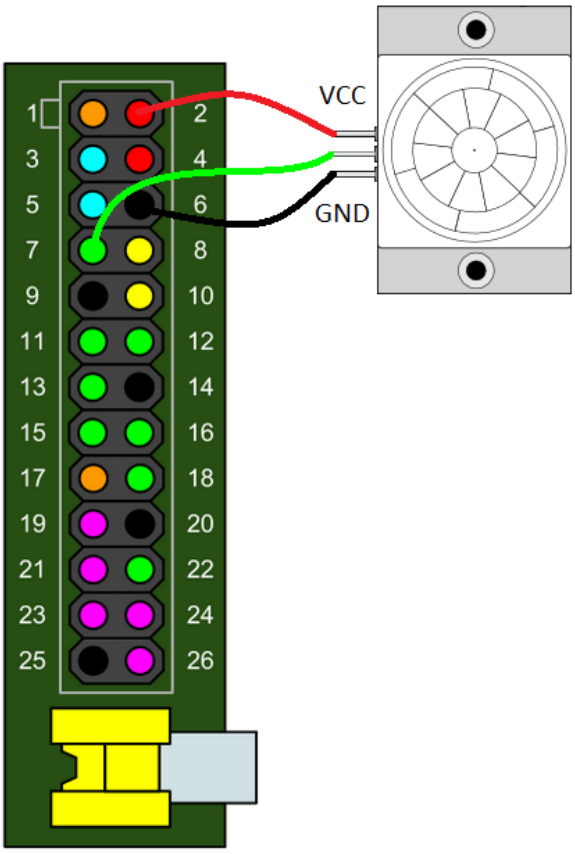
Make an infrared Santa detector

You will build a detector that spots any movement and sets off an alarm. Along the way you will learn how to connect a passive infrared (PIR) sensor to the Raspberry Pi, and how to control the flow of your Scratch program by responding to the input from the sensor.

Introduction: What is a PIR module?

PIR stands for Passive Infra Red. You might have seen these before, as they are very common these days. You would most often find them in the corners of rooms for burglar alarm systems. All objects whose temperatures are above absolute zero emit infrared radiation. Infra red wavelengths are not visible to the human eye, but they can be detected by the electronics inside one of these modules.

The sensor is regarded as passive because it doesn't send out any signal in order to detect movement. It adjusts itself to the infrared signature of the room it's in and then watches for any changes. Any object moving through the room will disturb the infrared signature, and will cause a change to be noticed by the PIR module.



We don't need to worry about its inner workings. What we're interested in are the three pins on it; we can connect those to the Raspberry Pi GPIO pins. One pin is for +5 volts, one pin is for ground and the other is the sensor pin (the middle pin on our Pi). This sensor pin will receive power whenever motion is detected by the PIR module. We can then see that happening on the Raspberry Pi and take action accordingly.

Step 1: Connect the PIR motion sensor

Before booting your Raspberry Pi, connect the PIR module to the Raspberry Pi.

Refer to the diagram to the right for pin numbers. If you look closely at the pins on your PIR module, you'll see some white text on the PCB

By Andrew Mulholland. Based off Raspberry Pi Foundation HoC resources
Northern Ireland Raspberry Jam

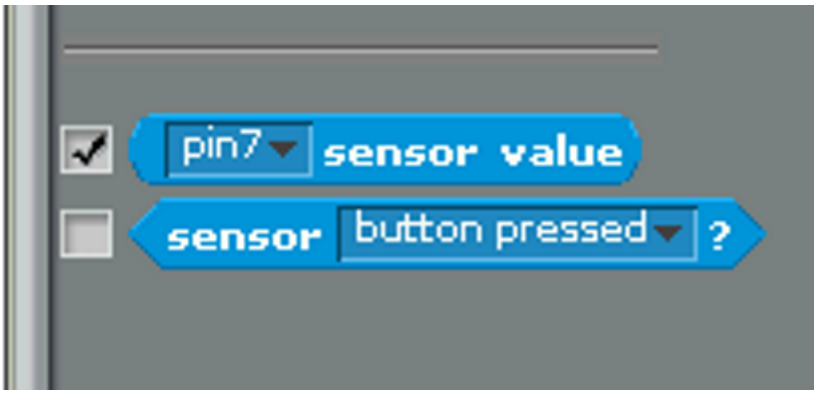
near the base of each one. ***VCC*** is for a +5 volts input. Take one of the female-to-female jumpers and connect the VCC pin to pin 2 on the Pi (coloured red on the diagram); this will make the Pi provide 5 volts of power to the PIR module. Use another jumper to connect ***GND*** on the module to pin 6 on the Pi (coloured black on the diagram); this completes the circuit and allows current to flow back out of the module into ground. Now do the same for the sensor pin ***OUT***; you can use any of the green pins on the Pi for this. I am going to use pin 7, since it's the first general purpose one.

Note: If you have a different PIR module to the one pictured then your pin layout might be different; this is why I include the labels ***VCC, GND*** and ***OUT***.

Step 2: Test the sensor

Because we are using the GPIO pins, we need to run a special version of Scratch called ***Scratch GPIO 5***. Double-click this icon on the desktop to run it.

Scratch uses the 'Sensing' blocks to check if there is any input on the GPIO pins. If there is an input, the value of the pin changes from ***0*** to ***1***. As you connected the PIR sensor to pin 7 of the Pi, we need to monitor that.



- Click on the drop-down menu on the ***sensor value*** block and choose ***pin7***.
- Tick the check box to the left of the block to display the pin value on screen.

Test the PIR sensor by waving your hand in front of it. When it detects movement, the value on the screen should change from ***0*** to ***1***.

If the value doesn't change, check that the correct pins are connected.

Step 3: Draw a sprite and add sound

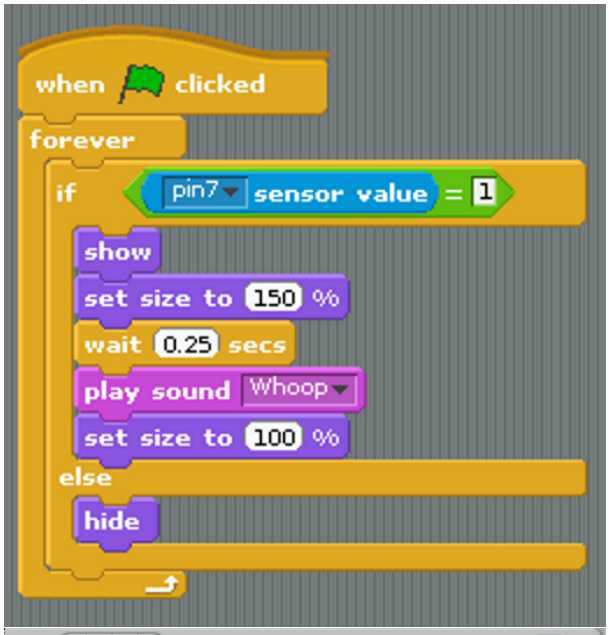
Click on the ***Costumes*** tab and draw a Santa sprite. This will be displayed when the PIR senses movement.

Click on the ***Sounds*** tab and import a sound from the ***Electronic*** folder. I've used a siren called 'Whoop' here.

Step 4: Program what happens when the detector spots movement

Now we have a sensor that reports when it is on or off, we can use this value to control the flow of our program.

Build the following script:



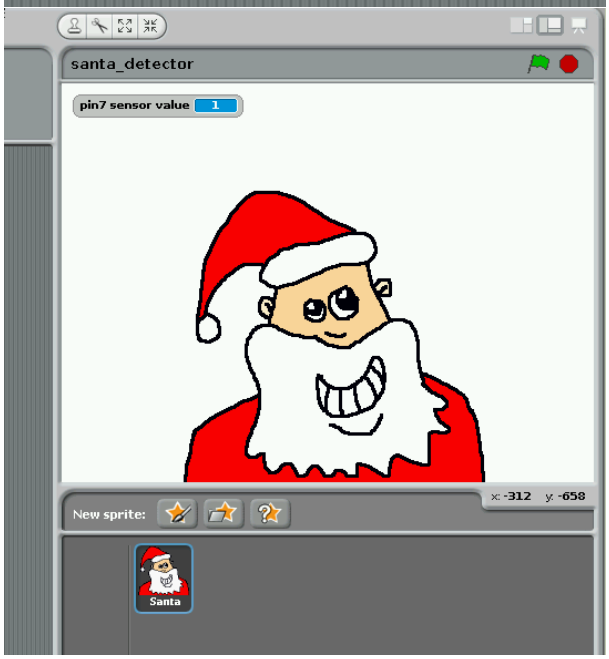
What is the program doing?

The ***if*** statement continuously checks the pin 7 sensor value. When this changes to ***1*** it does the following:

- Shows the sprite
- Makes the sprite bigger
- Waits a bit
- Plays a sound
- Makes the sprite normal size

It keeps doing this as long as the sensor value is ***1*** i.e. when the PIR detects movement.

The ***else*** statement simply hides the sprite when the sensor value is ***0***.



Now what?

Now you can choose what happens when Santa is detected.

Here are a few ideas

- Make the background flash
- Animate the sprite using costumes
- Change the sound ('Santa Claus is coming to town' would be good :))
- Use different graphic effects instead of ***change size***
- Display a message
- Hook up some RGB LEDs