Allen G. | Cris F.O. | Abdalla E.

ENGR 114 | Fall 2018

December 11th, 2018

Final Report

## **Stay Frosty**

#### **Problem Statement:**

The objective of this experiment is to use a PIR motion sensor to control a fan as a on/off switch. The PIR sensor will be used to detect if there is someone in front of the fan where then the raw sensor data will be transferred to the MATLAB® program. If the sensor data passes a certain threshold the fan will be activated and then controlled by the user if they decide to keep than fan on or switch it off using MATLAB®.

#### **Hardware Setup:**

Group Member Names: Allen George, Cristopher Falcon Ortiz, Abdalla Elattar

Course and Quarter: ENGR114 Fall 2018

Date: 12/9/2018

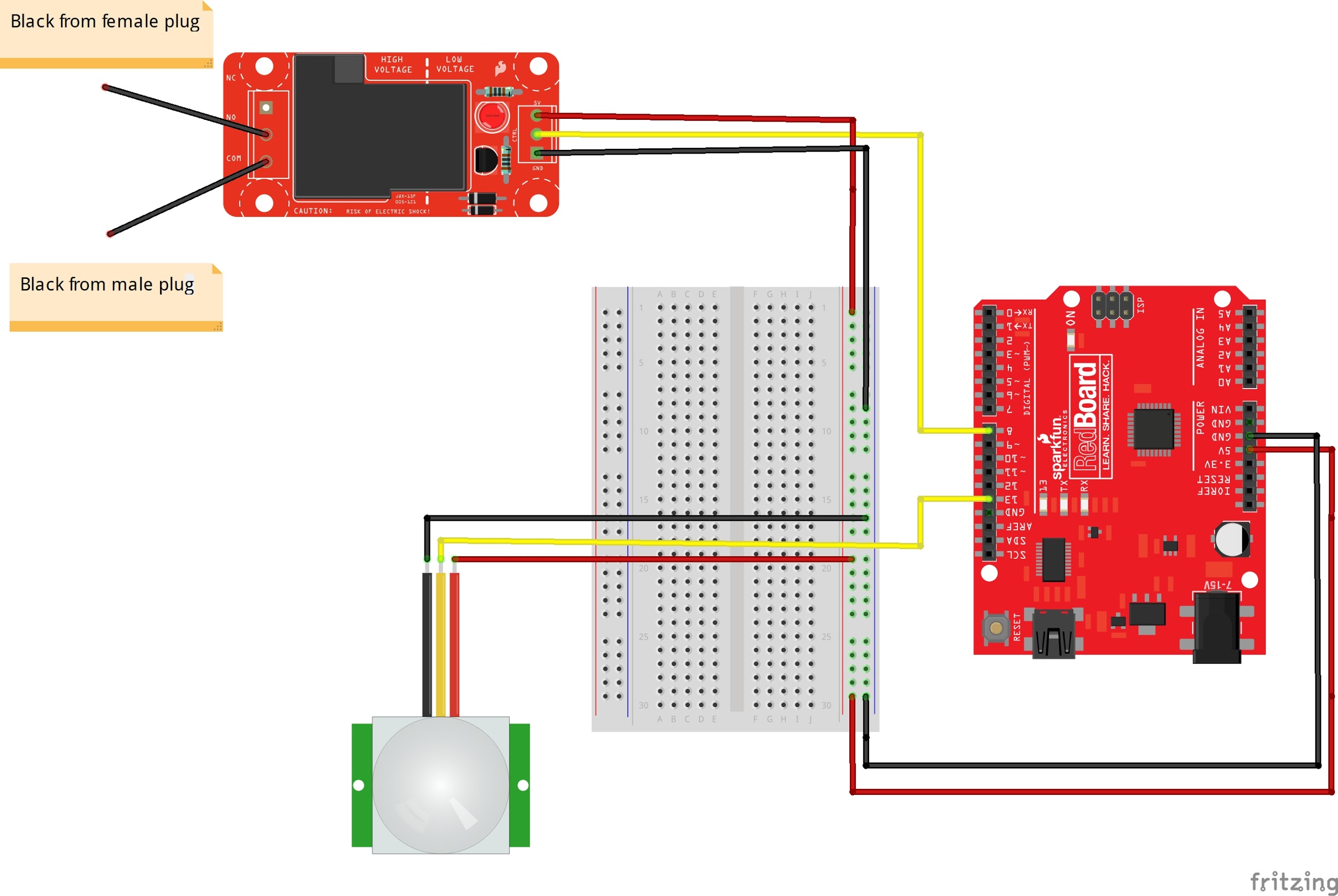
Revision Number: 1

**Stay Frosty Hardware Setup**

**Bill of Materials:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part Name | Purpose | Item Name | URL | Price |
| Arduino Uno | Runs our program | Sparkfun Redboard | https://www.sparkfun.com/products/13975 | $19.95 |
| Breadboard | Building circuit | Breadboard - Self-Adhesive | https://www.sparkfun.com/products/12002 | $4.95 |
| Enclosure | Enclosure for Relay | Sparkfun Big Red Box | https://www.sparkfun.com/products/11366 | $8.95 |
| PIR Sensor | Input sensor for program | PIR (motion) sensor | https://www.adafruit.com/product/189 | $9.95 |
| Relay | Lets us power high voltage devices with low voltage Arduino | SparkFun Beefcake Relay Control Kit | https://www.sparkfun.com/products/13815 | $8.95 |
| Fan | Output of program | Received from Peter Kazarinoff | Received from Peter Kazarinoff | Received from Peter Kazarinoff |
| Jumper Cables | Connects components | Jumper Wires Standard 7" M/M - 30 AWG | https://www.sparkfun.com/products/11026 | $1.95 |
| Extension Cable | Connects to high voltage end of relay to plug into wall socket and fan | Received from PCC Maker Lab | Received from PCC Maker Lab | Received from PCC Maker Lab |
| Nylon Dome Strain Relief Connectors | Helps prevent component damage | Received from PCC Maker Lab | Received from PCC Maker Lab | Received from PCC Maker Lab |
| Mini-USB cable | Uploads program to Arduino | SparkFun USB Mini-B Cable - 6 Foot | https://www.sparkfun.com/products/11301 | $3.95 |
| Standoffs/Screws | Lifts relay off floor of enclosure | Received from PCC Maker Lab | Received from PCC Maker Lab | Received from PCC Maker Lab |
| Wire Nuts | Connects wires together and prevents them from contacting other elements/wires | Received from PCC Maker Lab | Received from PCC Maker Lab | Received from PCC Maker Lab |

**Hardware Schematic**:

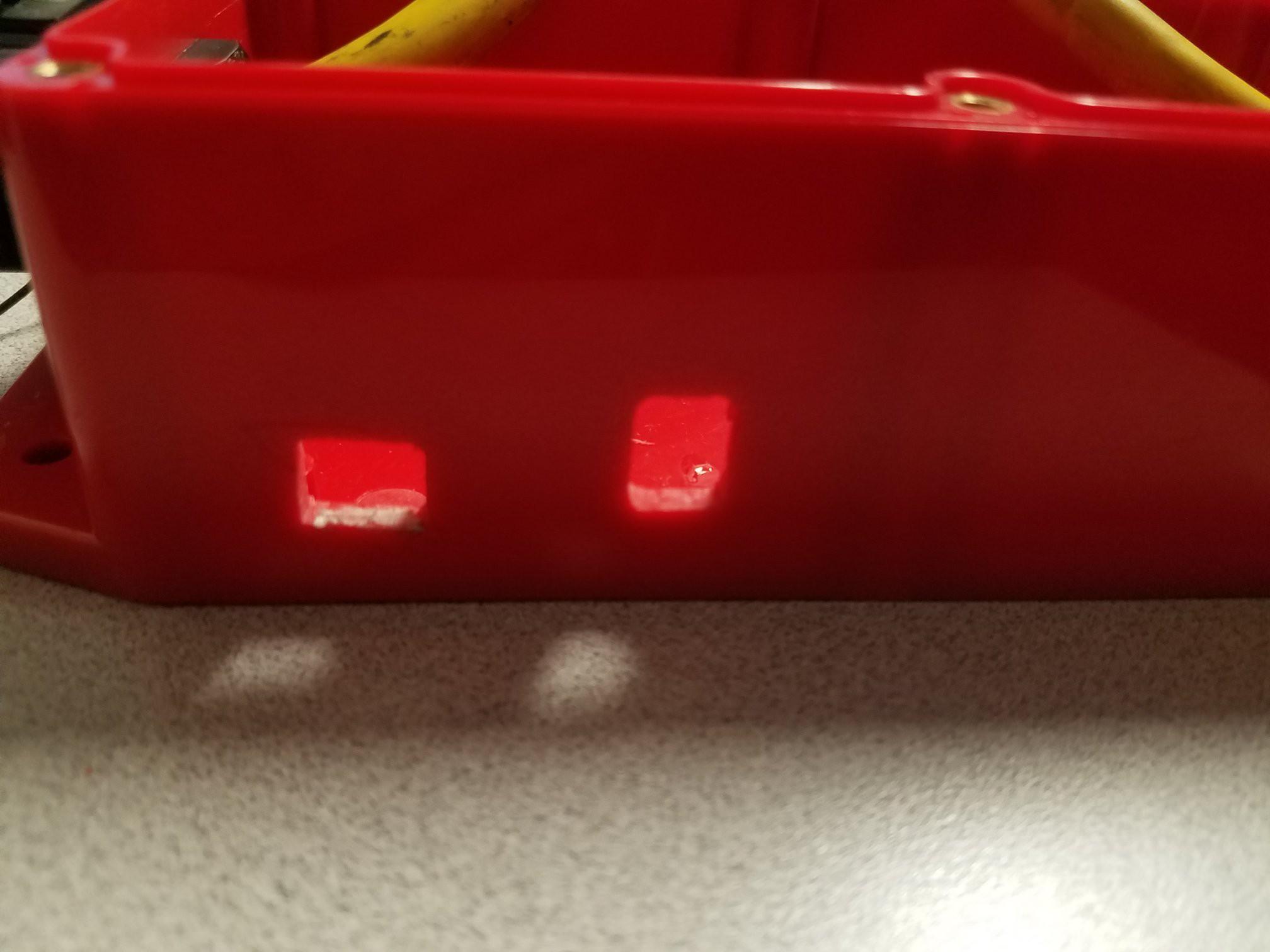


**Hookup Guide:**

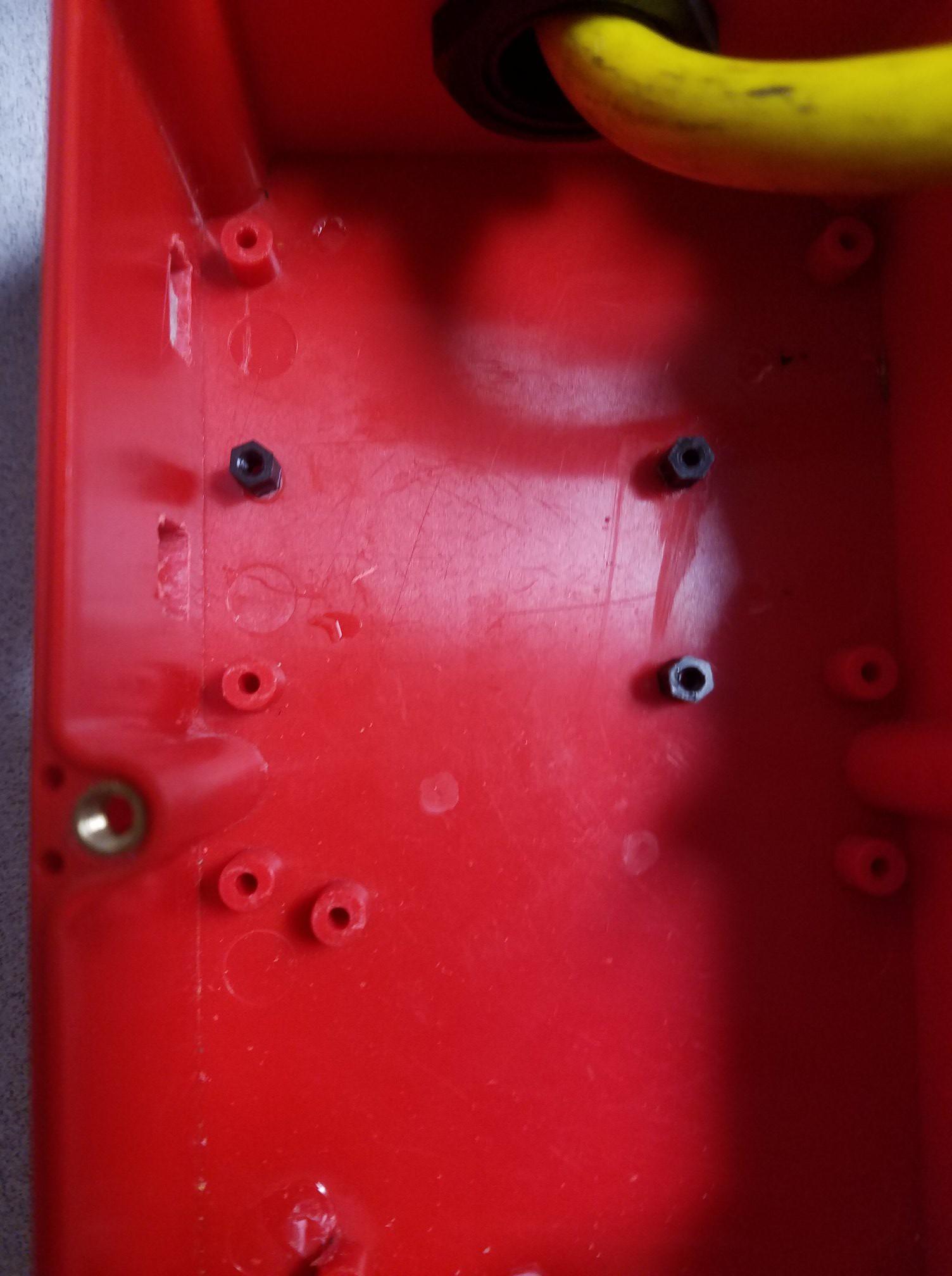
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Part | Pin | Connector | Pin | Part |
| PIR Sensor | 13 | Jumper Wire | 13 | Arduino Uno |
| Relay | 8 | Jumper Wire | 8 | Arduino Uno |

**Step 1:** Build the Beefcake relay. Follow instructions at https://learn.sparkfun.com/tutorials/beefcake-relay-control-hookup-guide?\_ga=2.126438346.678827907.1495830299-657127905.1456517273

**Step 2:** Drill 3 holes into relay enclosure. 2 Large circular holes on each side to insert strain relief connectors, and 1 small hole so that the jumper cables have a point of exit.

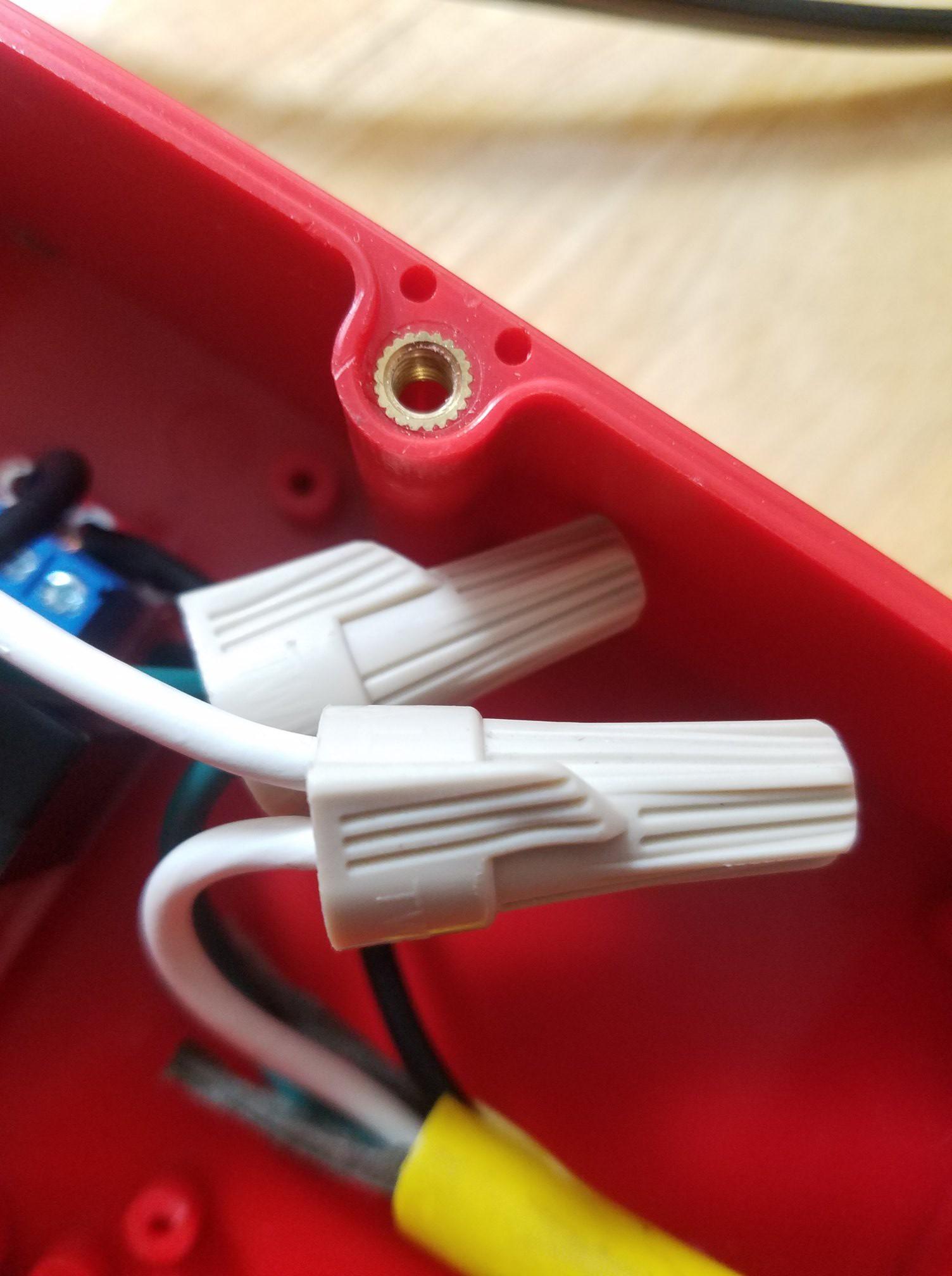
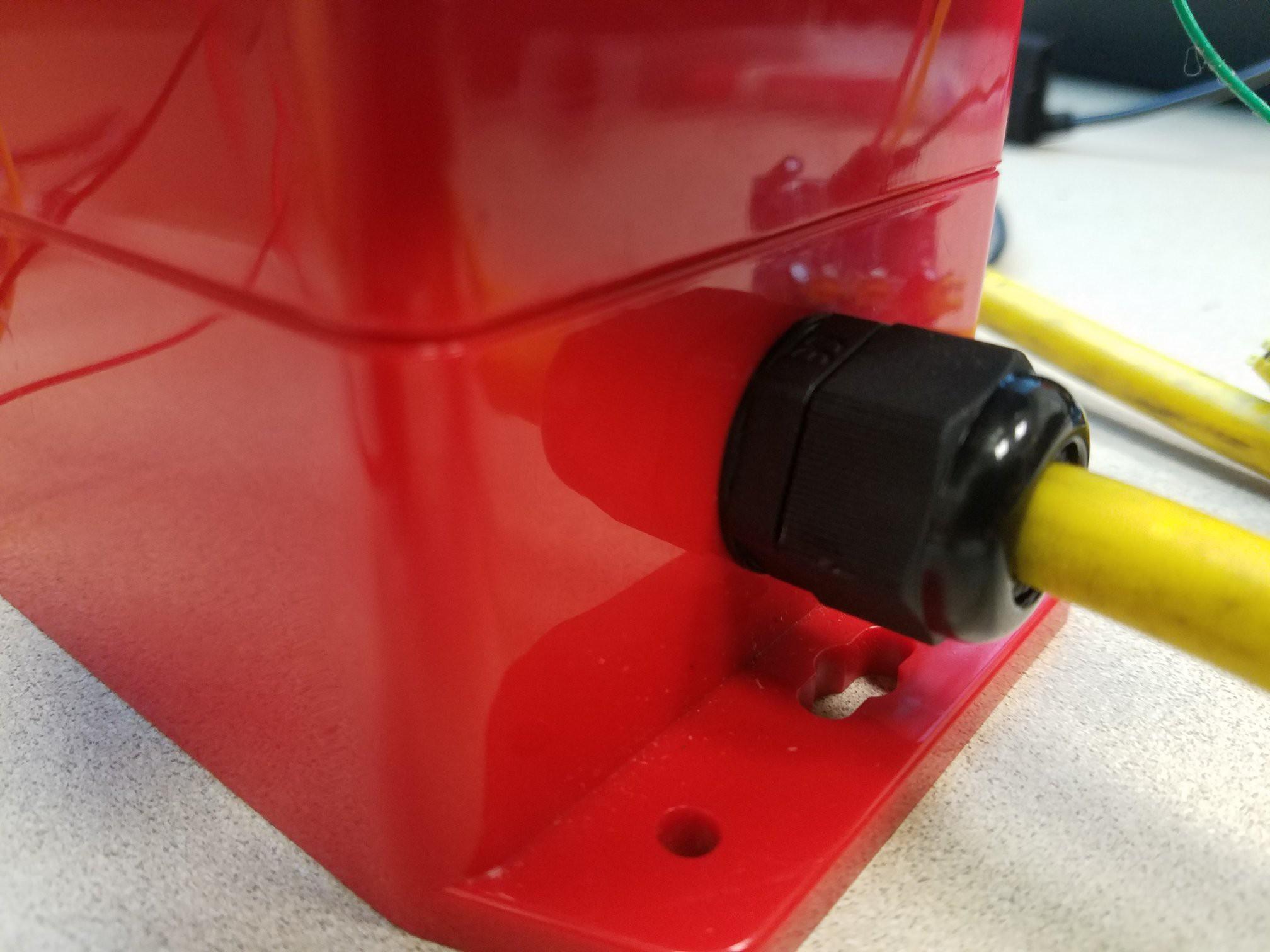


**Step 3**: Add standoffs onto the bottom of the enclosure, so that the relay will be raised off the ground. Simply add a dab of super glue to the bottom and place them where they need to be, so that the 4 standoffs line up with the 4 holes on the relay.



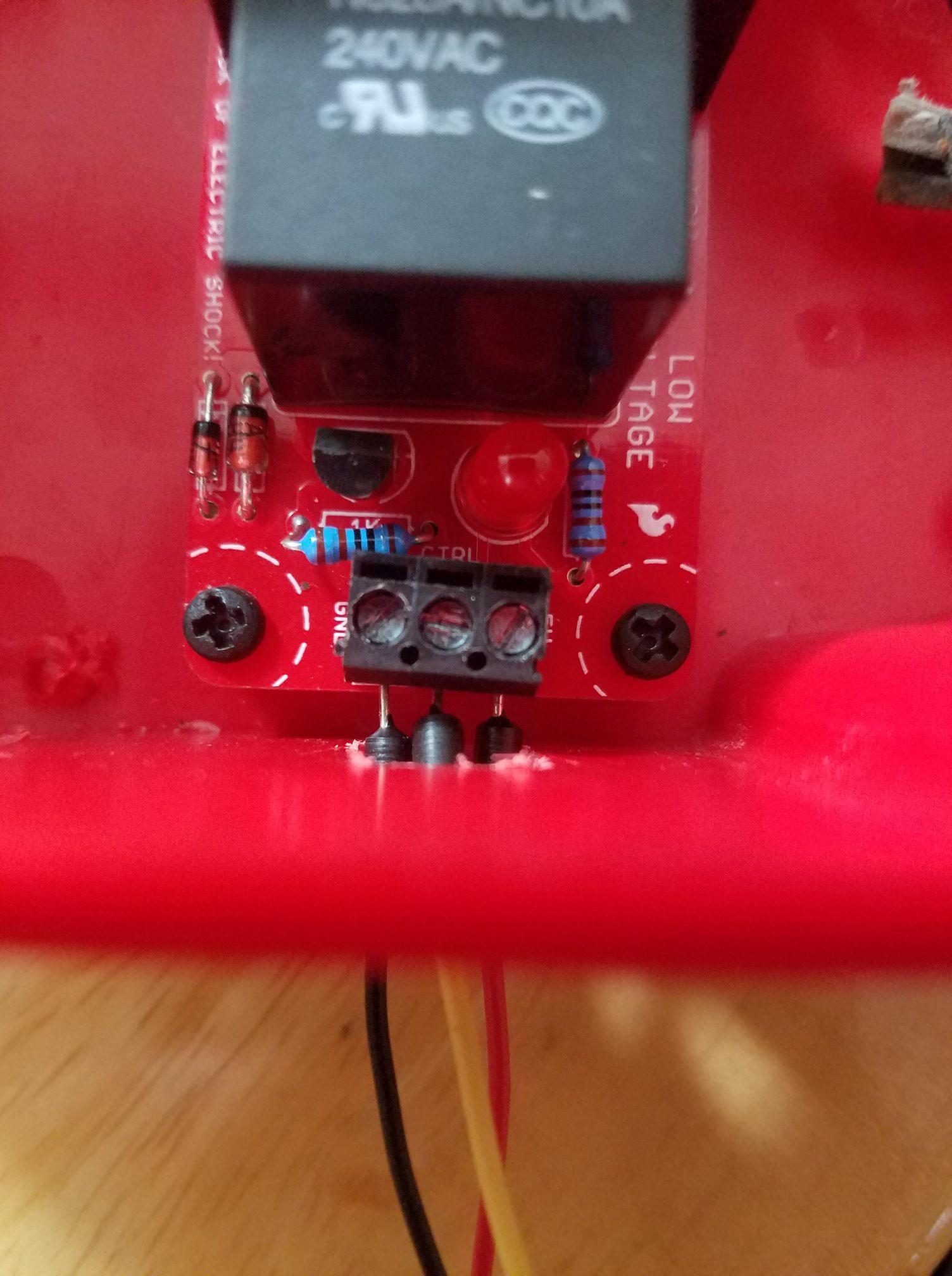
**Step 4:** Cut the extension cable so that the male end is much longer than the female end. The f emale end should be around a foot long and the male end should be the remaining amount. After that use a box cutter or another type of knife to remove about 4 inches of the rubber cover, exposing 3 wires on each end, black, white, and green. Finally, strip off about half an inch of the protective covering off each wire using wire strippers.

**Step 5:** Slip the strain relief connectors on each end of the extension cables, and connect them to the enclosure. Secure the strain relief connectors onto the enclosure. After that, using wire nuts, connect the white (neutral) wires together, and the green (ground) wires together.



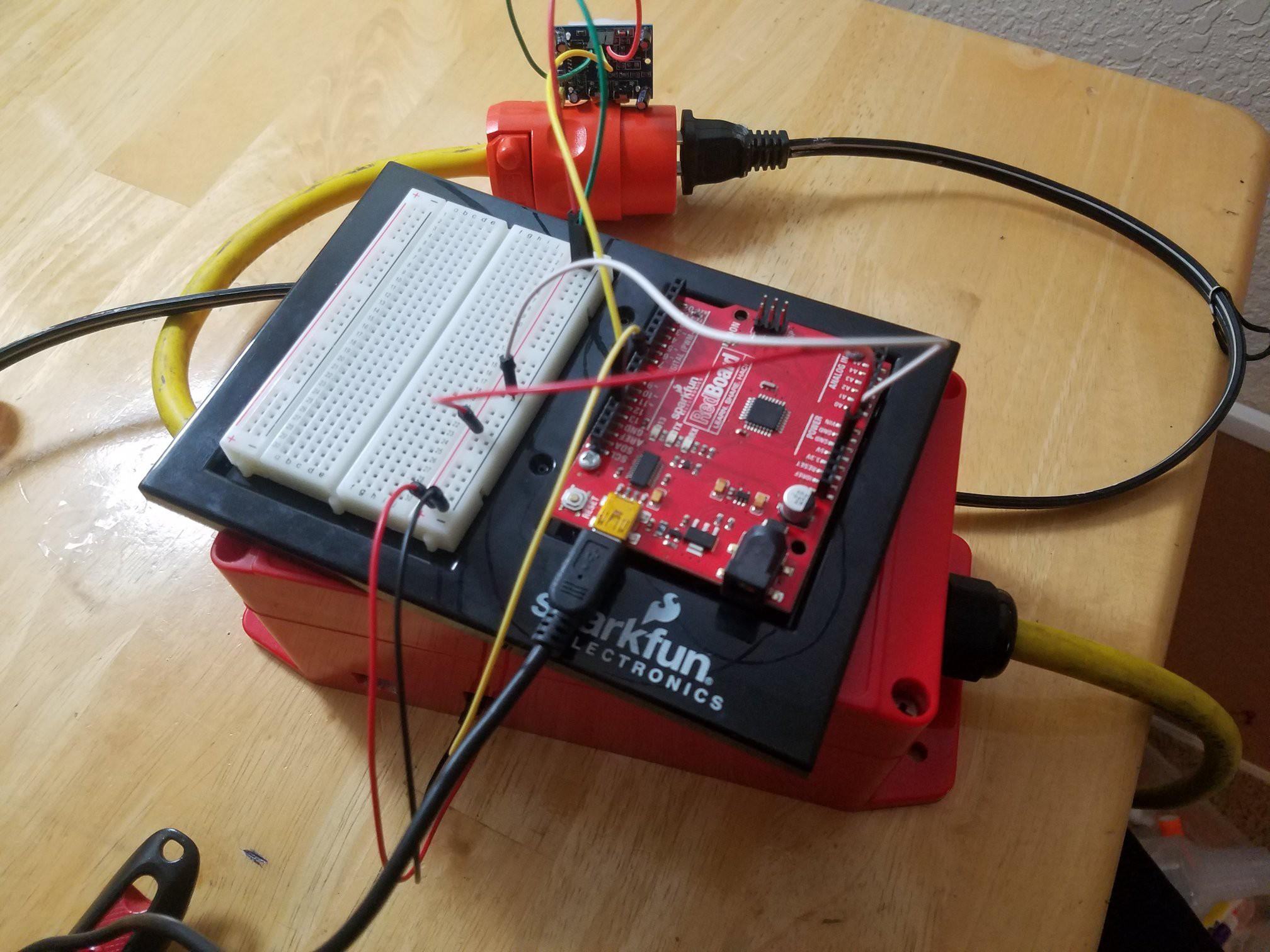
**Step 6:** Connect the black (hot) wires into the relay. The black from the male plug should go in the common (COM) terminal, and the black from the female plug should go in the normally open (NO) terminal. Once each wire is in, secure the connection by tightening the screws on the terminal. 

**Step 7:** Using the standoff screws, fasten the relay onto the enclosure into the standoffs we installed earlier. The low voltage end should be facing the hole that was drilled earlier, so that the terminals are fully accessible through the hole. Once all the wires are in and the relay is secured, tighten the strain relief connectors while allowing a little bit of slack.

**Step 8:** Connect jumper wires to the low voltage end. Black into ground (GND), yellow into control (CTRL), and red into power (5V). Secure the connections by tightening the screws. Now that the relay is complete, place the top of the enclosure back on and secure it with the screws it comes with.

**Step 9:** Using a breadboard and the Arduino Uno, build the circuit specified by the Fritzing diagram, with the relay connected to digital pin 8 and the PIR sensor connected to digital pin 13.

**Step 10:** Plug fan into extension cable. After that, plug extension cable into a wall socket, and connect the mini-USB cable to your computer/laptop. The setup is now complete and you can now use the program.



#### **Code:**

**Matlab Code:**

Final Project MATLAB Code Allen George, Cristopher Falcon Ortiz, Abdalla Elattar

* Authors: Allen George, Cristopher Falcon Ortiz, Abdalla Elattar
* Date: 12/11/18
* Course: ENGR114 Fall 2018
* Description: Accepts Arduino input and responds by sending outputs to Arduino, and allows user input to control a fan.

**Contents**

* Open Serial Port
* Analyzing Arduino input
* User input to turn on & off fan
* Close the serial port

**Open Serial Port**

clc, clear; % clears command window and workspace variables

Port = 'COM7'; % Port the Arduino is connected to

delete(instrfindall); % deletes any connected ports

a = serial(Port); % create the serial port

fopen(a); % open the serial port

pause(1); % pause for 1 second to make sure a connection is made

out = instrfind('Port' , Port); % see if the port you specified is open

fprintf('Serial Port is Open\n') % lets user know port is open

**Analyzing Arduino input**

fprintf('\nMove in front of sensor to activate fan\n')

serial\_read = zeros(1,5); % Initializes serial\_read as a 1x5 zeroes matrix so that our % if loop can function

while (1) % Sets up an infinite loop

serial\_read(end+1) = fscanf(a,'%d'); % read the serial input as a decimal number

pause(.50); % wait half a second before next serial reading

if serial\_read(end-4:end) == ones(1,5) % If the last 5 digits of serial\_read are one, activate loop.

send\_str = 'H'; % Sends H to arduino, which turns fan on.

fprintf(a, '%s', send\_str);

pause(1);

break % Ends infinite while loop

end

end

**User input to turn on & off fan**

while (1) % Sets up infinite loop

% asks user for input continuously. Invalid inputs are ignored.

user\_input = input('\nType H to turn on the LED, L to turn off the LED, or Q to quit: ','s');

if strcmp(user\_input ,'L') % If user types L, turns fan off

fprintf('[Fan Turns Off]\n')

send\_str = 'L';

fprintf(a, '%s', send\_str); % Sends L to arduino

pause(1); % 1 second pause

elseif strcmp(user\_input, 'H') % If user types H, turns fan off

fprintf('[Fan Turns On]\n')

send\_str = 'H';

fprintf(a, '%s', send\_str); % Sends H to arduino

pause(1); % 1 second pause

elseif strcmp(user\_input, 'Q') % If user types Q, turns fan and program off.

fprintf('[Program Terminates]\n')

send\_str = 'L';

fprintf(a, '%s', send\_str); % Sends L to arduino

pause(1); % 1 second pause

break % Ends loop

end

end

**Close the serial port**

fclose(a);

delete(a);

clear a;

fprintf('\nSerial Port is closed\n')

**Arduino Code:**

// Allen George, Cris, Abdulla Final Project Arduino Code

const int relayPin = 8; // the pin that the LED is attached to

int incomingByte; // a variable to read incoming serial data into

int sensorPin = 13;

int sensorValue; // An integer variable to store the potentiometer reading

void setup() {

Serial.begin(9600); // initialize serial communication:

pinMode(relayPin, OUTPUT); // initialize the relay pin as an output

pinMode(sensorPin, INPUT); // initialize the PIR sensor pin as an input

delay(3000); // 3 seconds delay to calibrate PIR Sensor

}

void loop() {

sensorValue = digitalRead(sensorPin); // Reads the value of the sensor. 1 for HIGH, 0 for LOW

Serial.println(sensorValue); // Output reading to the serial line

delay(500); // .5 second delay to not overwhelm code

if (Serial.available() > 0) { // see if there's incoming serial data:

incomingByte = Serial.read(); // read the oldest byte in the serial buffer:

if (incomingByte == 'H') { // if it's a capital H (ASCII 72), turn on the relay:

digitalWrite(relayPin, HIGH);

}

if (incomingByte == 'L') { // if it's an L (ASCII 76) turn off the relay:

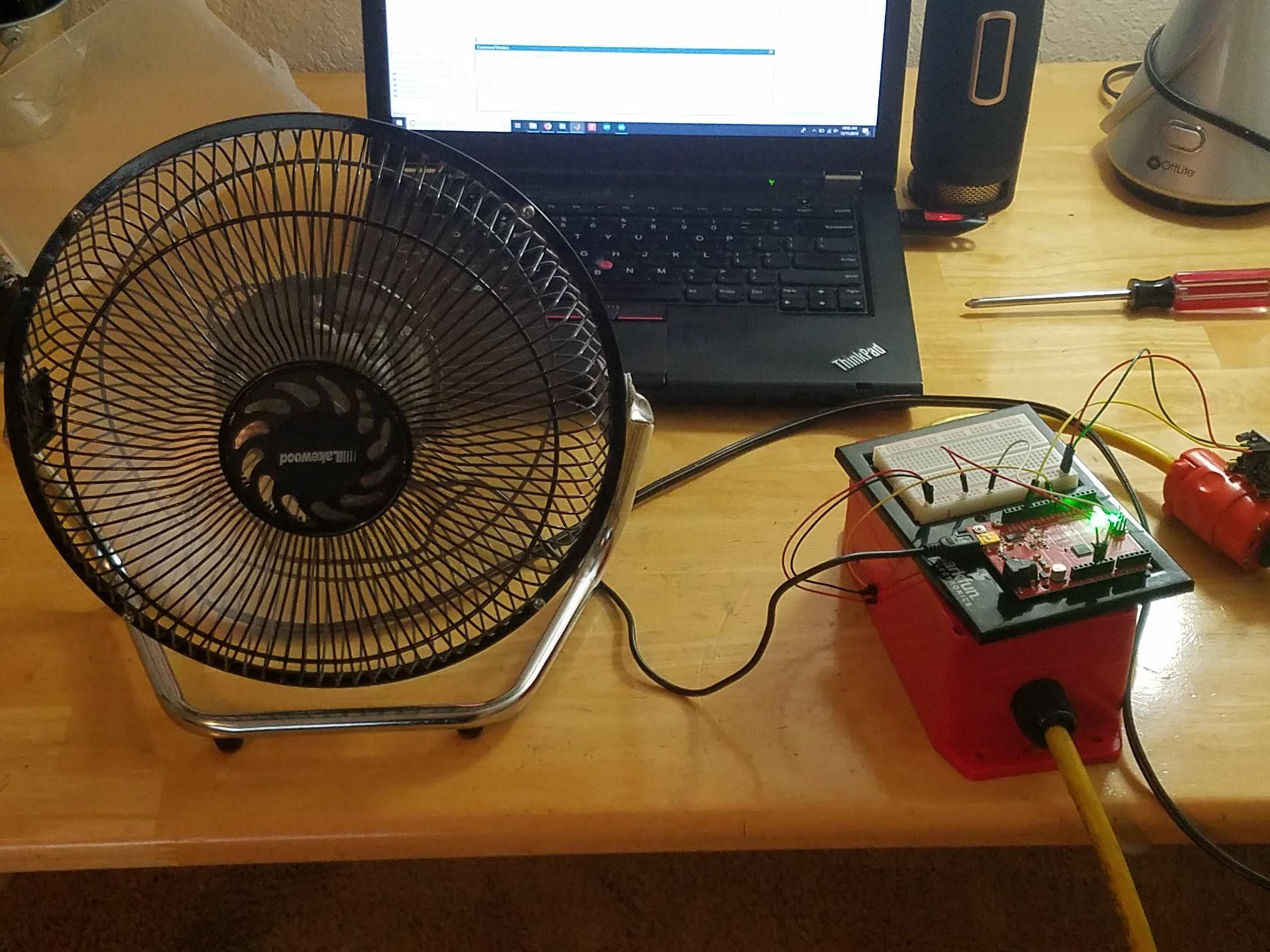
digitalWrite(relayPin, LOW);

}

}

}

**Results:**

****

#### **Future Work:**

Another group of students could take this build and use it for other commonly used house appliances, such as a toaster, television, shower, etc.. What this would do is that all the projects could be brought together and be implemented into a smart home. Where a smart home would detect motion and turn on specific appliances throughout the day. An example of this would be if there is motion detected in the bedroom early in the morning then the toaster will turn on to toast some bread, the shower may turn on so warm steamy water is ready for the user to get into and so on. The motion detection will help greatly with conserving energy for future smart homes, where appliances will turn off after certain time if they’re no longer needed.

#### **License:**

The FreeBSD Copyright

Copyright 1992-2018 The FreeBSD Project.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

The views and conclusions contained in the software and documentation are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the FreeBSD Project.