```
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
&lt;/pre&gt;

import RPi.GPIO as GPIO
from time import time

def setup():
GPIO.setmode(GPIO.BOARD) # Numbers GPIOs by physical location
GPIO.setup(11, GPIO.IN, pull_up_down=GPIO.PUD_DOWN)

def binary_aquire(pin, duration):
# aquires data as quickly as possible
t0 = time()
results = []
while (time() - t0) &lt; duration:
results.append(GPIO.input(pin))
return results

def on_ir_receive(pinNo, bouncetime=150):
# when edge detect is called (which requires less CPU than constant
# data acquisition), we acquire data as quickly as possible
data = binary_aquire(pinNo, bouncetime/1000.0)
if len(data) &lt; bouncetime:
return
rate = len(data) / (bouncetime / 1000.0)
pulses = []
i_break = 0
# detect run lengths using the acquisition rate to turn the times in to
microseconds
for i in range(1, len(data)):
if (data[i] != data[i-1]) or (i == len(data)-1):
pulses.append((data[i-1], int((i-i_break)/rate*1e6)))
i_break = i
# decode ( &lt; 1 ms "1" pulse is a 1, &gt; 1 ms "1" pulse is a 1, longer than 2
ms pulse is something else)
# does not decode channel, which may be a piece of the information after the long
```

```
1 pulse in the middle
outbin = ""
for val, us in pulses:
if val != 1:
continue
if outbin and us &gt; 2000:
break
elif us &lt; 1000:
outbin += "0"
elif 1000 &lt; us &lt; 2000:
outbin += "1"
try:
return int(outbin, 2)
except ValueError:
# probably an empty code
return None

def destroy():
GPIO.cleanup()

if __name__ == "__main__":
setup()
try:
print("Starting IR Listener")
while True:
print("Waiting for signal")
GPIO.wait_for_edge(11, GPIO.FALLING)
code = on_ir_receive(11)
if code:
print(str(hex(code)))
else:
print("Invalid code")
except KeyboardInterrupt:
pass
except RuntimeError:
# this gets thrown when control C gets pressed
# because wait_for_edge doesn't properly pass this on
pass
print("Quitting")
destroy()
&lt;pre&gt;
```