



William Law CE Primary School  
Learning Living Loving Together

## **INSTRUCTION MANUAL TO MAKE A DOOREEN**

- Building the Circuit
- Amending the code to send a message from the Raspberry Pi to a Microbit attached by USB
- Creating receiving code on the attached Microbit
- Amending the Microbit code to send a radio message to another unconnected Microbit
- Creating receiving code on the unattached Microbit
- Designing a box to store Dooreen with Tinkercad
- Checking the Tinkercad design in Netfabb
- 3D printing the Dooreen Box
- Dooreen in Action

### **Building the Circuit**

You will need a Raspberry Pi and components to build a sensor circuit. We already had a CamJam Sensor Edukit but you can purchase one for £8 from The PiHut and download free worksheets. We used worksheet 5 to build a circuit.

## CamJam EduKit #2 - Sensors

Delivery  
from just  
£2.50



20,000+  
customer  
reviews



No Quibble  
returns  
policy



Secure  
payment  
system



Next Day  
Delivery  
Order by 2PM\*



In Stock

£8.00

Add to Cart



99 Reviews

1 Questions \ 1 Answers

Share this Product

Share

Tweet

+1

## CamJam EduKit Sensors Worksheet Five

### Project Passive Infrared Sensor

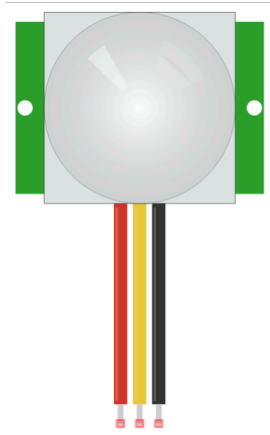
**Description** In this project, you will learn how to wire and program a passive infrared sensor that detects movement near it.

### Equipment Required

- Your Raspberry Pi
- 400 Point Breadboard
- Passive Infrared Sensor
- 6 x m/f jumper wires

# The Parts

## The Passive Infrared Sensor



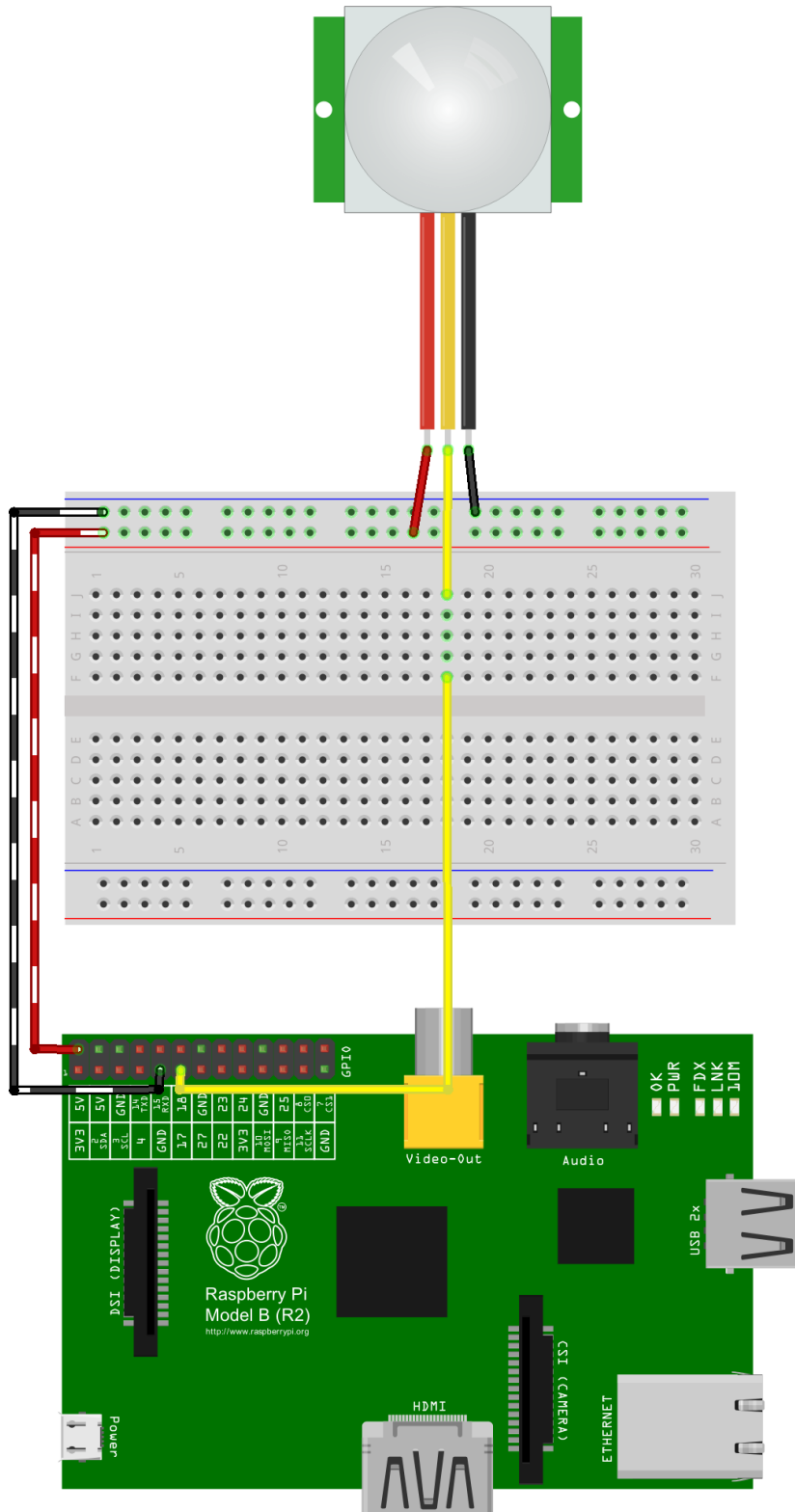
The main component of this circuit is itself another circuit board that has a PIR, or Passive Infrared sensor on it. These devices are commonly used in burglar alarms, lights that come on when people approach, and some CCTV cameras.

There are three connectors on the bottom of the PIR, marked VCC, OUT and GND. A 5-volt power supply is applied to VCC pin, with GND pin going to 'ground'. The OUT pin will 'go high' when movement is detected. You will notice two 'potentiometers' on the bottom that are used for adjusting the sensitivity (marked Sx) and how long the sensor pin stays high when it senses motion (marked Tx). To make the PIR more sensitive, turn the Sx potentiometer clockwise with a small screwdriver. To start with, you should set it to the middle. You may want to experiment with the Tx potentiometer once you have written the code. However, to start with you should turn it all the way anti-clockwise to make the PIR report movement for the shortest time.

The diagram below shows how to connect the PIR sensor. The PIR circuit is much simpler than the other circuits, mainly because the sensor contains a large amount of its own circuitry.

Power is supplied from the 5v pin, and not the 3.3v that the other circuits use.

Use three jumper wires to connect the PIR pins to the breadboard. The power input pin is marked 'VCC', the negative marked with 'GND', and the sensor pin with 'OUT'. A second jumper wire connects pin 17 to the breadboard.



Open the Python IDLE editor and type in the following code:

```
# Import Python header files
import RPi.GPIO as GPIO
import time

# Set the GPIO naming convention
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
# Set a variable to hold the GPIO Pin identity
PinPIR = 17
print("PIR Module Test (CTRL-C to exit)")
# Set pin as input
GPIO.setup(PinPIR, GPIO.IN)
# Variables to hold the current and last
states
Current_State = 0
Previous_State = 0
try:
    print("Waiting for PIR to settle ...")
    # Loop until PIR output is 0
    while GPIO.input(PinPIR)==1:
        Current_State = 0
    print("  Ready")
    # Loop until users quits with CTRL-C
    while True:
        # Read PIR state
        Current_State = GPIO.input(PinPIR)
        # If the PIR is triggered
        if Current_State==1 and
Previous_State==0:
            print("  Motion detected!")
            # Record previous state
            Previous_State=1
            # If the PIR has returned to ready
state
            elif Current_State==0 and
Previous_State==1:
```

```
        print("  Ready")
        Previous_State=0
        # Wait for 10 milliseconds
        time.sleep(0.01)
except KeyboardInterrupt:
    print("  Quit")
# Reset GPIO settings
GPIO.cleanup()
```

Once complete, save the file.

## Running the Code

Select the Run Module menu option, under the Run menu item. Alternatively, you can just press the F5 key.

When the PIR detects movement, it will print 'Motion detected!' on the screen once and once only. If the movement stops it will return to the steady state.

## How the Code Works

The code above introduces a few concepts that may not have been used in the previous worksheets. Let's take a look at some parts of the code that you may not be familiar with. The whole code is not repeated in full below, just the parts that are of interest.

```
PinPIR = 17
```

A variable, *PinPIR*, is being used to store the pin number of the PIR sensor pin. This allows you to change which pin is used in only one place in the code, and makes it easier to code by not having to remember the pin number, just the pin name you have given it.

```
try:
```

The main code is contained within a *try.except* construct. The

code within the *try* will continue to be run until the *KeyboardInterrupt* keys are pressed. This is a special key combination that is defined within Python that will interrupt a program when pressed. For the Raspberry Pi, this is 'Ctrl + c', which is pressing the Ctrl key down and pressing the 'c' key.

```
while GPIO.input(PinPIR)==1:  
    Current_state = 0
```

In the first *while* loop after the *try*, the code first waits until the PIR does not see any movement. The *Current\_State* variable is set to 0, indicating no movement.

```
while True:
```

The code then enters an 'eternal' loop; *while True:* means that the loop will always run unless the interrupt keys are pressed.

```
Current_State = GPIO.input(PinPIR)
```

The *Current\_State* is then set to the value of the input pin. If there is no movement, this will be 0. If there is movement, this will be 1.

```
#If the PIR is triggered  
if Current_State==1 and Previous_State==0:  
    print("Motion detected!")  
    #Record previous state  
    Previous_State=1
```

If the PIR has been triggered, but on the last check it was not, then you will be notified by the message "Motion detected!". The 'previous state' will then be set to show that motion has been detected.

```
elif Current_State==0 and Previous_State==1:  
    print =(" Ready")  
    Previous_State=0
```

If the current state shows that there is no movement, but the previous state shows that there was movement, then you will be notified that everything is still around the sensor with the

message "Ready".

```
#Wait for 10 milliseconds  
time.sleep(0.01)
```

The code then sleeps for 0.01 of a second. This is here to stop the code from continuously flipping between seeing movement and not seeing movement.

```
Except KeyboardInterrupt:  
    print("Quit")  
  
    #Reset GPIO settings  
    GPIO.cleanup()
```

If the interrupt keys are pressed (Ctrl+c), the program will end, but before it does, the GPIO pins will be reset to their default state.



# MICROBITS

You will need two Microbits. We borrowed Microbits from our Code Club but you can buy them from websites such as Pimoroni for £13 each

£0  
0 items [View](#)

Checkout

- Raspberry Pi 3
- micro:bit**
- Arduino & Co.
- Kits
- Electronics
- Prototyping
- Tools
- Kids
- More...
- NEW!

## micro:bit

	micro:bit Complete Starter Kit MBIT0001	In stock	
	£37.50 / <a href="#">view more details</a>		
	micro:bit Prototyping Kit MBIT0002	17 in stock	
	£28.50 / <a href="#">view more details</a>		
	micro:bit Essentials Kit MBIT0003	In stock	
	£16		
	micro:bit only MBIT0004	In stock	
	£13		

The BBC micro:bit is a pocket-sized codeable computer with motion detection, a built-in compass, LED display, and Bluetooth technology built in.



## Amending the Code to send a message to an Attached Microbit

We had to `import serial` so the Raspberry pi could send a message by USB to an attached Microbit.

We then had to give instructions on how they would communicate by adding these lines of code:

```
#set up usb port communication
PORT = "/dev/ttyACM0"
BAUD = 115200
s = serial.Serial(PORT)
s.baudrate = BAUD
s.parity = serial.PARITY_NONE
s.databits = serial.EIGHTBITS
```

```
s.stopbits = serial.STOPBITS_ONE
s.readline()
```

We then added the following code to when the PIR sensed movement so that as well as print motion detected so we knew it had worked it send a message 'hello' to the Microbit.

```
s.write("hello".encode('utf-8'))
```

## The Raspberry Pi complete code with all parts included.

```
# Import Python header files
import RPi.GPIO as GPIO
import time
import serial
# Set the GPIO naming convention
GPIO.setmode(GPIO.BCM)
GPIO.setwarnings(False)
# Set a variable to hold the GPIO Pin identity
PinPIR = 17
#set up usb port communication
PORT = "/dev/ttyACM0"
BAUD = 115200
s = serial.Serial(PORT)
s.baudrate = BAUD
s.parity = serial.PARITY_NONE
s.databits = serial.EIGHTBITS
s.stopbits = serial.STOPBITS_ONE
s.readline()

print("PIR Module Test (CTRL-C to exit)")
# Set pin as input
GPIO.setup(PinPIR, GPIO.IN)
# Variables to hold the current and last
states
Current_State = 0
Previous_State = 0
```

```
try:
    print("Waiting for PIR to settle ...")
    # Loop until PIR output is 0
    while GPIO.input(PinPIR)==1:
        Current_State = 0
    print("  Ready")
    # Loop until users quits with CTRL-C
    while True:
        # Read PIR state
        Current_State = GPIO.input(PinPIR)
        # If the PIR is triggered
        if Current_State==1 and
Previous_State==0:
            print("  Motion detected!")
            s.write("hello".encode('utf-8'))
            time.sleep(1)

            # Record previous state
            Previous_State=1
        # If the PIR has returned to ready
state
        elif Current_State==0 and
Previous_State==1:
            print("  Ready")
            Previous_State=0
            # Wait for 10 milliseconds
            time.sleep(0.01)
except KeyboardInterrupt:
    print("  Quit")
    # Reset GPIO settings
    GPIO.cleanup()
```

# Microbit receiving Raspberry Pi Message and sending new message to unconnected Microbit

We created this code in Mu, saved it and then flashed it to the attached Microbit.

```
from microbit import *
import radio
radio.on()

visitor = ('caller')
caller = {'hello': visitor}

def get_sensor_data():

    a, b = button_a.was_pressed(),
    button_b.was_pressed()

    print(a, b)

while True:
    sleep(500)
    get_sensor_data()

    try:
        bytestring = uart.read()
        icon = caller[(str(bytestring))[2:-1]]
        display.show(icon)
        radio.send("caller")

    except:
        pass
        display.clear()
```

This code will read the message sent by the Raspberry Pi by USB and if the message reads 'Hello' then it will display 'Caller' on itself.

The radio.send code line then sends a message “caller” to another Microbit which has already been flashed with a receive code and is connected to a battery pack.

## **Unattached Microbit receiving message from attached Microbit**

We then created this code and flashed it to a separate microbit, which was powered by a battery pack. This microbit can then be worn and it receives the message from the other attached Microbit by radio.

```
from microbit import *
from microbit import display
import radio

radio.on()
while True
    received_text = radio.receive()
    if (received_text == 'caller'):
        display.scroll("caller")
        sleep(100)
    else:
        display.clear()
```

When it receives the message ‘caller’ it displays caller.

## **Tinkercad**

<https://www.tinkercad.com>

You can sign up for a free account to design 3D objects. We have Google Education accounts so we used this to sign up. The website has lots of tutorials to show you how to create objects.

Once we had created a box we downloaded the file as STL as this is the type of file our 3D printer prints with.

## Checking in Netfabb

<https://www.netfabb.com/try-netfabb-premium-now>

Sometimes our designs do not always print and so we downloaded this software, we only have the basic version. This has an automatic repair button which checks there are no holes etc and corrects any faults.

We then export the repaired part in STL.

## 3D Printing

[\*\*http://robox.cel-uk.com\*\*](http://robox.cel-uk.com)

We have a Robox 3D printer. It comes with it's own AutoMaker software that we add our STL model to and it confirms whether it can print the item. If all is well, it will print our object.

## Dooreen in Action

You need to make sure the correct Mu code has been flashed to the correct Microbit.

Run the Raspberry Pi Python program and it will continue to respond if motion is detected until you quit.