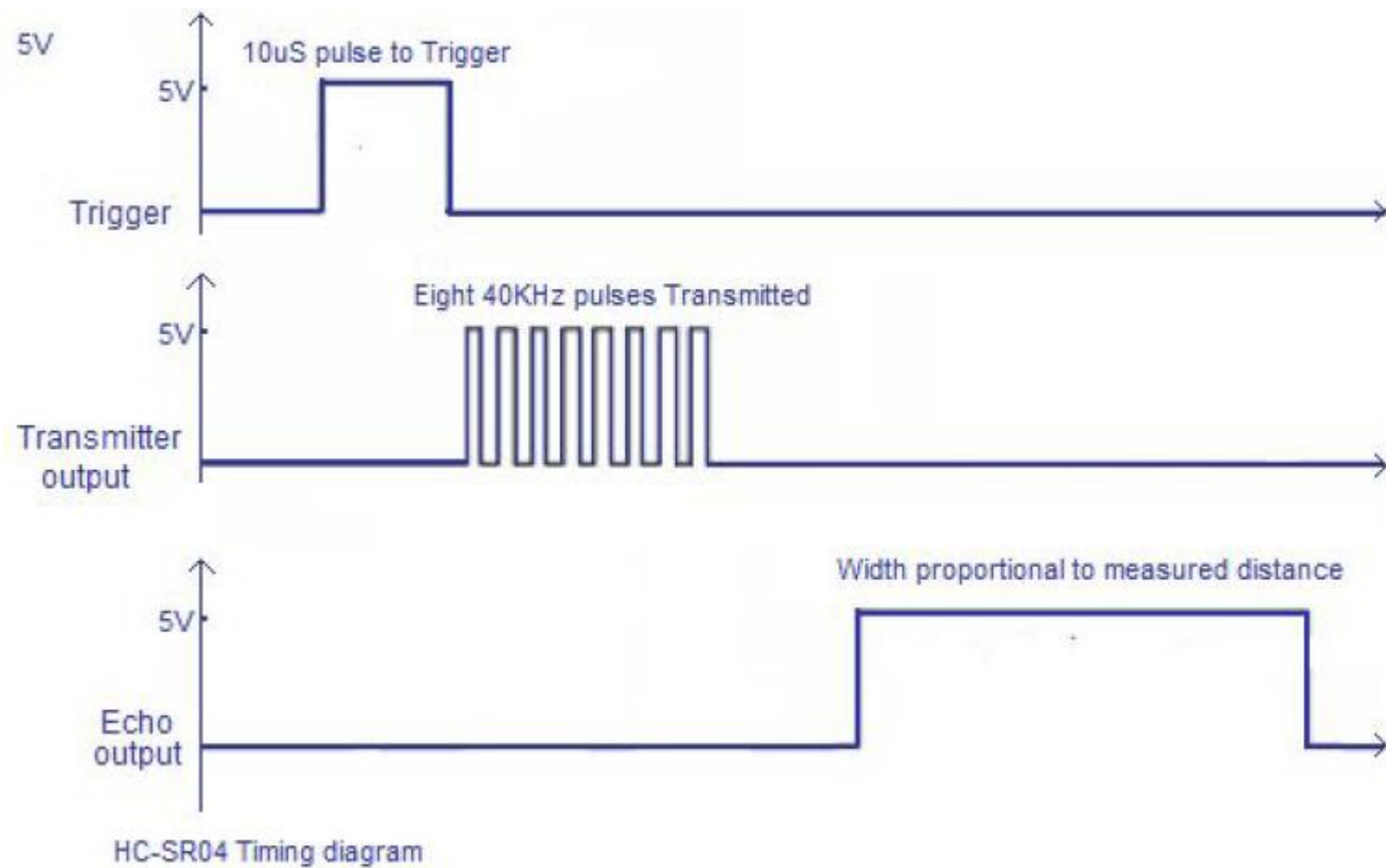
$$0.66(1000 + R2) = R2$$

$$660 + 0.66R2 = R2$$

$$660 = 0.34R2$$

$$1941 = R2$$





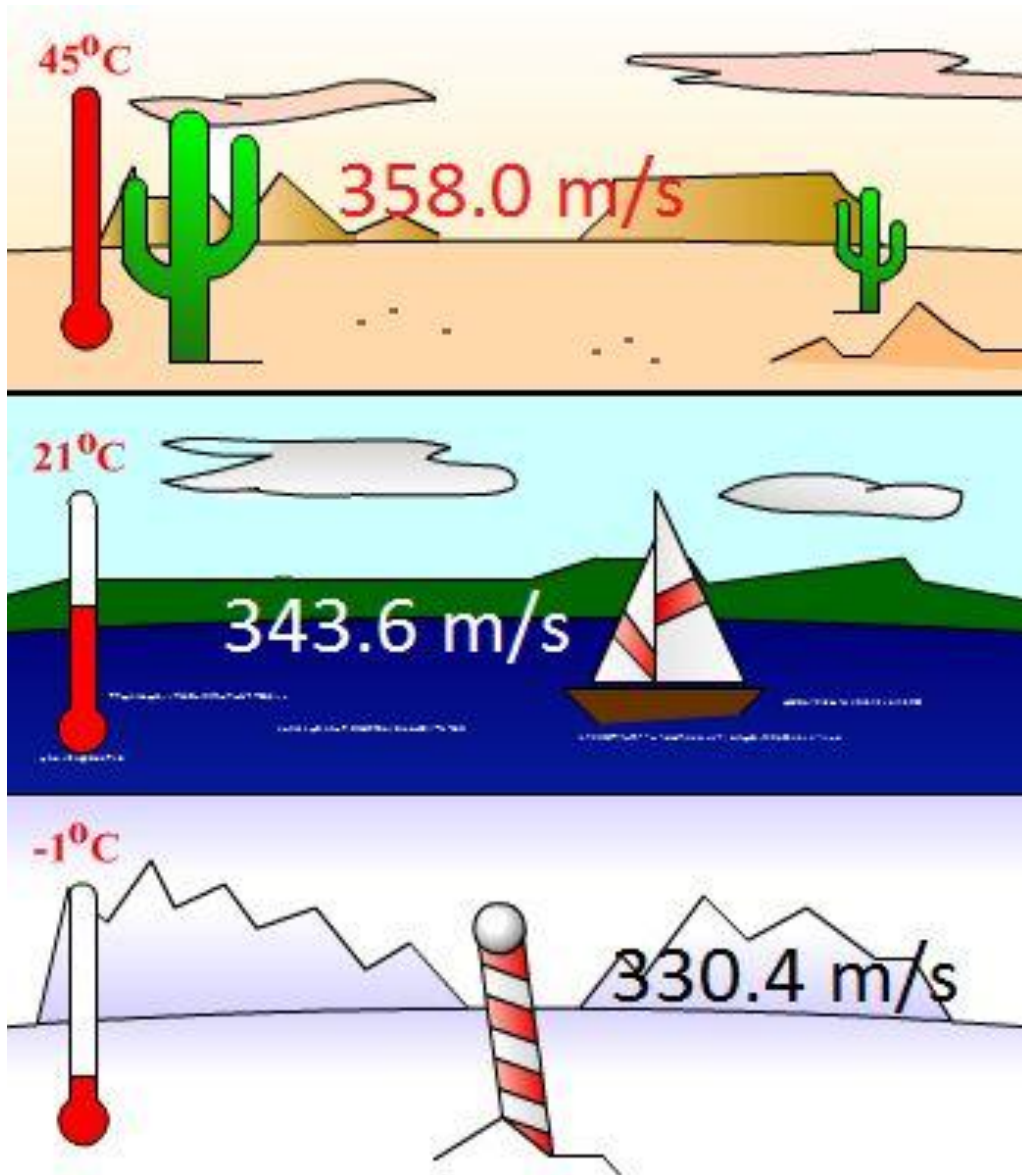
$$Speed = \frac{Distance}{Time}$$

$$34300 = \frac{Distance}{Time/2}$$

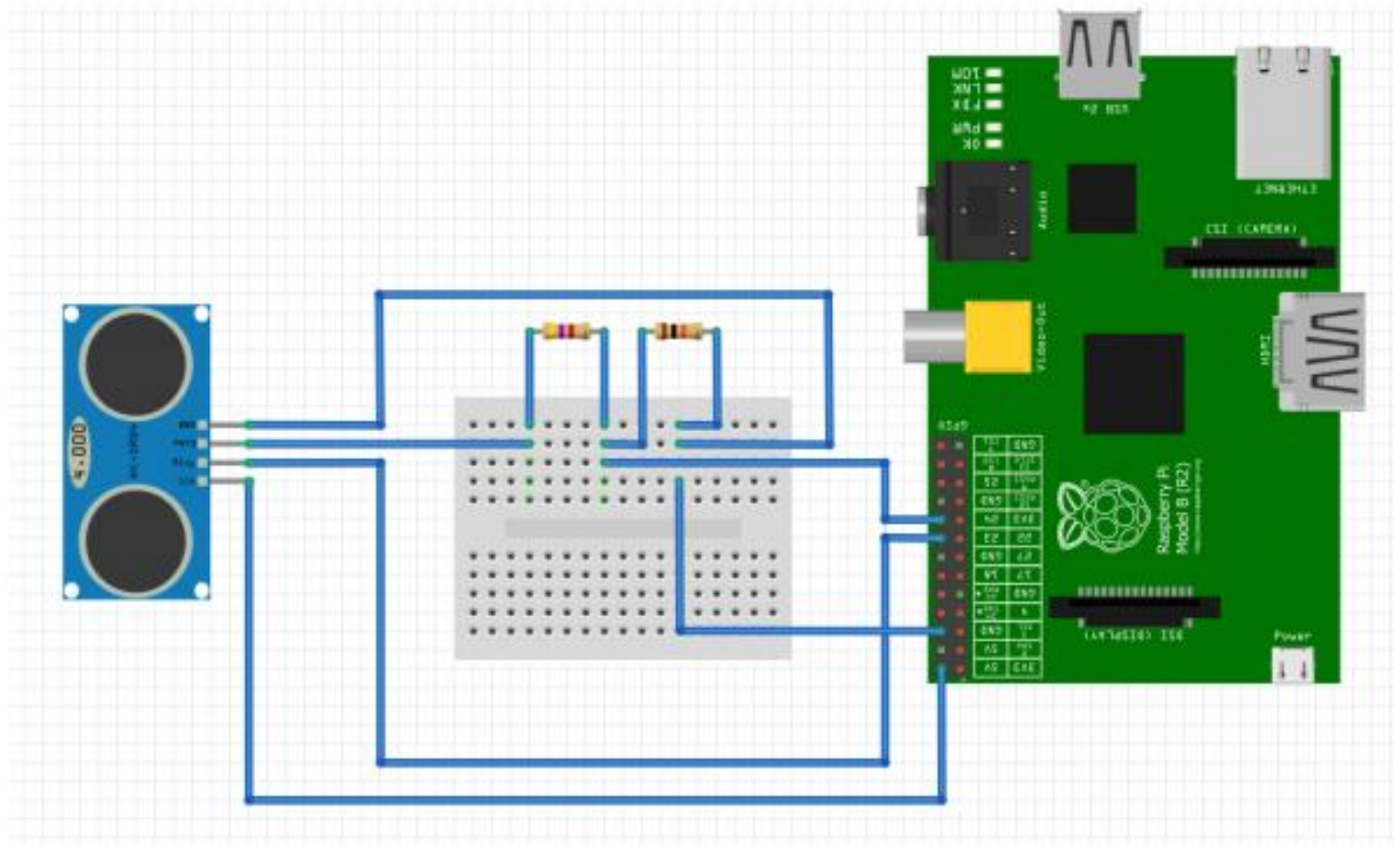
$$17150 = \frac{Distance}{Time}$$

$$17150 \times Time = Distance$$

ملاحظة مهمة :

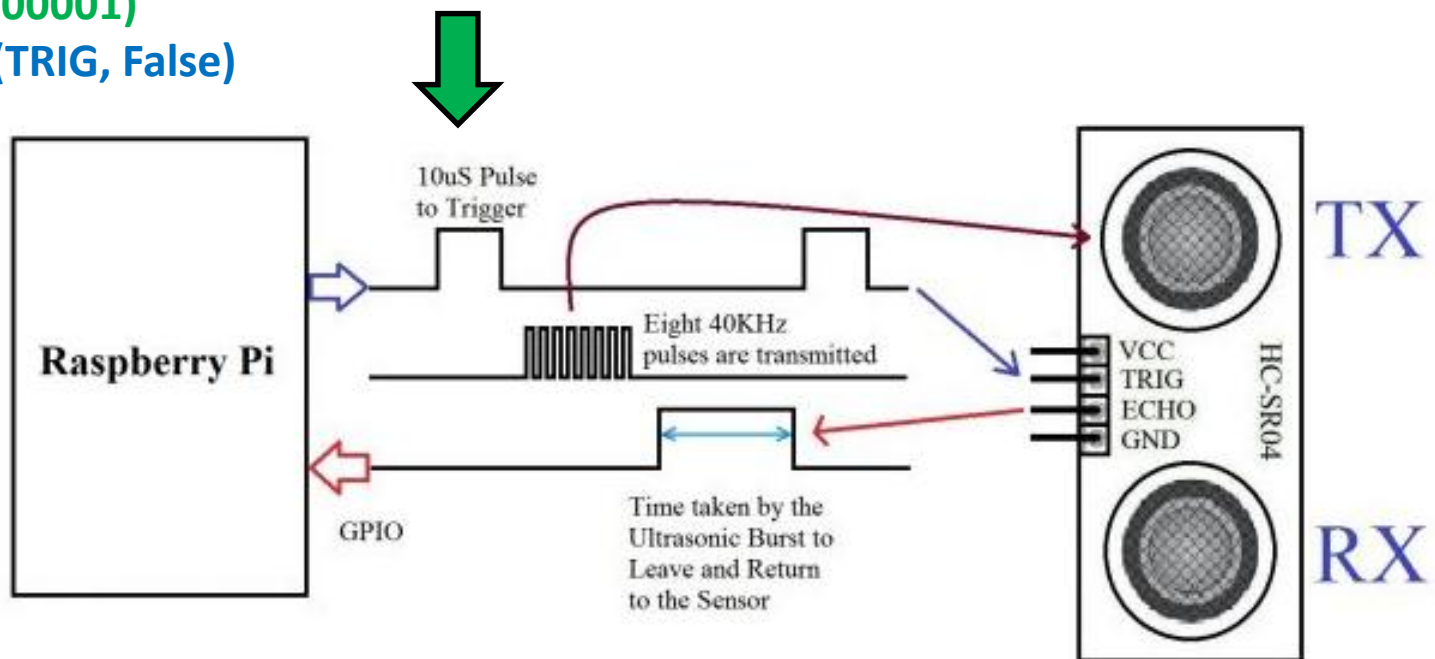


Ultrasonic Distance Measurement Using Python



Or $R1 = 470 \text{ ohm}$ $R2 = 1000\text{ohm}$

```
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BOARD)
TRIG = 16
ECHO = 18
Print("Distance Measurement In Progress")
GPIO.setup(TRIG,GPIO.OUT)
GPIO.setup(ECHO,GPIO.IN)
GPIO.output(TRIG, False)
Print("Waiting For Sensor To Settle")
time.sleep(2)
GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)
```



```
while GPIO.input(ECHO)==0:
```

```
    p_start = time.time()
```

```
while GPIO.input(ECHO)==1:
```

```
    p_end = time.time()
```

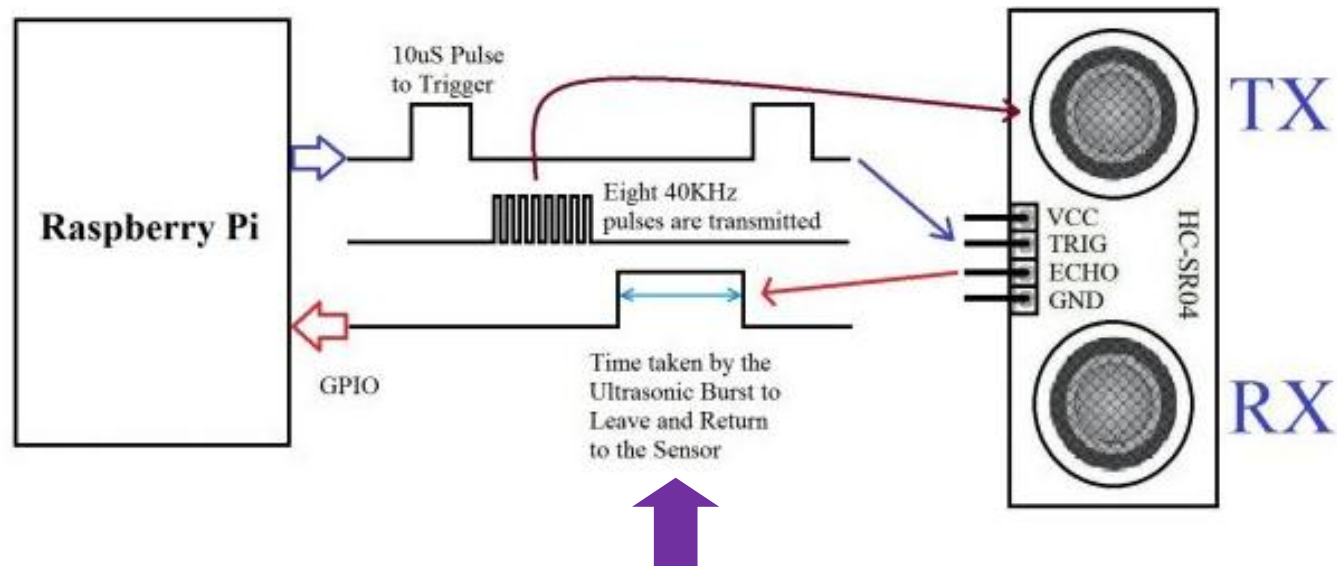
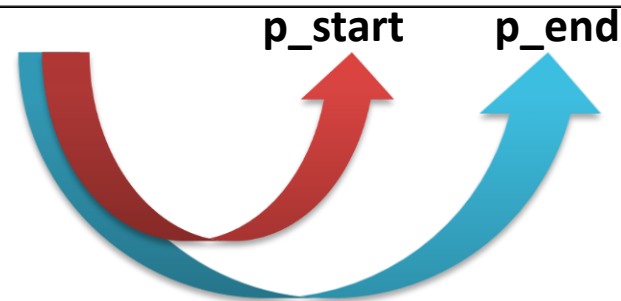
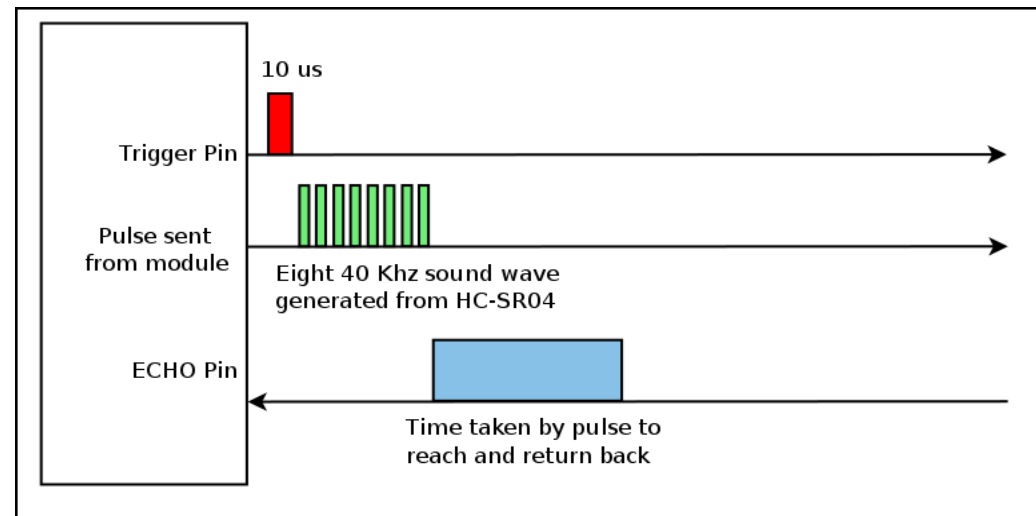
```
duration = p_end - p_start
```

```
distance = duration * 17150
```

```
distance = round(distance, 2)
```

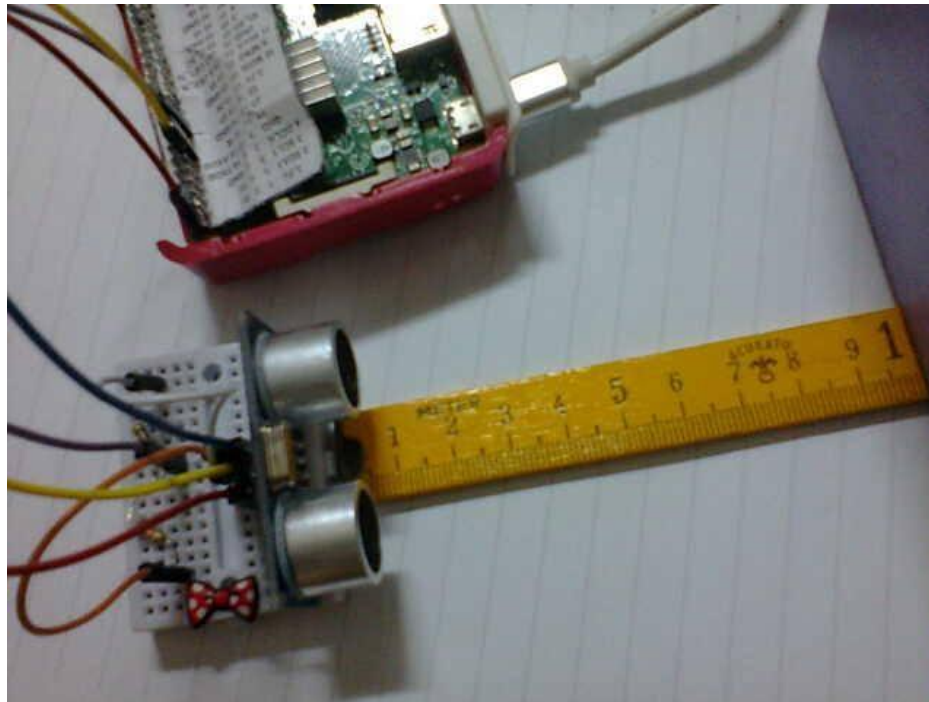
```
print ("Distance:",distance,"cm")
```

```
GPIO.cleanup()
```



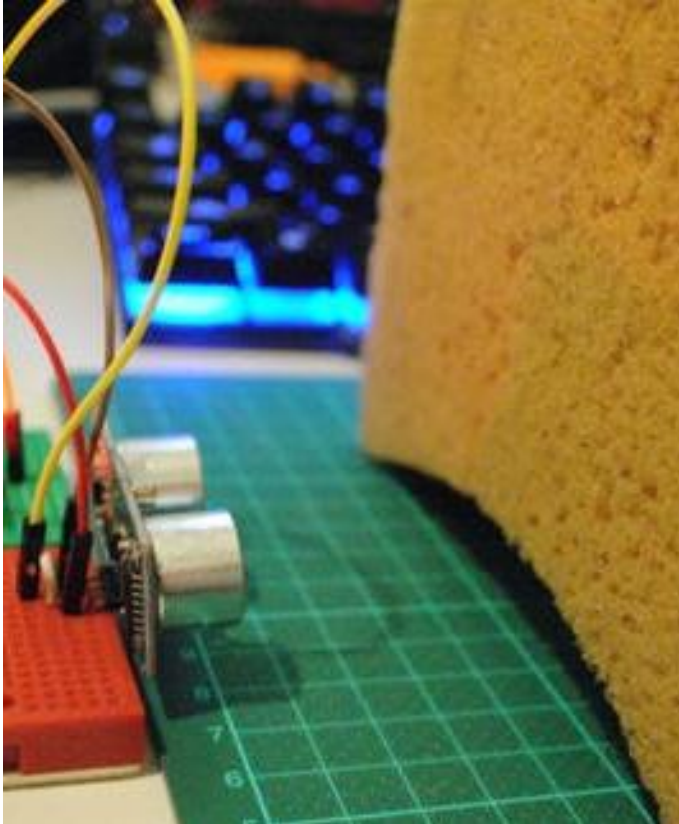
Ultrasonic Distance Measurement Using Python

```
pi@raspberrypi ~ $ sudo python range_sensor.py
Distance Measurement In Progress
Waiting For Sensor To Settle
Distance: 12.52 cm
pi@raspberrypi ~ $
```



التطبيق العملي:

قم بتجريب الكود وسجل النتائج التي تظهر معك بالمقارنة مع القياس بالمسطرة في الحالات التالية مع تسليمها في نهاية التجربة.



Ultrasonic Distance Measurement Using Python

تصميم مقياس المسافة بطريقة أخرى:

```
1  import time
2  import RPi.GPIO as GPIO
3
4
5  GPIO_TRIGGER = 11
6  GPIO_ECHO    = 15
7
8  GPIO.setmode(GPIO.BOARD)
9  print ("Ultrasonic Measurement")
10 GPIO.setup(GPIO_TRIGGER,GPIO.OUT)  # Trigger
11 GPIO.setup(GPIO_ECHO,GPIO.IN, pull_up_down= GPIO.PUD_DOWN)
12 GPIO.output(GPIO_TRIGGER, False)
```

Ultrasonic Distance Measurement Using Python

```
13  def measure():
14      GPIO.output(GPIO_TRIGGER, True)
15      time.sleep(0.00001)
16      GPIO.output(GPIO_TRIGGER, False)
17      start = time.time()
18      GPIO.wait_for_edge(GPIO_ECHO, GPIO.RISING)
19      start = time.time()
20      GPIO.wait_for_edge(GPIO_ECHO, GPIO.FALLING)
21      stop = time.time()
22      elapsed = stop - start
23      distance = (elapsed * 34300)/2
24      return distance
```

Ultrasonic Distance Measurement Using Python

```
25  def measure_average():
26      distance1=measure()
27      time.sleep(0.1)
28      distance2=measure()
29      time.sleep(0.1)
30      distance3=measure()
31      distance = distance1 + distance2 + distance3
32      distance = distance / 3
33      return distance
```

```
34  distance = measure_average()
35  print ("Distance (cm) : %.1f" % distance)
36  time.sleep(1)
37  GPIO.cleanup()
```

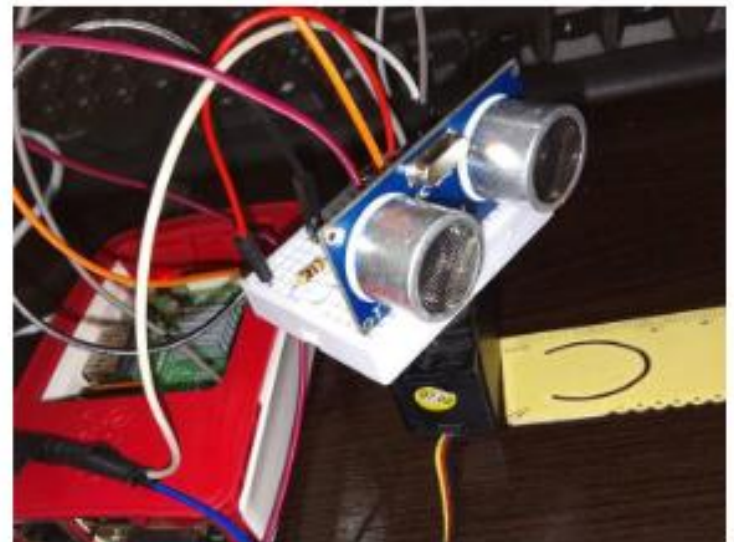
تصميم رادار بسيط باستخدام حساس المسافة ومحرك سيرفو:

```
1 import RPi.GPIO as GPIO
2 import time
3
4 GPIO.setmode(GPIO.BOARD)
5 # Servo position control func
6 def update(angle):
7     duty = float(angle) / 10 + 2.5
8     pwm.ChangeDutyCycle(duty)
9 #IO pin numbers
10 TRIG = 11
11 ECHO = 13
12 SERVO = 37
13 LED = 15
```

ارسم مخطط
لخوارزمية
العمل




```
19 GPIO.setup(TRIG, GPIO.OUT)
20 GPIO.setup(ECHO, GPIO.IN)
21 GPIO.setup(LED, GPIO.OUT)
22 GPIO.setup(SERVO, GPIO.OUT)
23
24 pwm = GPIO.PWM(SERVO, 100)
25 pwm.start(5)
26
27 GPIO.output(TRIG, False)
28 #Initialise variables
29 direction = True
30 angle = 0
31 pos = 0
32 pos_prev = 17
33 pulse_start = 0
34 pulse_end = 0
35 b = [ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ]
36 c = [ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ]
```



```
39 while True :
40     #Send trigger (start) pulse to US sensor
41     GPIO.output(TRIG, True)
42     time.sleep(0.00001)
43     GPIO.output(TRIG, False)
44
45     #Wait until RX signal 0 or max delay
46     timeout = 0
47     while (GPIO.input(ECHO)==0):
48         timeout = timeout + 1
49         pulse_start = time.time()
50
51     #Wait for RX signal
52     while GPIO.input(ECHO)==1:
53         pulse_end = time.time()
54
55     #Calculate time of flight
56     pulse_duration = pulse_end - pulse_start
```

```
58     #convert time to distance
59     distance = pulse_duration * 17150
60     distance = round(distance, 2)
61
62     #Trap if no pulse recieved
63     if distance < 0:
64         distance = 0
65
66     distance = distance * 10
67
68     #Set max distance for display
69     if distance > 900:
70         distance = 900
71
72     if distance > 200:
73         GPIO.output(LED, True)
74         time.sleep(2)
75         GPIO.output(LED, 0)
76     print ("Distance: ", distance, "mm", "Angle: ", pos)
```

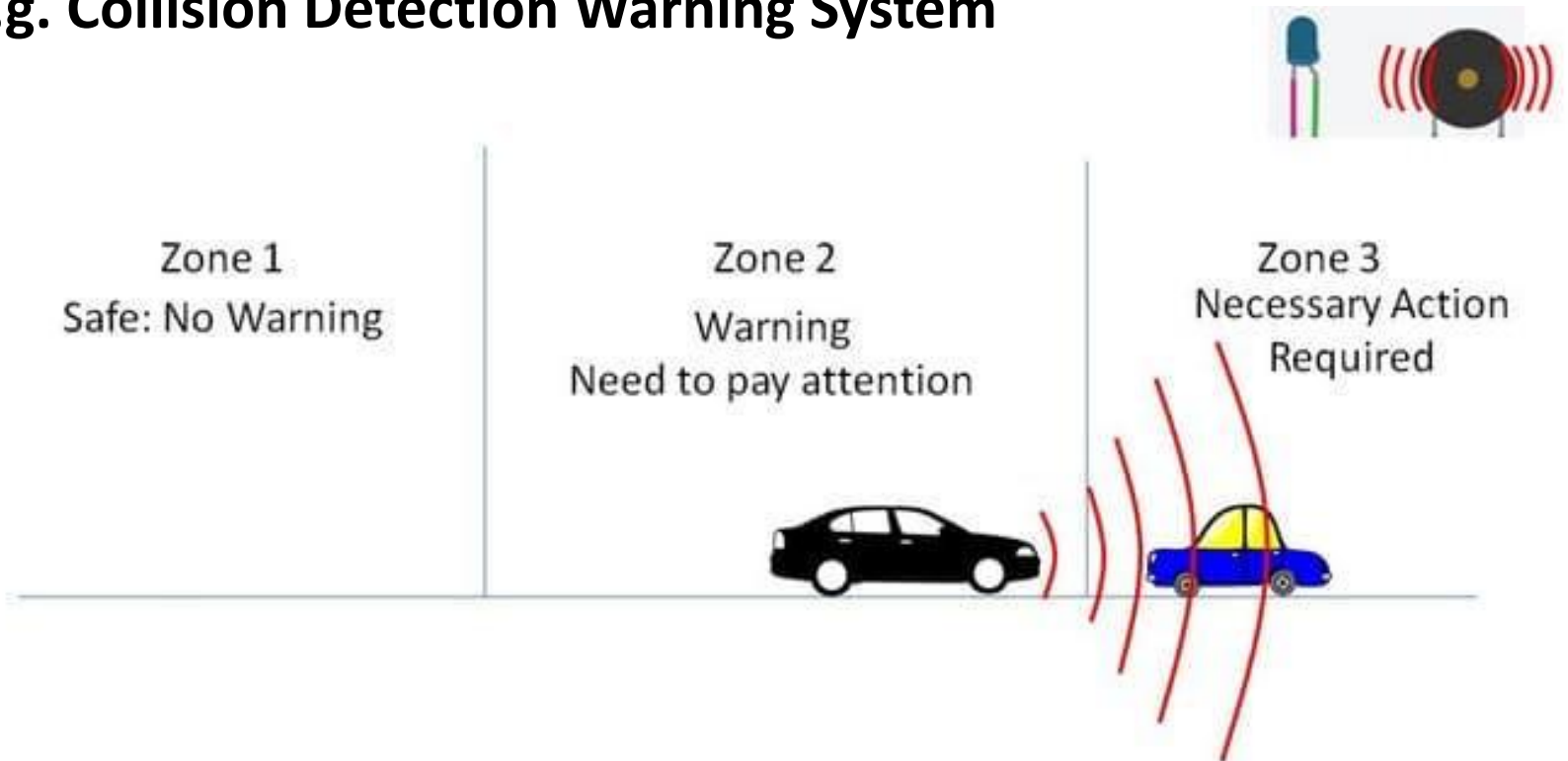
```
78      #Move sensor head
79      update(angle)
80      time.sleep(2)
81
82      #update log file
83      file = open('plot_1.dat', 'w')  #dat, txt , ....
84      outputString = "\n"
85
86      #this array contains a single value
87      #i.e. scan line on plot
88      c[pos] = 600
89      c[pos_prev] = 0
```

```
91      #update distance string with new value
92      for i in range(0,18):
93          if pos == i:
94              b[i] = distance
95
96              outputString = outputString + str(i*10) +
97                  "\t" + str(b[i]) + "\t" + str(c[i]) +
98                  "\n"
99
100     #write data to file
101     file.write(outputString)
102     file.close
103
104     #update scan direction and position
105     pos_prev = pos
```

```
107         if direction:
108             angle = angle + 10
109             pos = pos + 1
110         else:
111             angle = angle - 10
112             pos = pos - 1
113
114         if angle > 180:
115             direction = False
116             angle = 170
117             pos = 17
118
119         if angle < 0:
120             direction = True
121             angle = 10
122             pos = 1
```

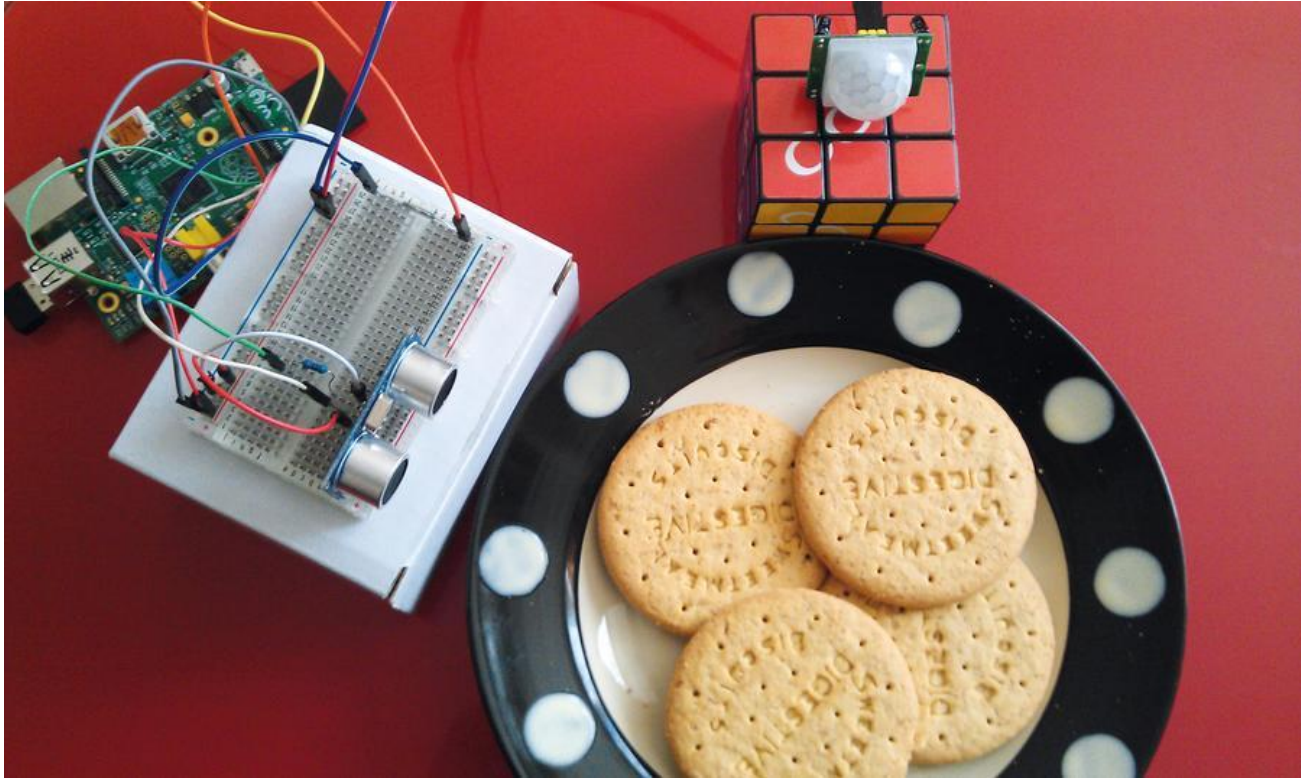
تطبيقات باستخدام حساس المسافة

e.g. Collision Detection Warning System

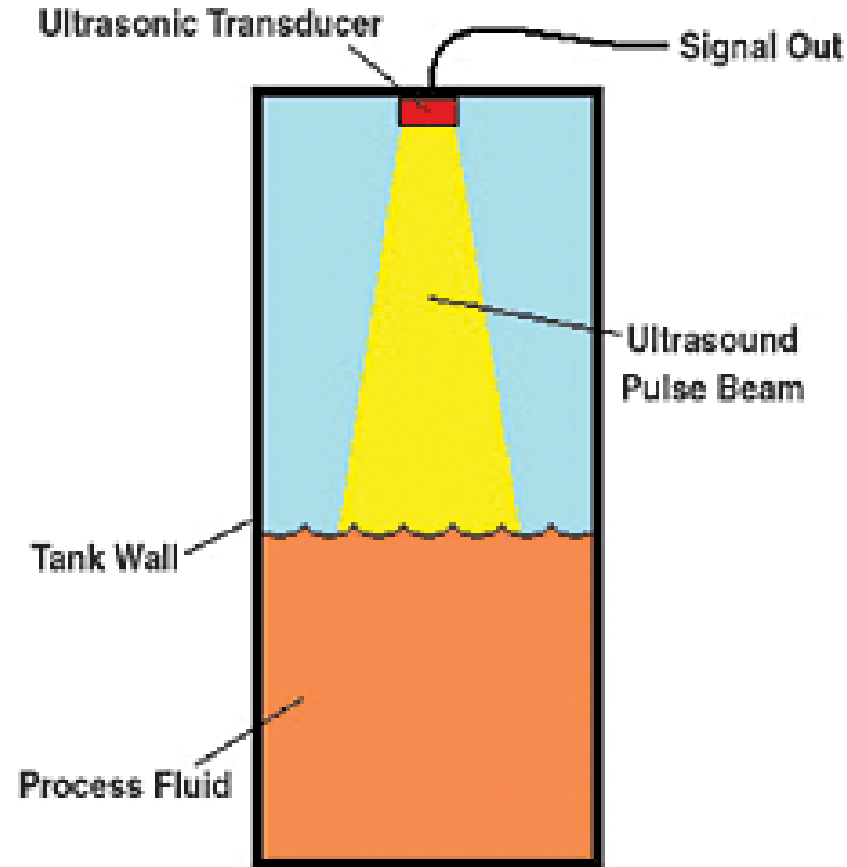


تطبيقات باستخدام حساس المسافة

Applications for Kids :



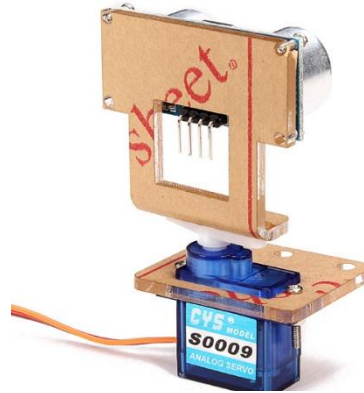
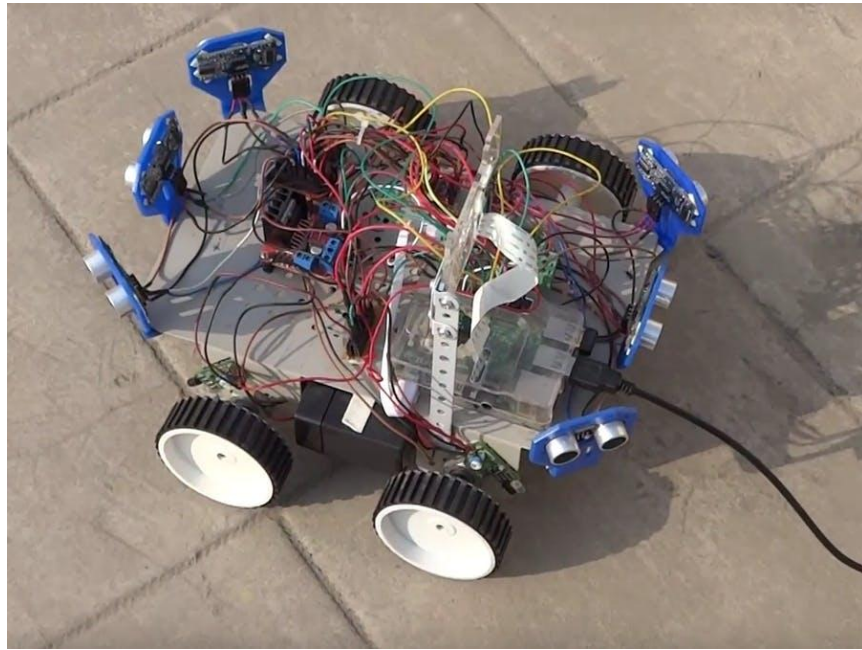
تطبيقات باستخدام حساس المسافة



مقياس مستوى سائل

تطبيقات باستخدام حساس المسافة

Robotics Applications :



تطبيقات باستخدام حساس المسافة

تصميم وتنفيذ نظام توجيه ناطق لمساعدة المكفوفين
Design and implementation of Talking guide system for blind

إعداد الطالب المهندس - أحمد تسقية

المشرف الرئيسي

د. عبد الإله ناولو

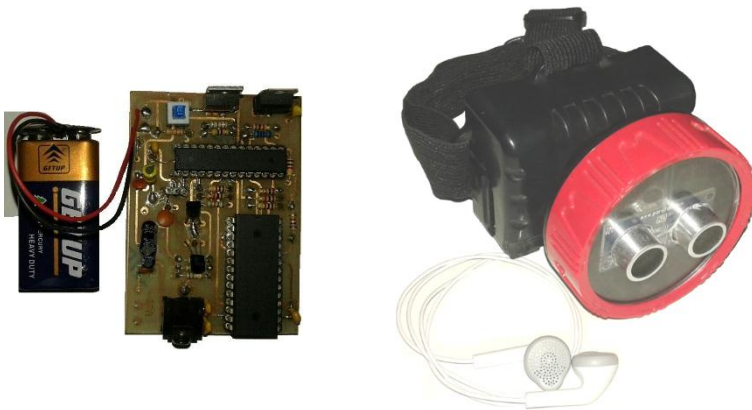
استثمار الحساس في أنظمة المساعدة الطبية

في هذا العمل في البداية تم اعتماد على أن سرعة انتشار الأمواج فوق الصوتية في الهواء ثابتة وهي 340 m/s وبناءً عليه تم حساب المسافة بالاعتماد على العلاقة التالية:

$$D = C_{air} \times T_d / 2$$

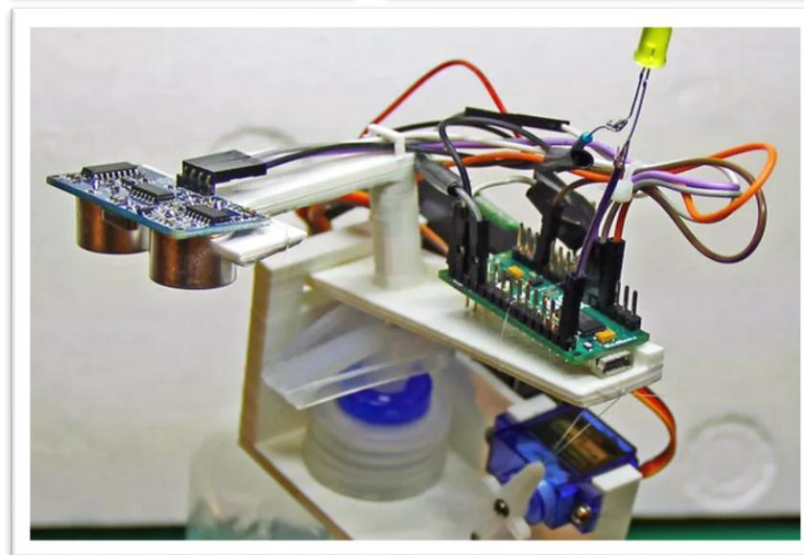
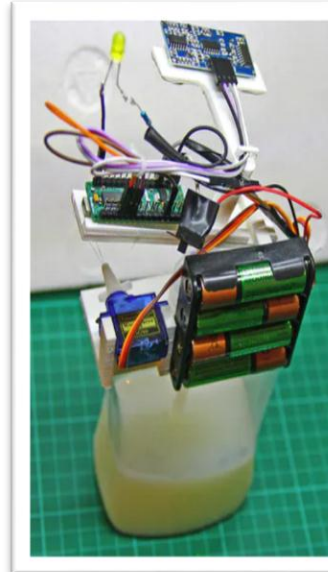
ثم تم اختبار العلاقة الأساسية لسرعة الانتشار المتعلقة بدرجة الحرارة وفق العلاقة التالية:

$$C_{air} = 331.5 + (0.6 \times T_c) \text{ [m/s]}$$



تطبيقات باستخدام حساس المسافة

Automatic Hand Sanitizer



تطبيقات باستخدام حساس المسافة

Wearable Tools



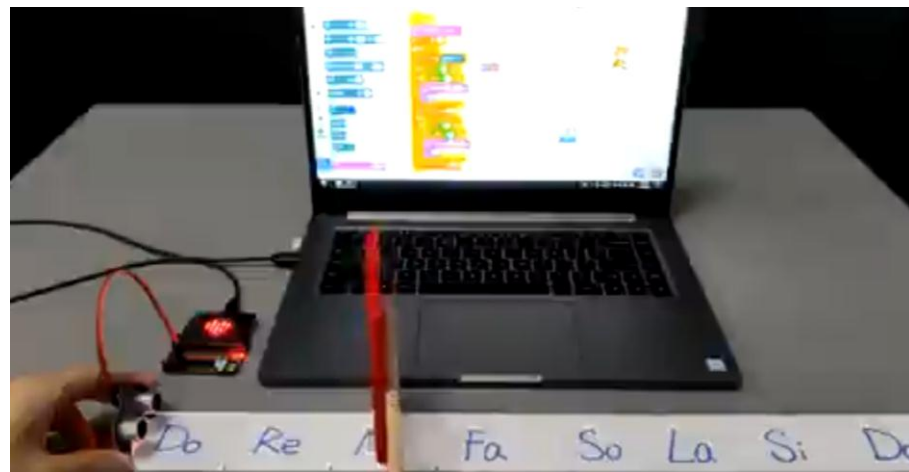
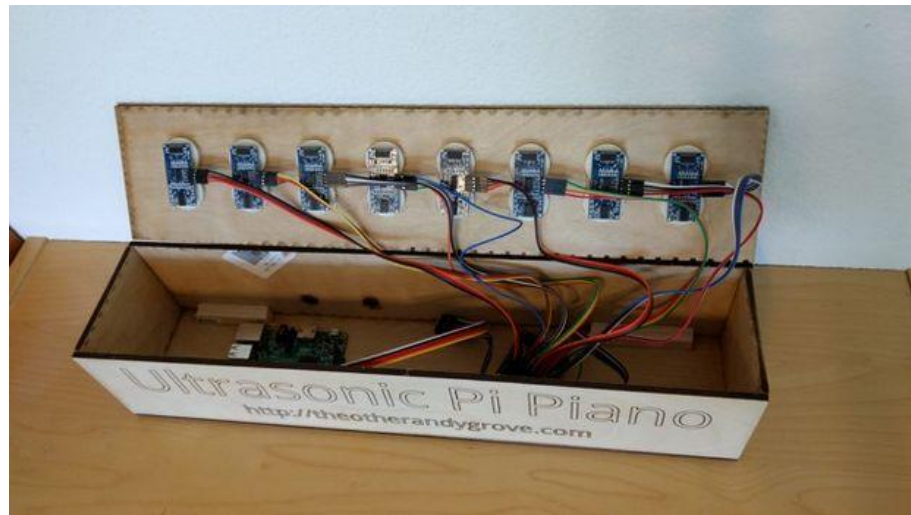
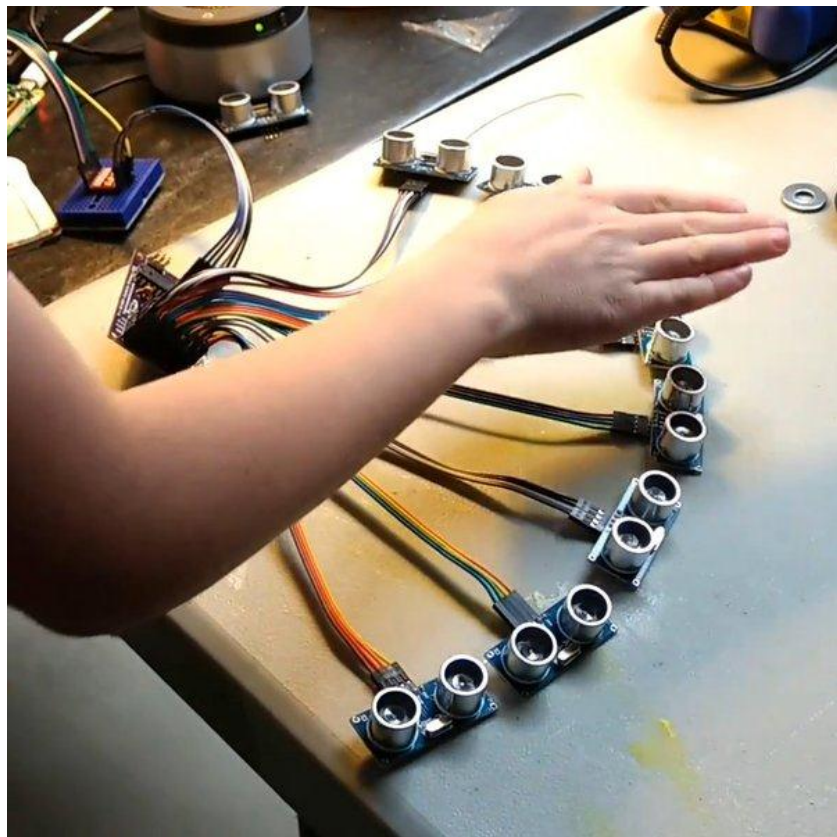
Ultrasonic Glasses for the Blind

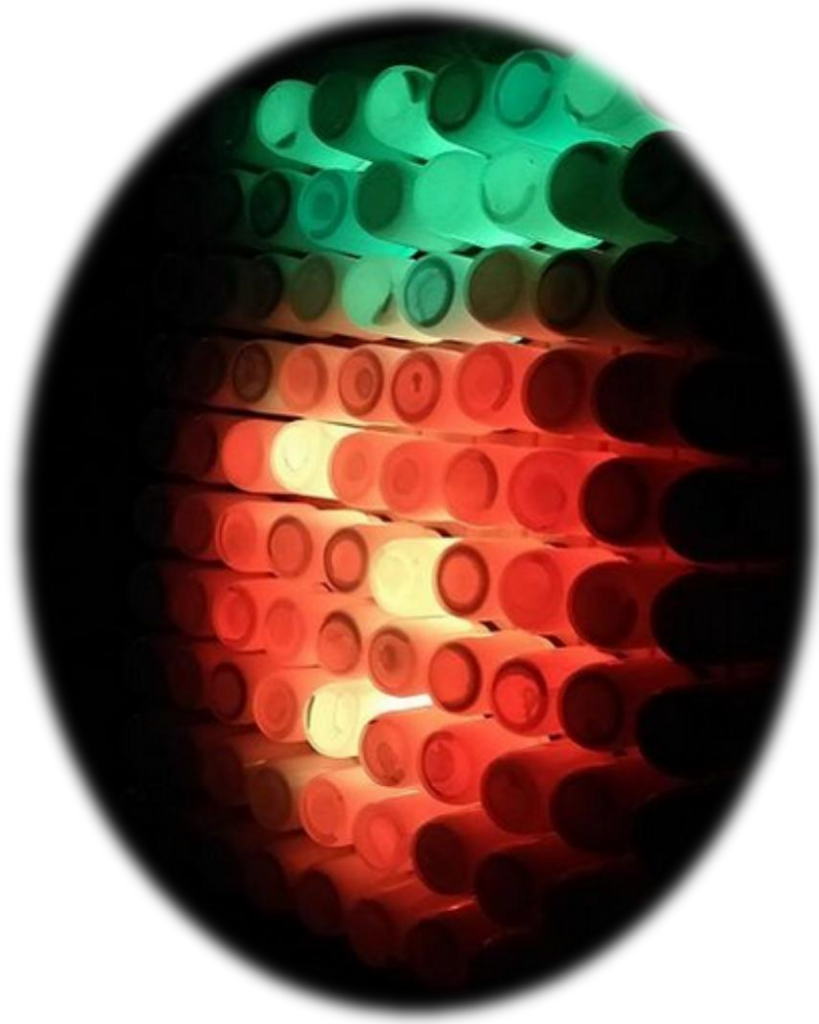


Covid 19 Distance Monitor

تطبيقات باستخدام حساس المسافة

Musical instruments(Theremin, Air Piano , ...)





Thanks

