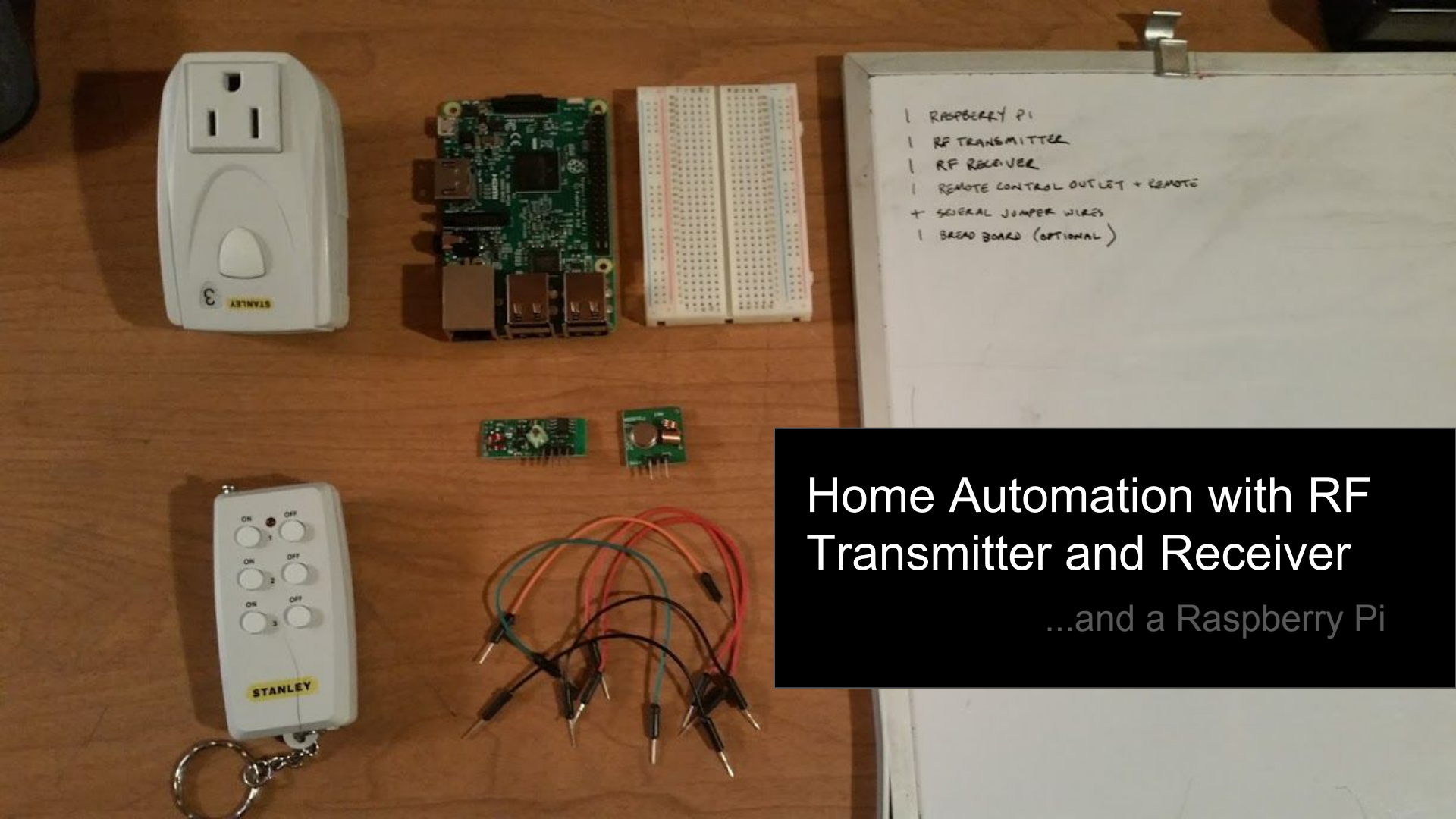
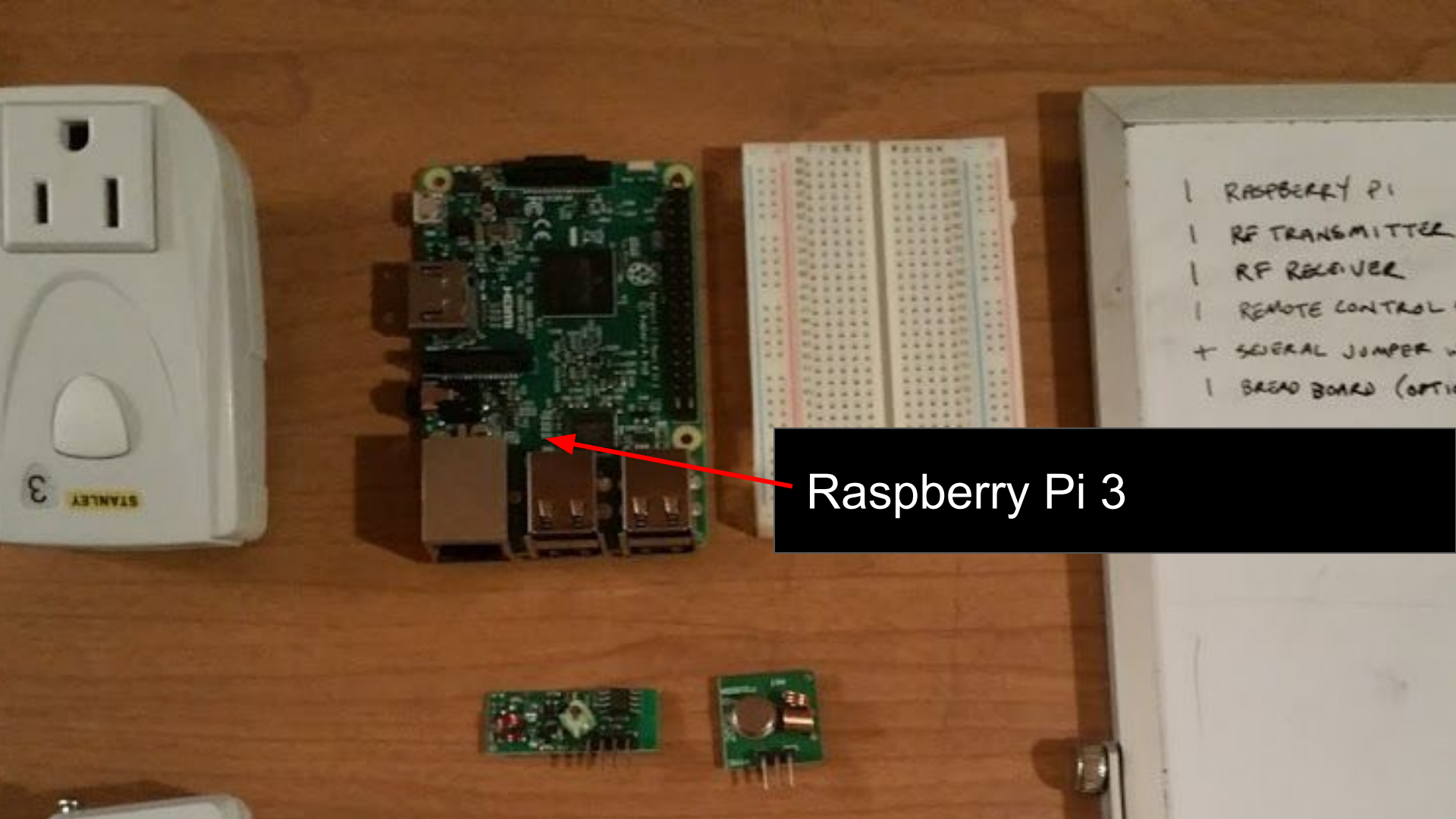
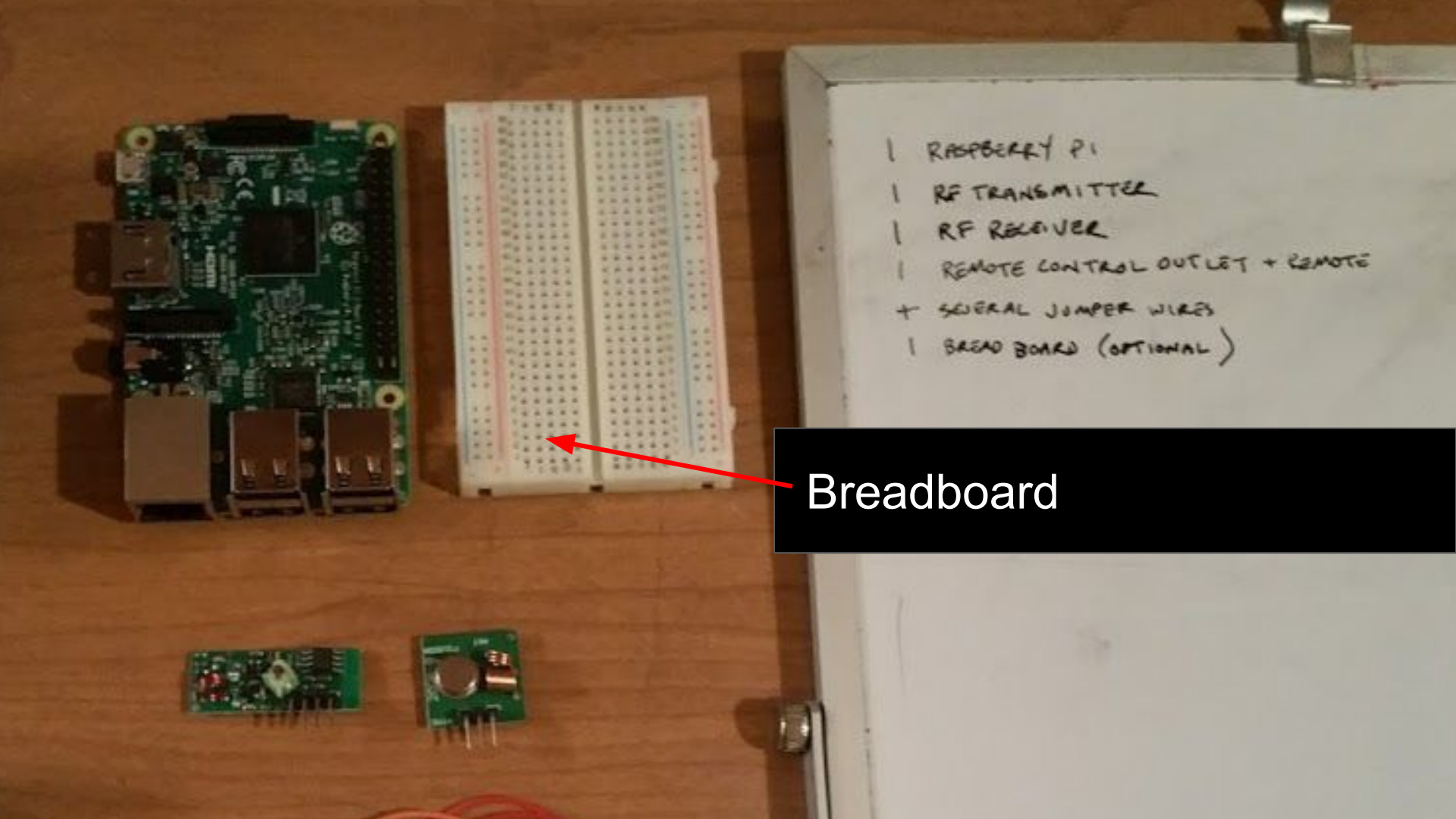
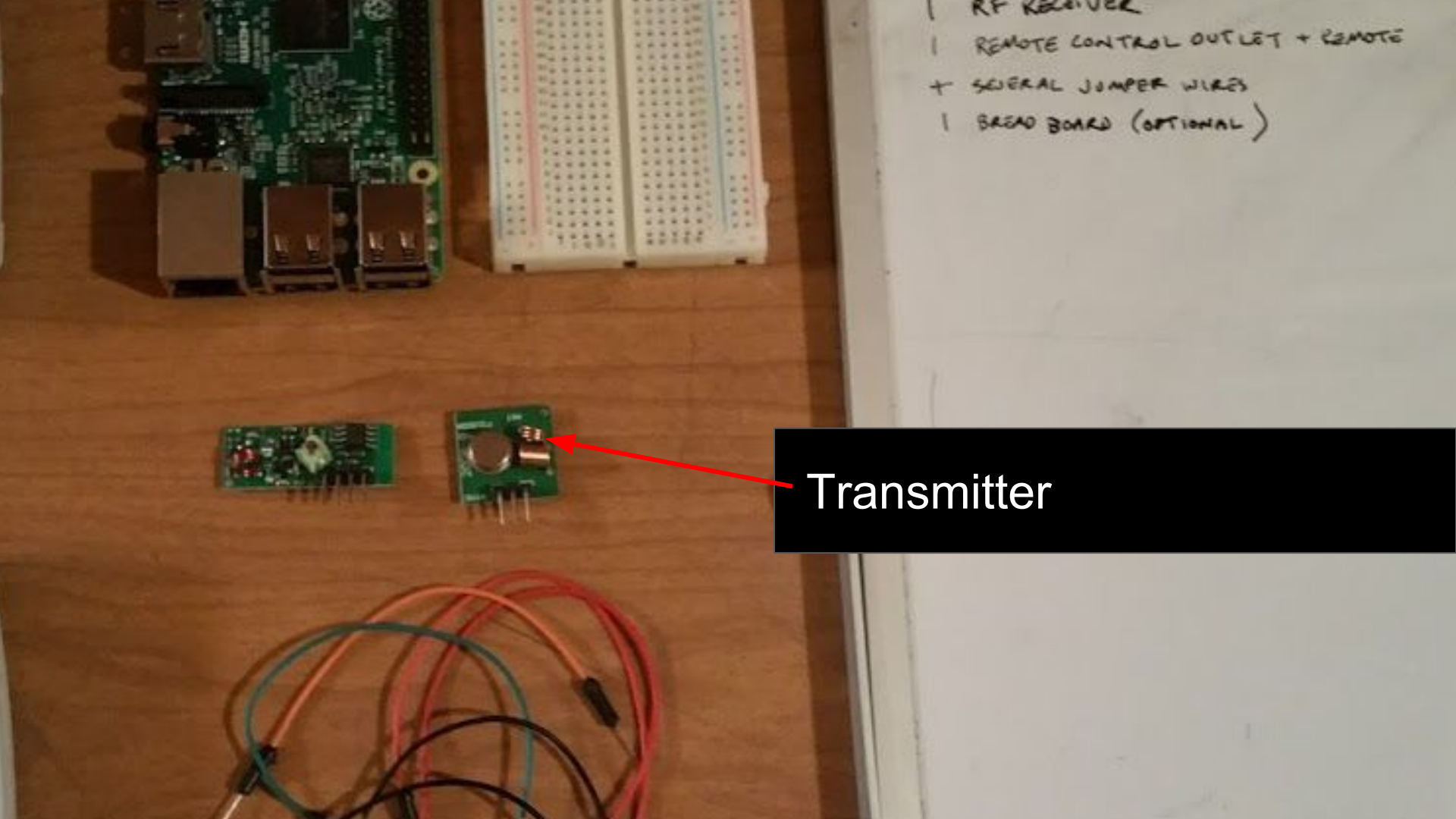


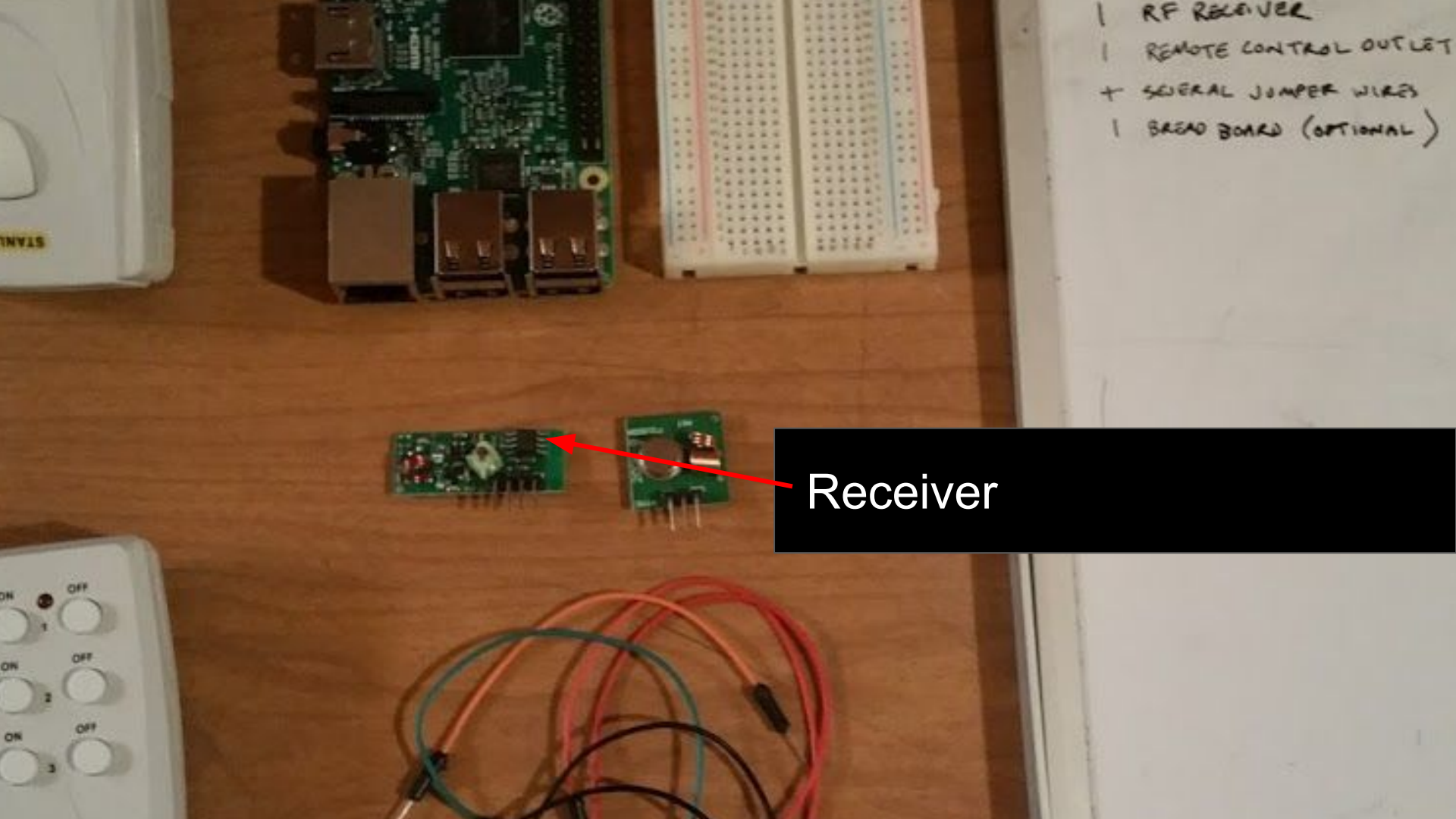

# Home Automation with RF Transmitter and Receiver

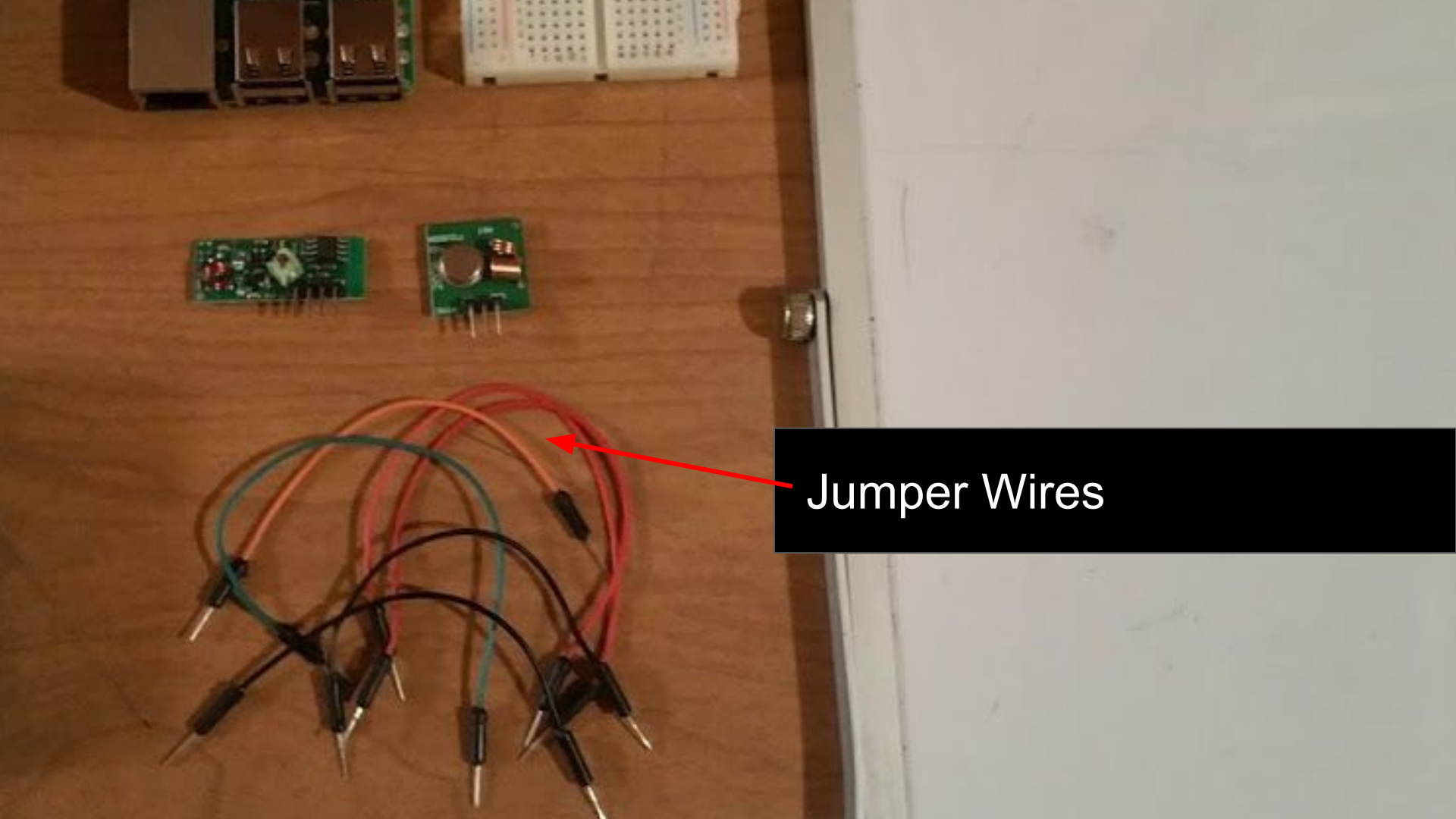
...and a Raspberry Pi

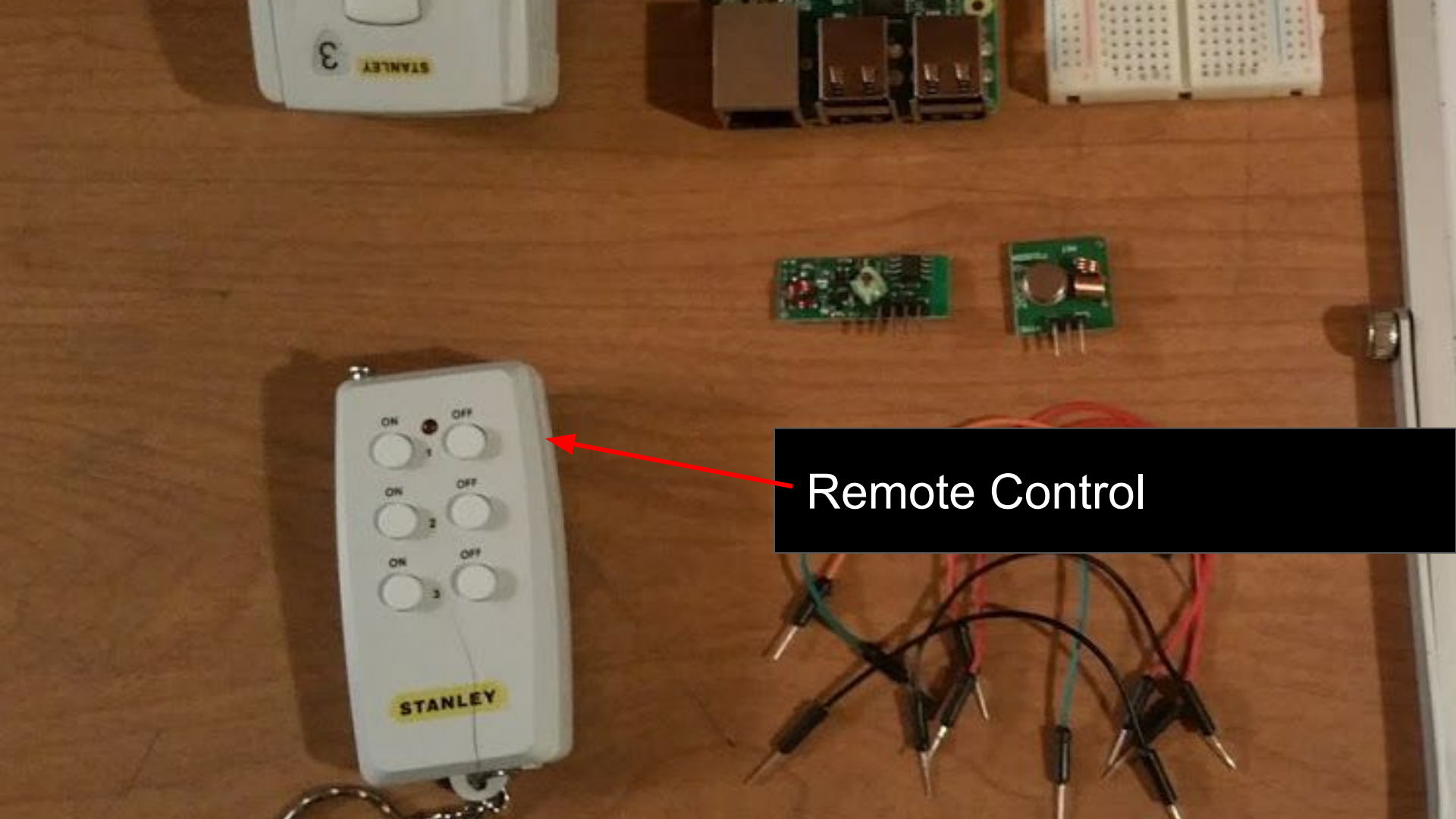

1 RASPBERRY PI
1 RF TRANSMITTER
1 RF RECEIVER
1 REMOTE CONTROL
+ SEVERAL JUMPER
1 BREAD BOARD
STANLEY
Raspberry Pi 3

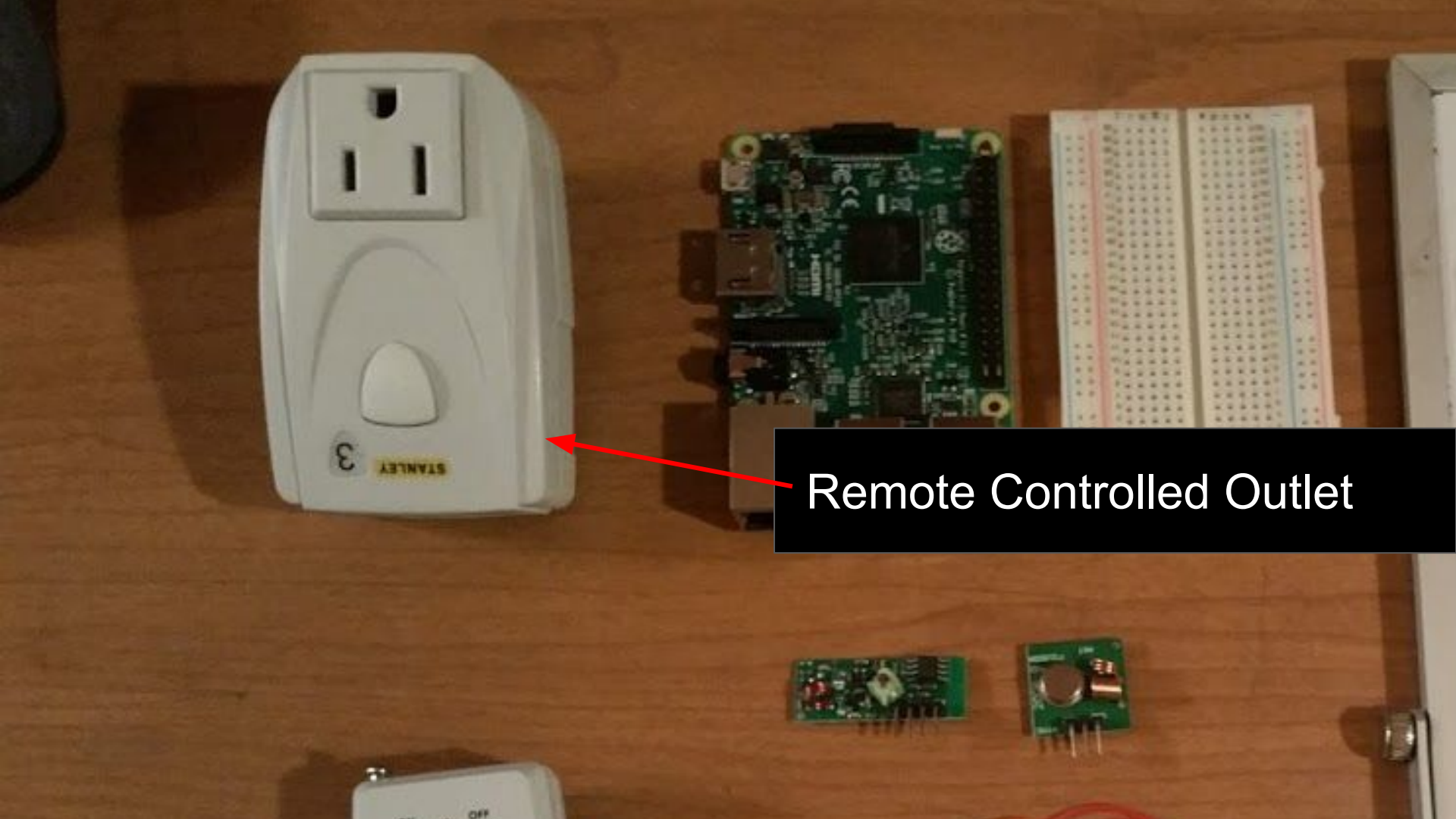

1 RASPBERRY PI
1 RF TRANSMITTER
1 RF RECEIVER
1 REMOTE CONTROL OUTLET + REMOTE
+ SEVERAL JUMPER WIRES
1 BREAD BOARD (OPTIONAL)
Breadboard

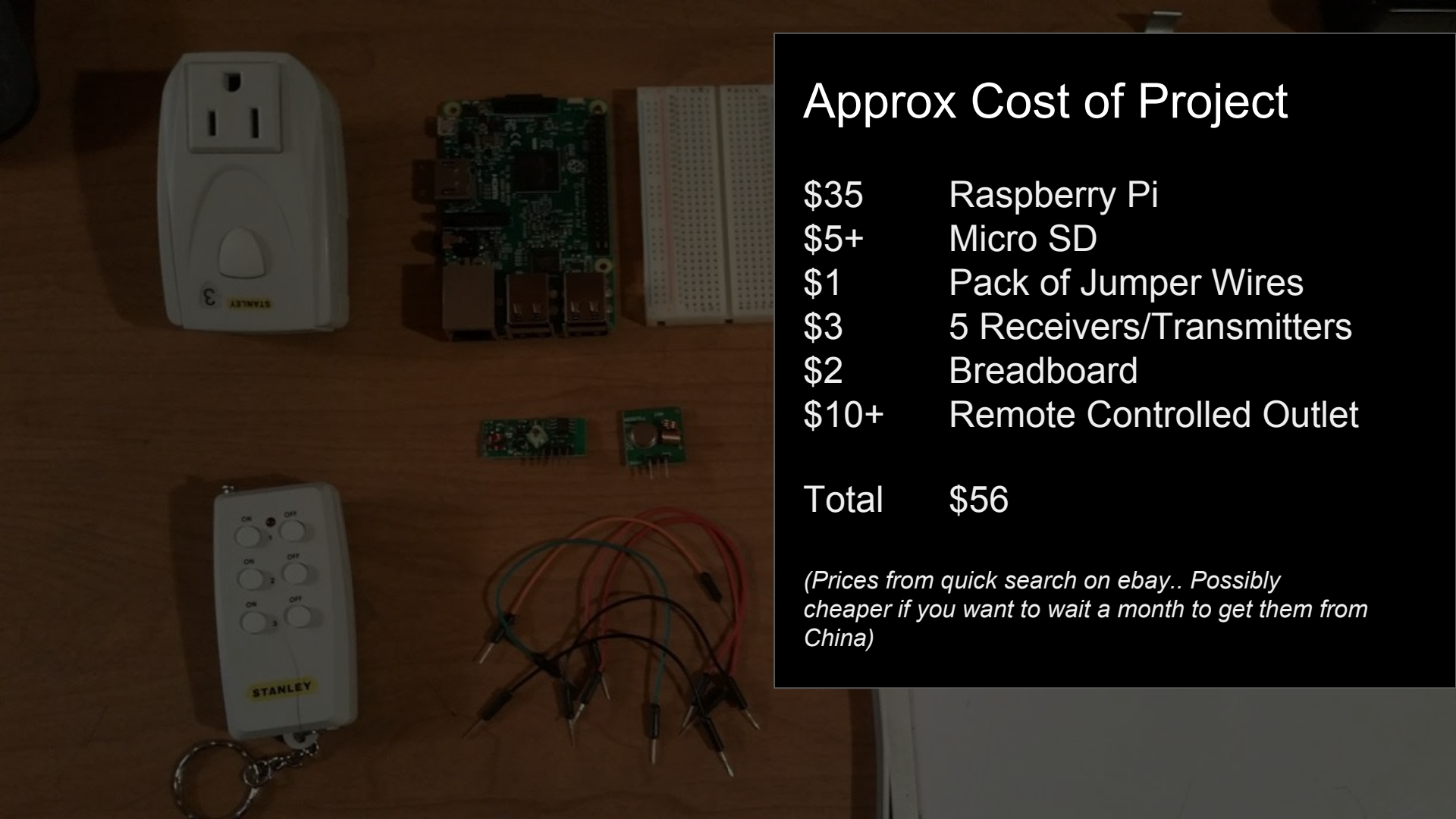

1 RF RECEIVER
1 REMOTE CONTROL OUTLET + REMOTE
+ SEVERAL JUMPER WIRES
1 BREAD BOARD (OPTIONAL)
Transmitter


1 RF RECEIVER
1 REMOTE CONTROL OUTLET
+ SEVERAL JUMPER WIRES
1 BREAD BOARD (OPTIONAL)
Receiver


Jumper Wires


Remote Control


Remote Controlled Outlet

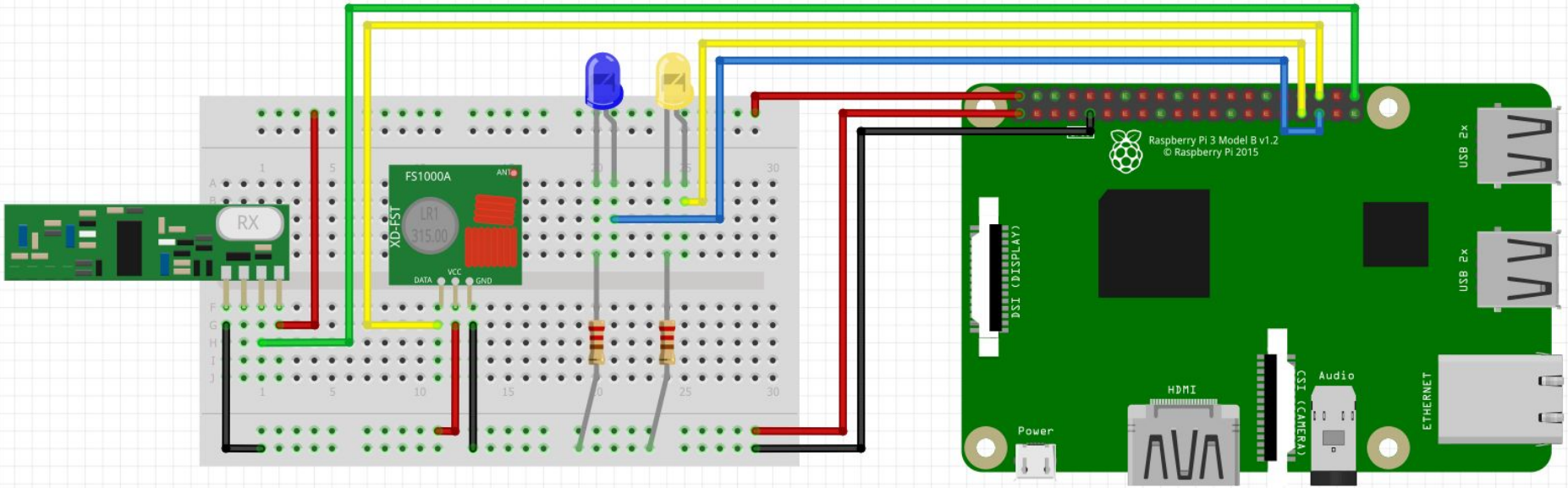


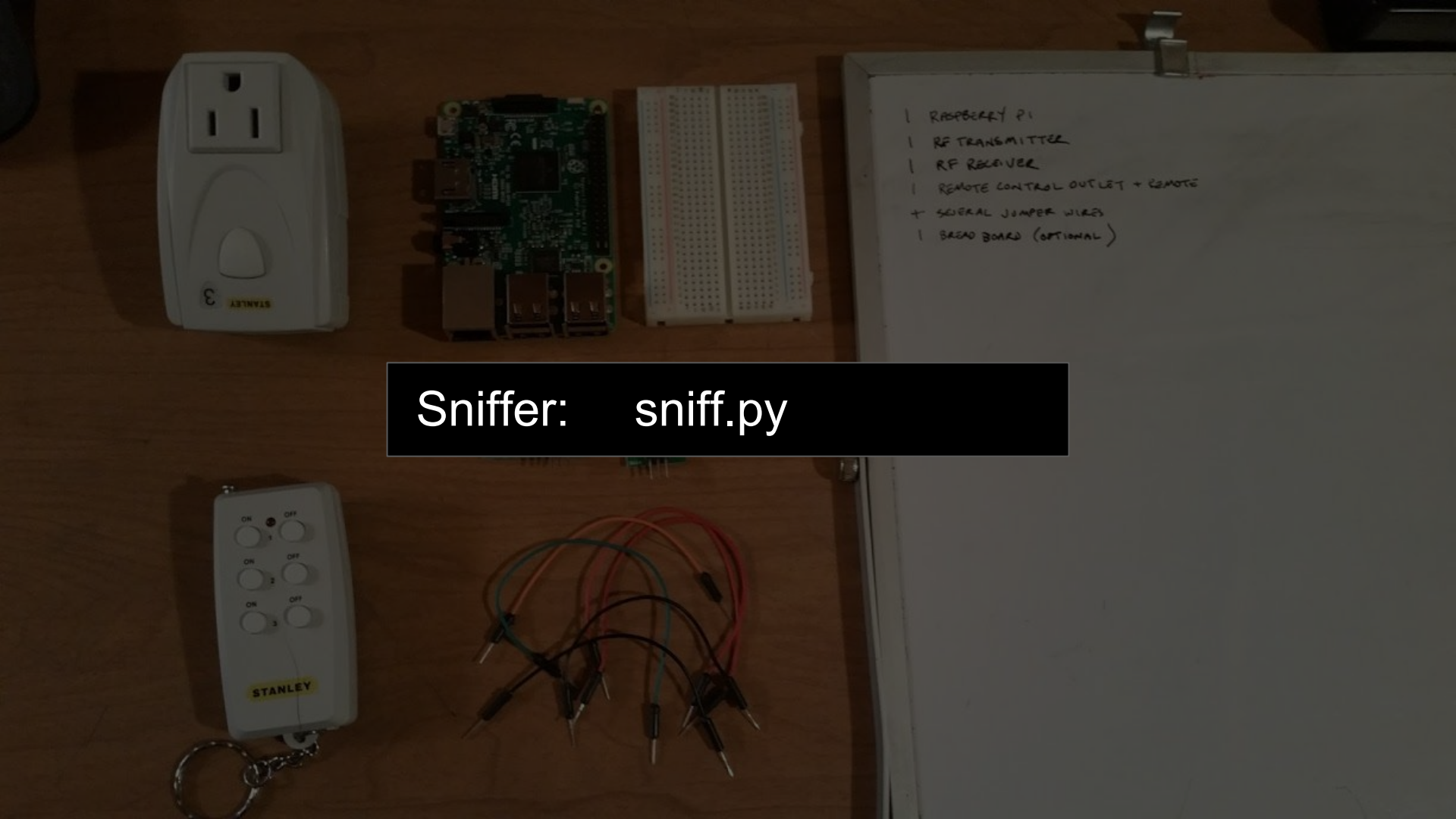

# Approx Cost of Project

| | |
|---|---|
| $35 | Raspberry Pi |
| $5+ | Micro SD |
| $1 | Pack of Jumper Wires |
| $3 | 5 Receivers/Transmitters |
| $2 | Breadboard |
| $10+ | Remote Controlled Outlet |
| Total | $56 |

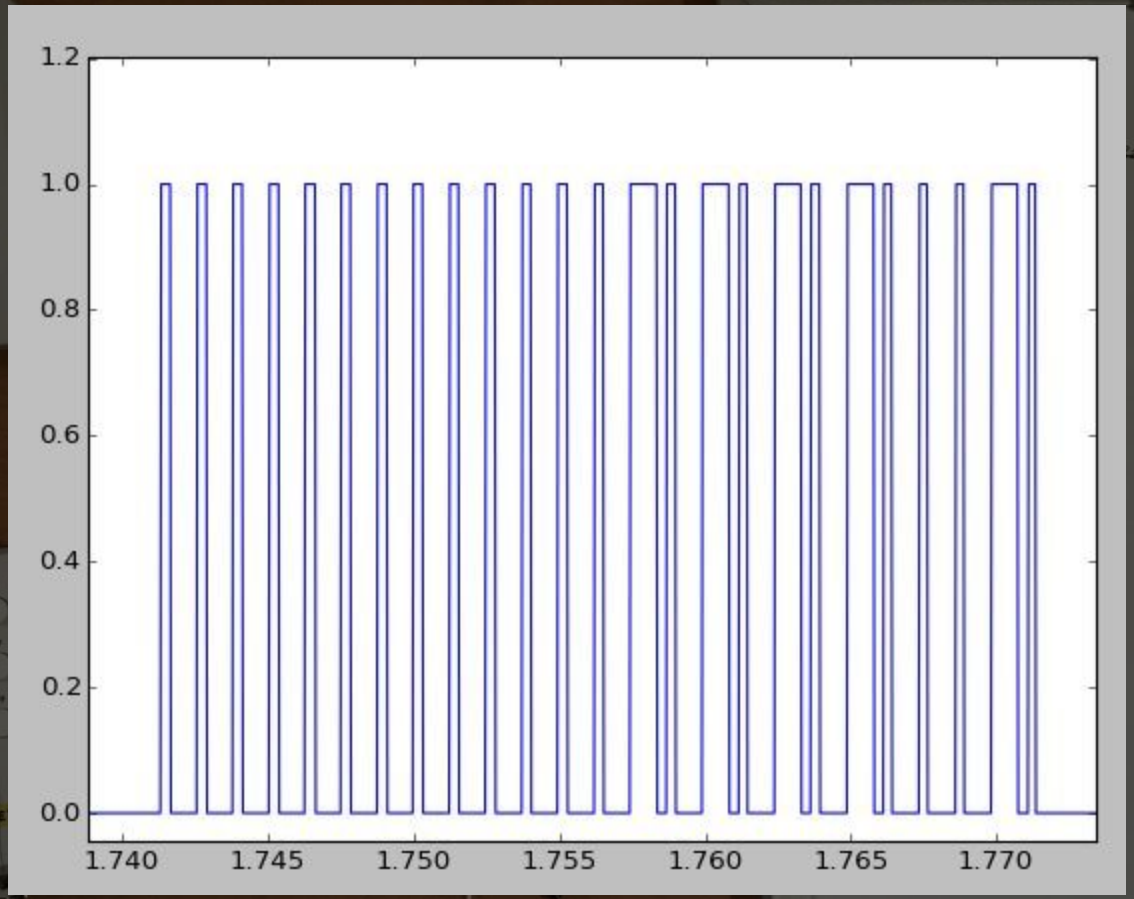
*(Prices from quick search on ebay.. Possibly cheaper if you want to wait a month to get them from China)*

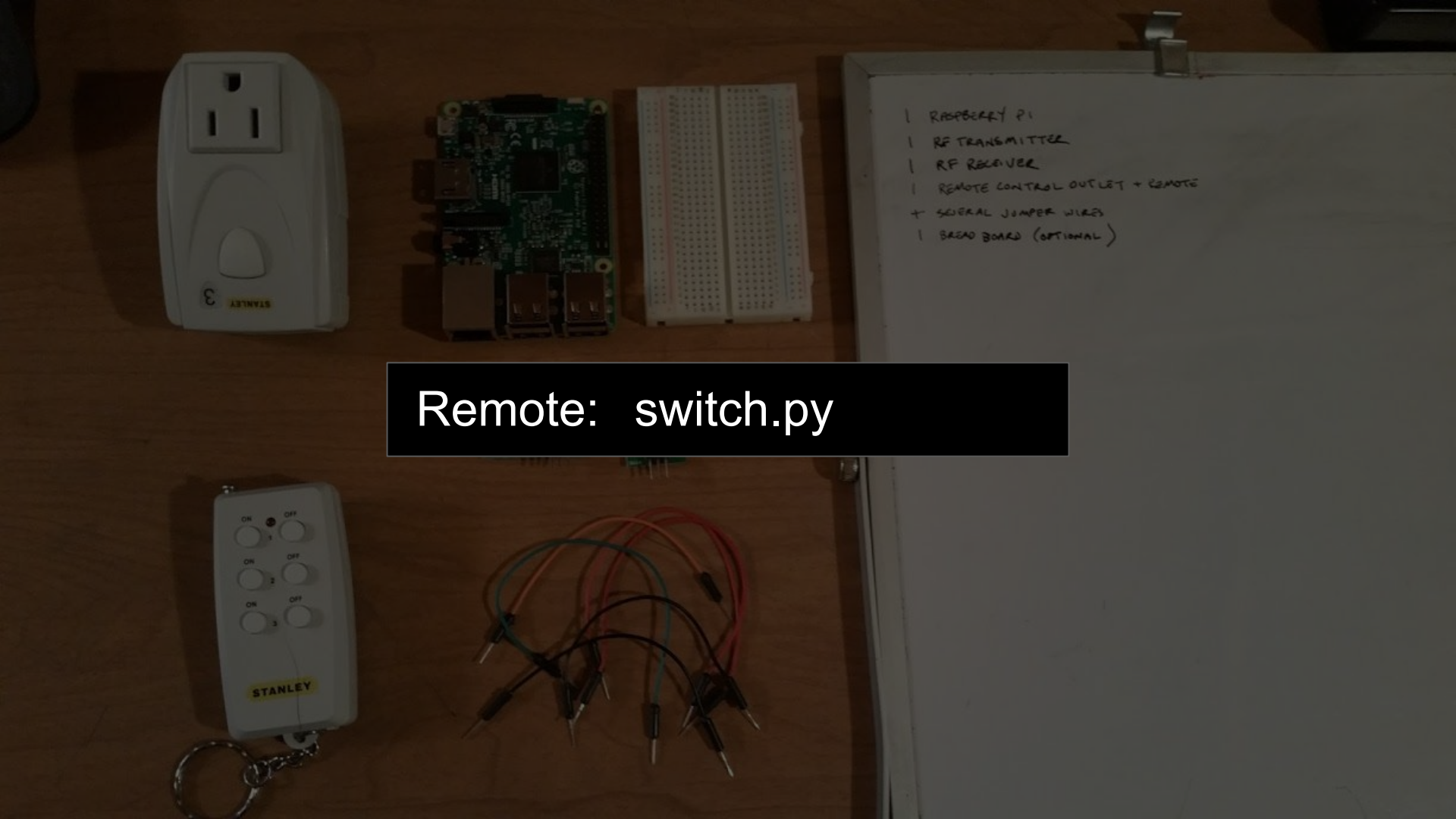
# Setup

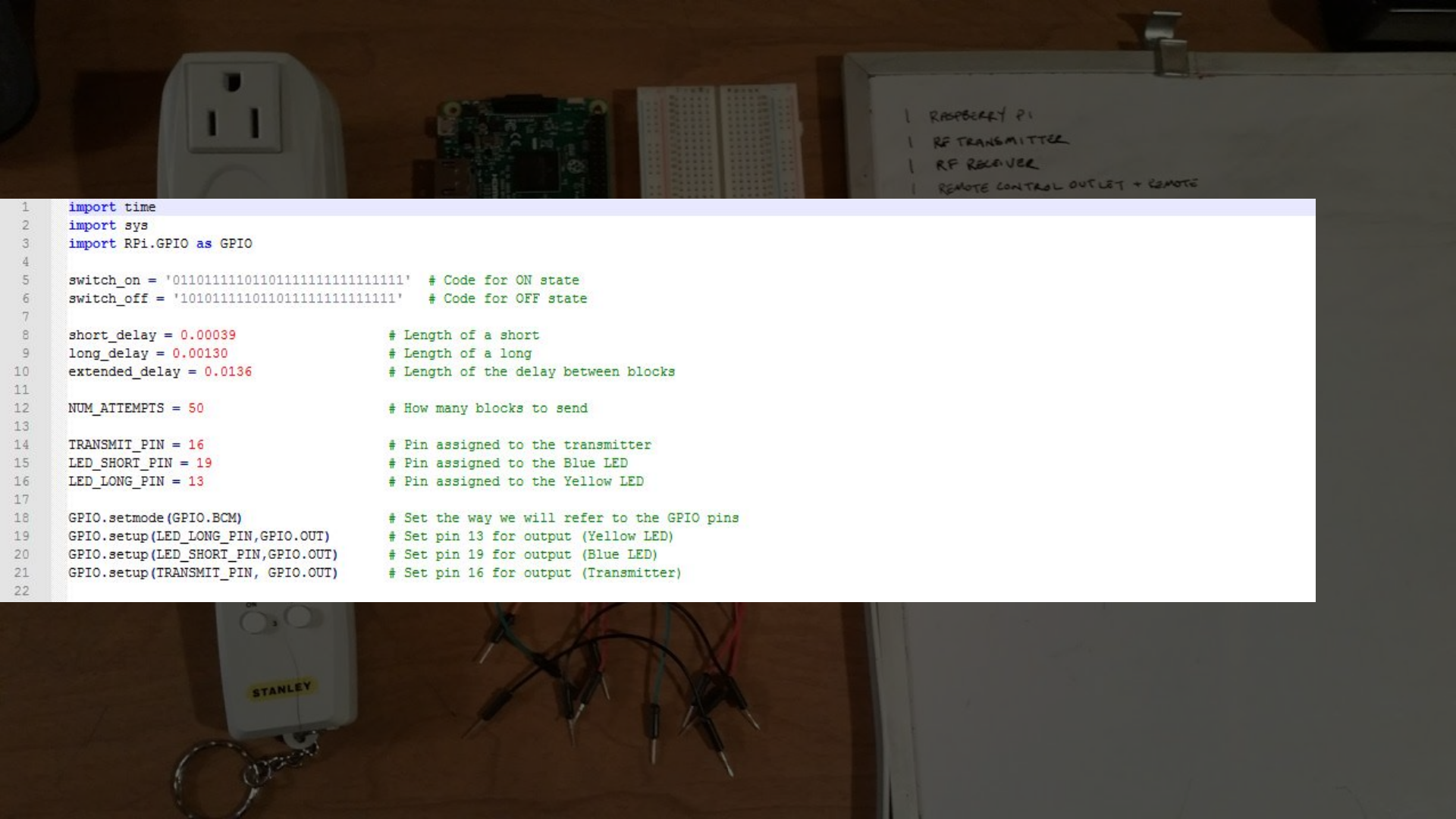

RX
FS1000A
ANT
XD-FST
LR1
315.00
DATA
VCC
GND
Raspberry Pi 3 Model B v1.2
© Raspberry Pi 2015
USB 2x
USB 2x
DSI (DISPLAY)
HDMI
Audio
CSI (CAMERA)
ETHERNET
Power


1 RASPBERRY PI
1 RF TRANSMITTER
1 RF RECEIVER
1 REMOTE CONTROL OUTLET + REMOTE
+ SEVERAL JUMPER WIRES
1 BREAD BOARD (OPTIONAL)
STANLEY
STANLEY
Sniffer: sniff.py

```
from datetime import datetime
import matplotlib.pyplot as pyplot
import RPi.GPIO as GPIO


RECEIVED_SIGNAL = [[], []]      # [[time of reading], [signal reading]]
MAX_DURATION = 2                # How many seconds to read the signal
RECEIVE_PIN = 21                # The data pin of the receiver

if __name__ == '__main__':

    GPIO.setmode(GPIO.BCM)                  # Set the way we will refer to the GPIO pins
    GPIO.setup(RECEIVE_PIN, GPIO.IN)        # Set pin 21 for input (Receiver)
    cumulative_time = 0                     # Start duration at 0
    beginning_time = datetime.now()         # Set time to now

    print '**Started recording**'

    while cumulative_time < MAX_DURATION:
        time_delta = datetime.now() - beginning_time
        RECEIVED_SIGNAL[0].append(time_delta)
        RECEIVED_SIGNAL[1].append(GPIO.input(RECEIVE_PIN))
        cumulative_time = time_delta.seconds

    print '**Ended recording**'
    print len(RECEIVED_SIGNAL[0]), 'samples recorded'   # Output how many samples were recorded
    GPIO.cleanup()

    print '**Processing results**'
    for i in range(len(RECEIVED_SIGNAL[0])):
        RECEIVED_SIGNAL[0][i] = RECEIVED_SIGNAL[0][i].seconds + RECEIVED_SIGNAL[0][i].microseconds/1000000.0

    print '**Plotting results**'
    pyplot.plot(RECEIVED_SIGNAL[0], RECEIVED_SIGNAL[1])
    pyplot.axis([0, MAX_DURATION, -1, 2])
    pyplot.show()
```


1.2
1.0
0.8
0.6
0.4
0.2
0.0
1.740
1.745
1.750
1.755
1.760
1.765
1.770


1 RASPBERRY PI
1 RF TRANSMITTER
1 RF RECEIVER
1 REMOTE CONTROL OUTLET + REMOTE
+ SEVERAL JUMPER WIRES
1 BREAD BOARD (OPTIONAL)
STANLEY
STANLEY
Remote: switch.py




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

switch_on = '01101111101101111111111111111'  # Code for ON state
switch_off = '10101111101101111111111111111'   # Code for OFF state

short_delay = 0.00039                  # Length of a short
long_delay = 0.00130                   # Length of a long
extended_delay = 0.0136                # Length of the delay between blocks

NUM_ATTEMPTS = 50                      # How many blocks to send

TRANSMIT_PIN = 16                      # Pin assigned to the transmitter
LED_SHORT_PIN = 19                     # Pin assigned to the Blue LED
LED_LONG_PIN = 13                      # Pin assigned to the Yellow LED

GPIO.setmode(GPIO.BCM)                 # Set the way we will refer to the GPIO pins
GPIO.setup(LED_LONG_PIN,GPIO.OUT)      # Set pin 13 for output (Yellow LED)
GPIO.setup(LED_SHORT_PIN,GPIO.OUT)     # Set pin 19 for output (Blue LED)
GPIO.setup(TRANSMIT_PIN, GPIO.OUT)     # Set pin 16 for output (Transmitter)

```

```
22
23  def transmit_code(code):
24
25      for t in range(NUM_ATTEMPTS):
26          for i in code:
27              if i == '1':
28
29                  # Short send | 1:
30                  GPIO.output(TRANSMIT_PIN, 1)              # Send high to the transmitter
31                  GPIO.output(LED_SHORT_PIN,GPIO.HIGH)      # Turn ON the Blue LED (a 1 has Started)
32                  time.sleep(short_delay)                   # Keep sending for the length of a short period
33
34                  # Pause before next send:
35                  GPIO.output(TRANSMIT_PIN, 0)              # Send low to the transmitter
36                  GPIO.output(LED_SHORT_PIN,GPIO.LOW)       # Turn OFF the Blue LED (a 1 has Finished)
37                  time.sleep(long_delay)                    # Send low for period of long delay before next send
38
39              elif i == '0':
40
41                  # Long send | 0:
42                  GPIO.output(TRANSMIT_PIN, 1)              # Send high to the transmitter
43                  GPIO.output(LED_LONG_PIN,GPIO.HIGH)       # Turn ON the Yellow LED (a 0 has Started)
44                  time.sleep(long_delay)                    # Keep sending for the length of a long period
45
46                  # Pause before next send:
47                  GPIO.output(LED_LONG_PIN,GPIO.LOW)        # Send low to the transmitter
48                  GPIO.output(TRANSMIT_PIN, 0)              # Turn OFF the Yellow LED (a 0 has Finished)
49                  time.sleep(short_delay)                   # Send low for period of short delay before next send
50
51              else:
52                  continue
53
54          GPIO.output(TRANSMIT_PIN, 0)                      # Send low for extended period
55          GPIO.output(LED_SHORT_PIN,GPIO.LOW)               # Ensure Blue LED is off
56          GPIO.output(LED_LONG_PIN,GPIO.LOW)                # Ensure Yellow LED is off
57          time.sleep(extended_delay)                        # Pause before sending next block
58
59      GPIO.cleanup()
60
61  if __name__ == '__main__':
62      for argument in sys.argv[1:]:
63          exec('transmit_code(' + str(argument) + ')')
64
```