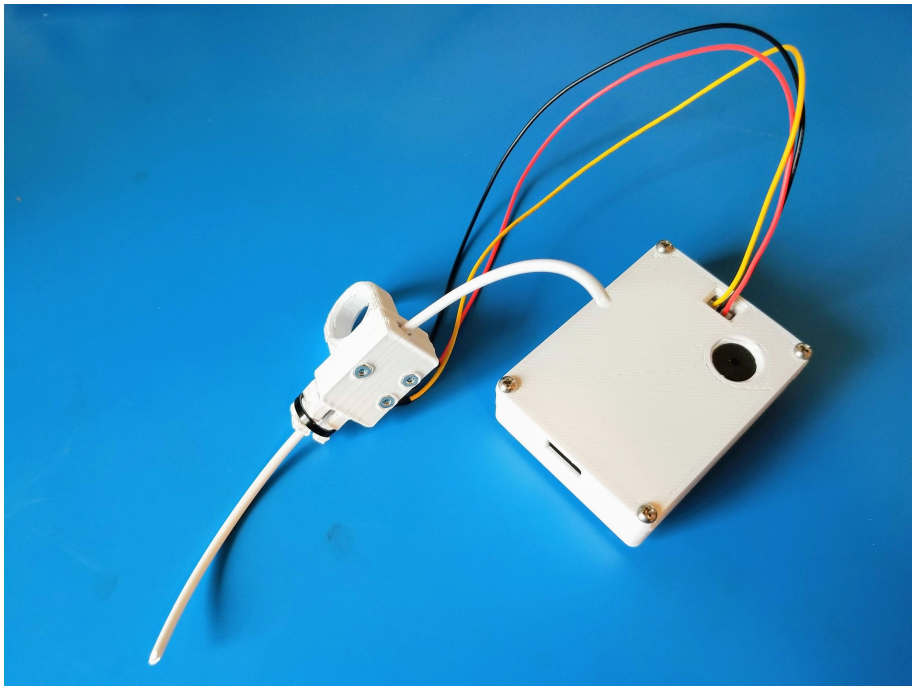
Wemos D1 Mini Filament Sensor User Manual



Swallowtail Electronics

Swallowtail Electronics May 2020

Contents

List of Figures

List of Tables

Chapter 1

Getting Set-up & Using the Sensor

This chapter describes how to get your Wemos D1 Mini Filament Sensor set-up and connected to the WiFi.

1.1 Included Items

The product package should include the following items:

- Wemos Control Box
- Filament Switch
- Zip Ties
- USB-B Micro Cable
- USB Power Plug

1.1.1 Wemos Control Box

The Wemos Control Box contains the main PCB, a port for power input via USB, a port for the filament switch connector, and an opening for the piezo buzzer.

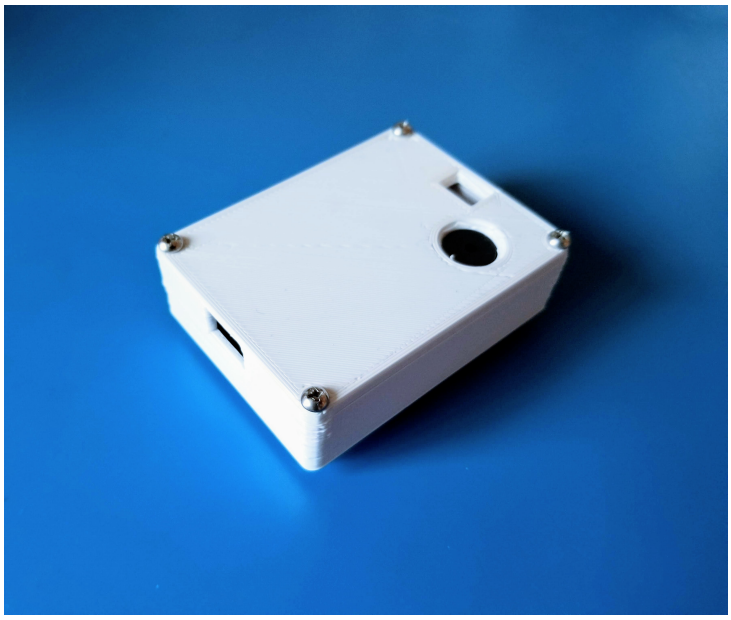


Figure 1.1: Wemos Control Box

1.1.2 Filament Switch

The filament switch contains the microswitch and has the hole for the filament. The loop on the side of the enclosure is compatible with the Lulzbot TAZ 6 Style. There are recommendations for other filament switches at `http://tristanluther.com/projects/email`.

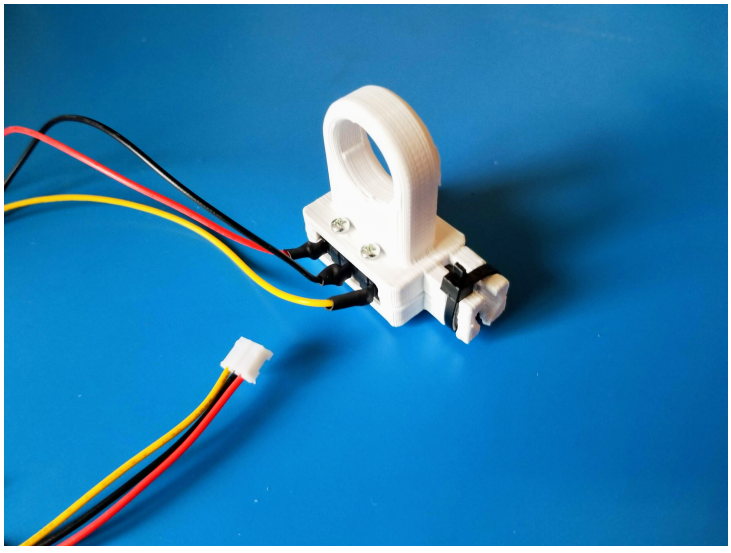


Figure 1.2: Filament Switch

1.1.3 Zip Ties

The zip tie included is meant to fasten the PTFE tube to the filament switch on the output feed of the switch.

1.1.4 USB-B Micro Cable

The USB cable supplies power to the system. The cord is plugged in though the front of the Wemos Control Box.

1.1.5 USB Power Plug

Used to connect the USB cable to the wall socket.

1.2 Set-up Method

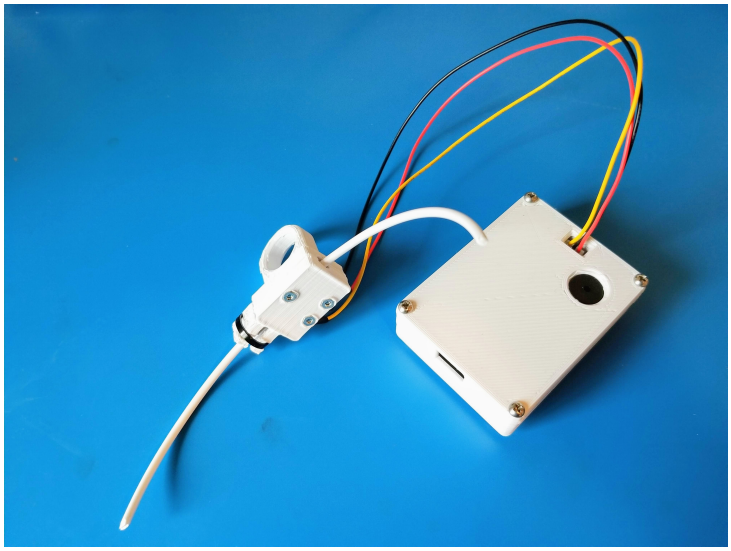


Figure 1.3: Entire Filament Sensor Set-up

1. Begin by securing the filament switch to the 3D printer (if using the Lulzbot Compatible Version plug the ring into the splint). Place the printer filament though the switch, ensure the filament properly depresses the switch. Plug the filament switch into the Wemos Control Box.

2. Power on the system by plugging it into the wall with the USB cable and plug.

3. Upon powering on the Wemos Filament Sensor will become a WiFi Access Point under the name: **WemosFilamentSensor**. Connect to it on your computer. The password to the access point is: **wem0sfun!**.

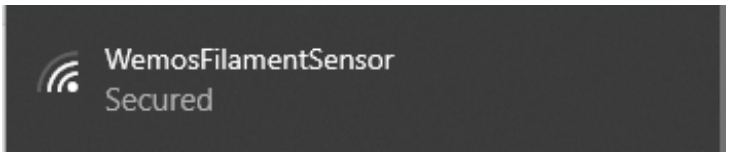


Figure 1.4: Access Point Name

4. Upon being connected, open an internet browser and input`wemos.local` into the address bar or `192.168.4.1` if that does not work. You will be directed to the Wemos Filament Sensor Initialization page.

5. Fill out the form then submit.

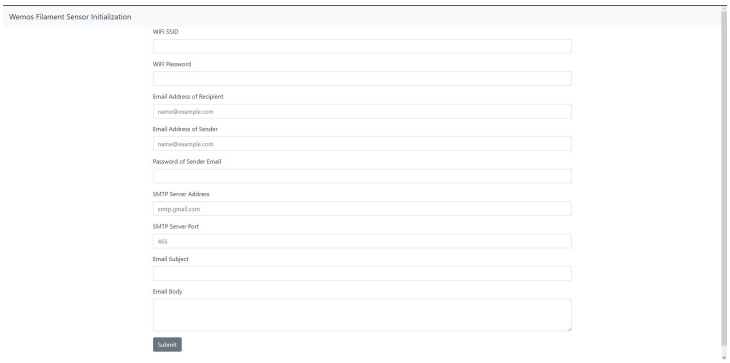


Figure 1.5: Empty Initialization Form

6. Upon submitting the form the Wemos Filament Sensor will disconnect you from the Access Point and close it. It will then function as a client to the WiFi credentials that you submitted. Once the sensor connects to the WiFi it will be activated and will send the inputted email once the filament switch is no longer depressed by the filament.

Chapter 2

Troubleshooting

2.1 Accessing the System Debugger

The Wemos Filament Sensor features an on-board debugger so you can troubleshoot a system that is not functioning as intended. *This tutorial will be for Windows 10, however the same concepts for reading the serial data apply.

Required Items:

- Computer with USB Ports
- Download PuTTY `https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html` or any similar program to access the Serial Ports on your computer
- Wemos Filament Sensor

1. Plug the Wemos Filament Sensor into your computer via one of the USB Ports.
2. Open PuTTY and select the Serial radio button in the 'Specify the destination you want to connect to' input group. For the 'Serial Line' input box input the COM Port that the Wemos Filament Sensor is attached to and set the Speed/Baud Rate to: **115200**. For how to find which COM Port the Wemos Filament Sensor is attached two continue to step 3, otherwise skip to step 4.

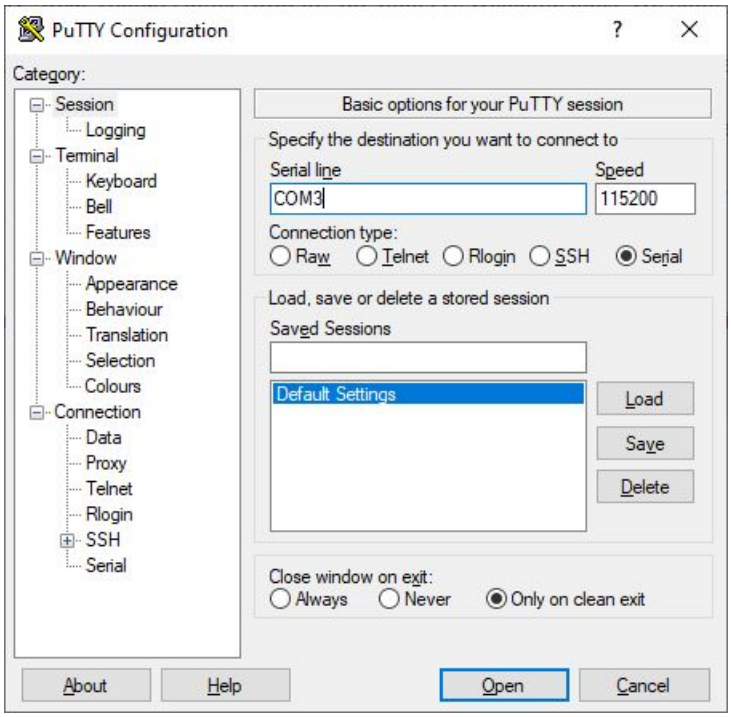


Figure 2.1: Using PuTTY with Wemos Filament Sensor

3. The COM Port can be found by going to the Device Manager and looking for the 'Ports (COM & LPT)' section. Look at the devices attached and find the COM that the device is attached to. The device should have the name **USB-SERIAL CH340**. In my case the serial port was COM3. *If the USB-SERIAL CH340 is not appearing you may need to install the driver for the CH340 here: `https://learn.sparkfun.com/tutorials/how-to-install-ch340-drivers/all#windows-710`.

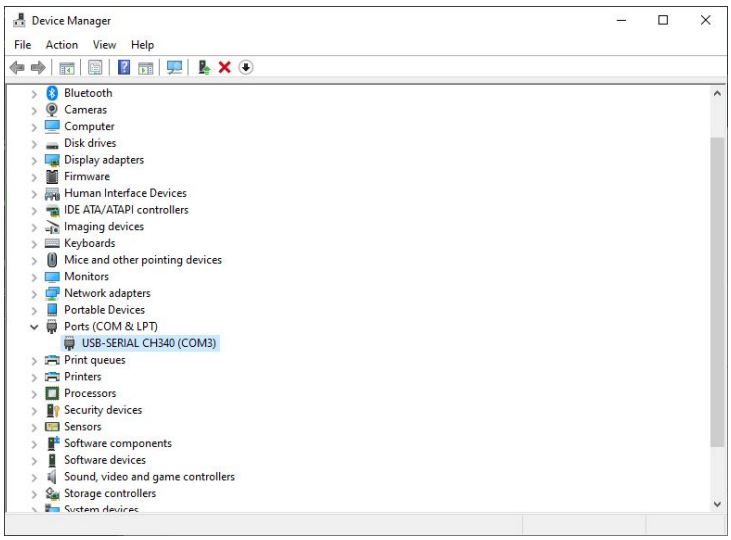


Figure 2.2: Device Manager View

4. Open the session in PuTTY and you should be greeted with a black serial screen. Go though the standard Wemos Filament Sensor set-up and you will see the info you sent via the web form outputted in the terminal, WiFi connection status, and current switch connection status as it happens. These outputs are the main method for determining if all Wemos Filament Sensor systems are working as intended.

5. If problems persist after attempting to go though the set-up procedure with the serial debugger open then verify there is not obvious hardware failure by inspecting the filament switch and control box. Use the System Documentation for reference.

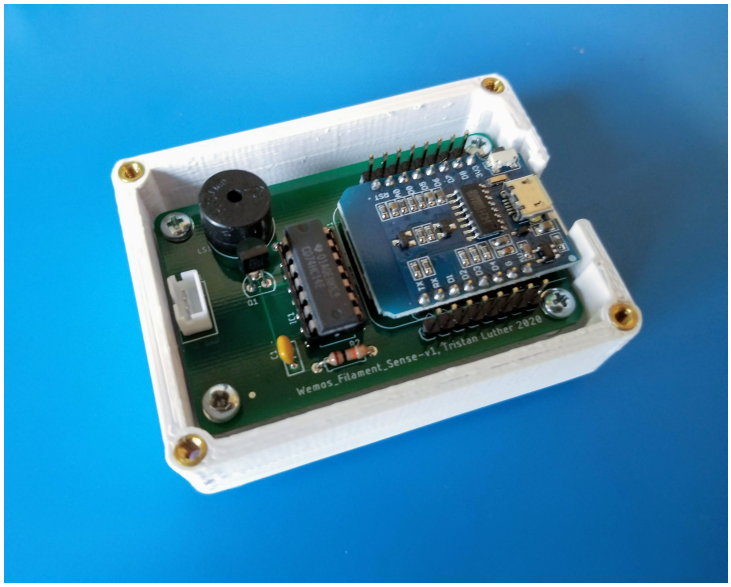
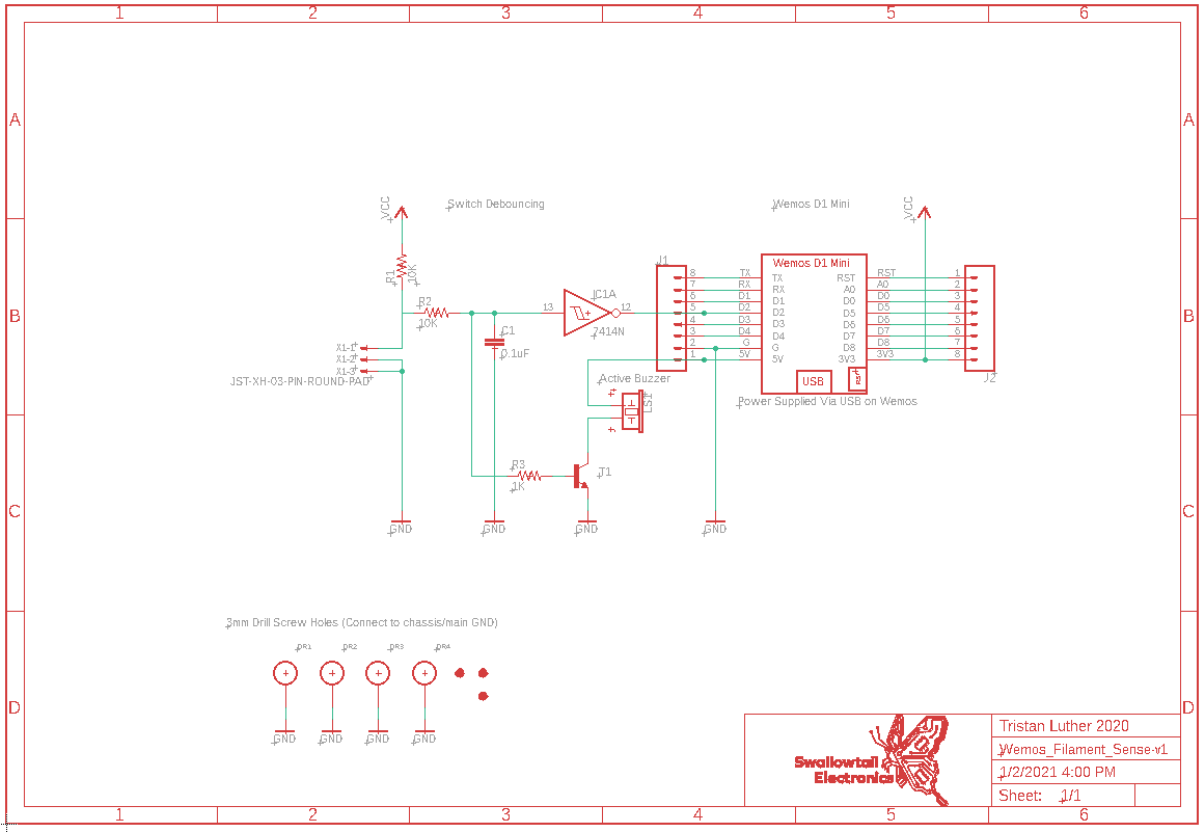


Figure 2.3: Wemos PCB

Chapter 3

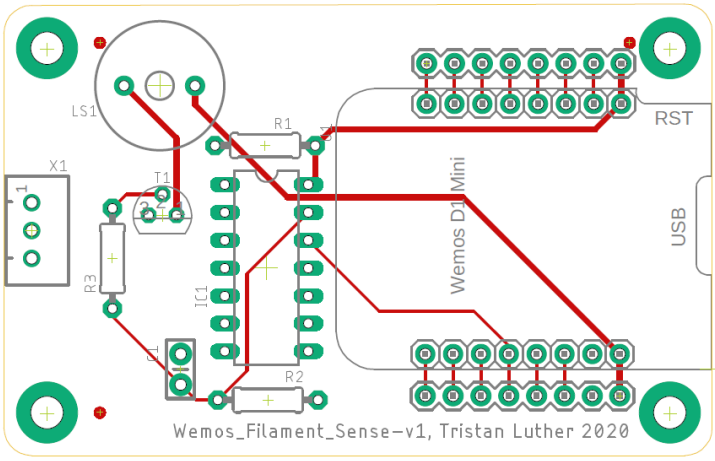
System Documentation

3.1 System Schematic

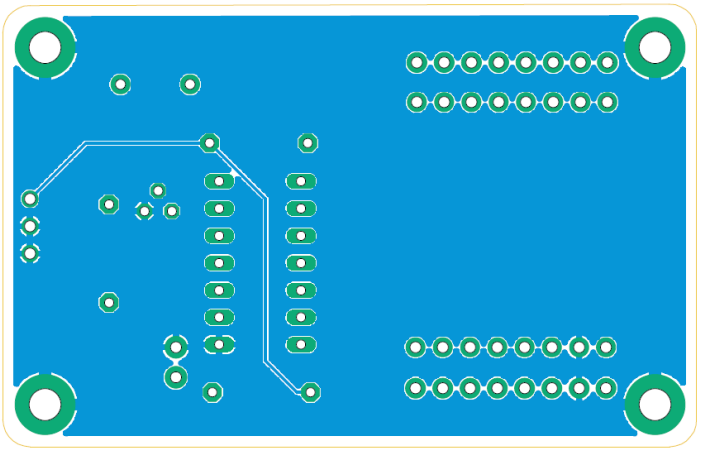


3.2 System PCB Layers

3.2.1 Top Copper & Silkscreen



3.2.2 Bottom Copper & Silkscreen



3.3 System Firmware

The system has three different firmware versions. The first is the default and recommended method which operates off of an Access Point that converts to a Client after filling out an HTML form with the necessary information to connect to the local WiFi send an email. The second is a client only approach where the user must edit the source code to configure the local WiFi, SMTP Server, Port, and email content. The advantage to this method is that in the event of the Wemos being disconnected from power it will not require the set-up form to be filled out again. The third and final method is the No Internet version in-which there is no email sent to the user but rather the on-board piezo buzzer is the main method to alert the user that your

3D printer is out of filament. This final method only uses the Wemos for the USB power and as a serial debugger for the status of the switch.

The source code for each is available on GitHub (`https://github.com/tristanluther28/email_filament_sensor`) as different branches and is listed in the following subsections.

3.3.1 Access Point/Client Hybrid (Default)

```
/*
 * Filename: wemos_filament_sense.ino
 * Author: Tristan Luther
 * Date: 3/23/2020
 * Purpose: Email Client to alert user of empty 3D printer.
 */

/*************** Libraries ********************/
#include "Arduino.h" //Tinfoil hat
#include <EMailSender.h> //Email Sending Class
#include <ESP8266WiFi.h> //802.11 Managment
#include <ESP8266WebServer.h>
#include <ESP8266mDNS.h>
/*
   encoded with  xxd -i index.html > buffer.h
   unsigned char needed to be changed to const char
   null-terminate the end of the array (0x00)
   save to flash mem with: ICACHE_RODATA_ATTR
*/
#include "buffer.h" //Contains HTML page

/************** Macros/Globals ****************/

//Email using Gmail use port 465 (SSL) and SMTP Server smtp.gmail.com
//The sending account needs the less secure app option https://myaccount.google.com/lesssecureapps?pli=1

//The two variables below set the SSID and Password for the ESP Access Point to fill out the form.
#define APSSID "WemosFilamentSensor"
#define APPSK  "wem0sfun!"
```

```
31 //Access Point SSID & Password
   const char *ssid = APSSID;
33 const char *password = APPSK;

35 //Client Network Credentials (will be filled out in the html form)
   String clissid;
37 String clipassword;

39 ESP8266WebServer server(80); //Server on port 80

41 //Variables below will contain the data form the users GET request form. This
       will be used in the client mode
   String emailSenderAccount; //Username of the account sending the email
43 String emailSenderPassword; //Password of the account sending the email
   String emailRecipient; //Who is receiving the email
45 String smtpServer; //Location of SMTP being used
   String smtpServerPort; //gmail SSL SMTP Port number
47 String emailSubject; //Subject line of the email to be sent
   String emailMessage; //Message in the email to be sent
49 char charBufferOne[512]; //Buffer for converting from string
   char charBufferTwo[512]; //Buffer for converting from string
51 char charBufferThree[512]; //Buffer for converting from string
   char charBufferFour[512]; //Buffer for converting from string
53 char charBufferFive[512]; //Buffer for converting from string
   char charBufferSix[512]; //Buffer for converting from string
55 char charBufferSeven[512]; //Buffer for converting from string

57 bool clientMode = false; //Variable for if we should be in either client or
       access point mode
   bool activeMode = false; //State machine variable for determining if we are
       looking for positive edge (filament loaded) or negative edge (filament gone)
59 uint8_t filamentSwitchPin = 4; //Pin that the filament switch is hooked up to (
       Wemos pin D2)
   bool currentStatus = false; //Used for debouncing of switch
61 bool lastStatus = false; //Used for debouncing of switch

63 /***************** Functions *****************/
```

```
65 //Send the HTML page to the user when someone connects
   void handleRoot() {
67  server.send(200, "text/html", index_html); //Send web page
   }
69
   //Handler for when the user submits the form
71 void handleForm() {
    clissid = server.arg("ssid");
73  clipassword = server.arg("wifipass");
    emailSenderAccount = server.arg("emailsen");
75  emailSenderPassword = server.arg("senpass");
    emailRecipient = server.arg("emailrec");
77  smtpServer = server.arg("serveraddr");
    smtpServerPort = server.arg("port");
79  emailSubject = server.arg("sub");
    emailMessage = server.arg("bod");
81
    Serial.println("Recieved from the user:");
83  Serial.print("SSID: ");
    Serial.println(clissid);
85  Serial.print("Password: ");
    Serial.println(clipassword);
87  Serial.print("Sender Account: ");
    Serial.println(emailSenderAccount);
89  Serial.print("Sender Account Password: ");
    Serial.println(emailSenderPassword);
91  Serial.print("Recipient Account: ");
    Serial.println(emailRecipient);
93  Serial.print("SMTP Server: ");
    Serial.println(smtpServer);
95  Serial.print("SMTP Port: ");
    Serial.println(smtpServerPort);
97  Serial.print("Email Subject Line: ");
    Serial.println(emailSubject);
99  Serial.print("Email Body: ");
    Serial.println(emailMessage);
101
    String s = "<h1>Succesfully Configured! Please disconnect</h1>";
103 server.send(200, "text/html", s); //Send web page
```

```
105  //Put the device is clientMode
     clientMode = true;
107  Serial.println("Leaving AP Mode/Switching to Client Mode");
}
109
//Function to set-up the Wemos Access Point
111 void WiFiAccessPoint(){
      WiFi.mode(WIFI_AP); //Switching to AP mode
113   WiFi.softAP(ssid, password); //Initializing the AP with ssid and password. This
        will create a WPA2-PSK secured AP
      IPAddress myIP = WiFi.softAPIP();
115   Serial.println("");
      Serial.print("AP IP address: ");
117   Serial.println(myIP);
      WiFi.begin(); //Begin transactions
119   if (!MDNS.begin("wemos")) {             // Start the mDNS responder for wemos.
       local
        Serial.println("Error setting up MDNS responder!");
121   }
      Serial.println("mDNS responder started");
123   return;
}
125
//Function to connect to WiFi with the given AP SSID & Password
127 void WiFiConnect(){
      server.close();
129   WiFi.softAPdisconnect(); //Disconnect from the previous mode
      WiFi.disconnect(); //Disconnect from pre-existing connections
131   WiFi.mode(WIFI_STA);
      Serial.println(); //Clear the line
133   Serial.print("Connecting"); //Print the status of the WiFi connection
      //Create the WiFi connection
135   WiFi.begin(clissid, clipassword);
      //Until connected, wait with prints to the terminal
137   while (WiFi.status() != WL_CONNECTED) {
        Serial.print(".");
139     delay(200);
      }
```

```
141   //Print the status of the WiFi connection
      Serial.println();
143   Serial.println("WiFi connected.");
      return;
145 }

147 //Parses the string to convert it to a unsigned 16-bit integer
    uint16_t Parse(String str){
149   char buff[64];
      str.toCharArray(buff, 64);
151   uint16_t num = (uint16_t)strtol(buff, NULL, 10);
      return num;
153 }

155 //Prepare the email to be sent
    void sendEmail(){
157   uint16_t port = Parse(smtpServerPort);
      Serial.print("Using Port: ");
159   Serial.println(port);
      //Sending data object contains config and data to send
161   emailSenderAccount.toCharArray(charBufferOne, 512);
      emailSenderPassword.toCharArray(charBufferTwo, 512);
163   smtpServer.toCharArray(charBufferThree, 512);
      smtpServerPort.toCharArray(charBufferFour, 512);
165   emailSubject.toCharArray(charBufferFive, 512);
      emailMessage.toCharArray(charBufferSix, 512);
167   emailRecipient.toCharArray(charBufferSeven, 512);
      //EMailSender emailSend(charBufferOne, charBufferTwo);
169   //The deafult values though the EMailSender class expect a SSL STMP Gmail
        connection/account
      //Can be changed with the commands below:
171   EMailSender emailSend(charBufferOne, charBufferTwo, charBufferOne,
        charBufferThree, port);
      //Connect to the SMTP server
173   EMailSender::EMailMessage message; //Delcare struct for SMTP
      //Modify the SMTP Struct based on the settings given in the Macros
175   message.subject = charBufferFive;
      message.message = charBufferSix;
177   //Fill the email response struct from the send
```

```
  EMailSender::Response resp = emailSend.send(charBufferSeven, message);
179  //Update the debug
  Serial.println("Sending status: ");
181  //Print to debug terminal
  Serial.println(resp.status);
183  Serial.println(resp.code);
  Serial.println(resp.desc);
185  return;
}
187
//Software debouncing function
189 bool debounceRead(bool last){
  //Read the current state of the button
191  bool current = digitalRead(filamentSwitchPin);
  //Check if the last recorded state is the same as the current
193  if(last != current){
    //Delay for 5ms to let the bouncing stop
195    delay(5);
    //Read the actual state
197    current = digitalRead(filamentSwitchPin);
  }
199  //Return the status
  return current;
201 }

203 /****************** Set-up ******************/
void setup(){
205  Serial.begin(115200); //Initialize UART connection at 115200 baud rate for
    debugging

207  //Set-up the pinouts
  pinMode(INPUT, filamentSwitchPin);
209  //Begin access point mode
  WiFiAccessPoint();
211
  //Function to handle the user first signing on
213  server.on("/", handleRoot);
  //Form to handle the user's submit
215  server.on("/recieve", handleForm);
```

```
  //Begin the webserver
217  server.begin();
  //While we are still waiting for the GET Request to set-up email/wifi, wait
    here
219  while(!clientMode){
    //Handle the clients requests while they are still in AP mode
221    server.handleClient();
  }
223
  //Set-up the WiFi connection
225  WiFiConnect();
}
227
/****************** Loop ********************/
229 void loop(){
 //If the button is low then wait for it to be high
231  if(!activeMode){
    //Serial.println("We are in: Non-active Mode");
233    //This block is the non-active mode. The non-active mode waits for a logic 1
    to enter the active mode
    //Software Debouncing
235    currentStatus = debounceRead(filamentSwitchPin);
    //If we just detected a positive edge
237    if(currentStatus == HIGH && lastStatus == LOW){
      //We are moving into active mode
239      activeMode = true;
      //Turn the buzzer off
241      Serial.println("Filament Loaded: Turning off alarm!");
    }
243    //Otherwise the current status becomes the previous
    lastStatus = currentStatus;
245  }
  else{
247    //This block is the active mode. The filament has been loaded and will send
    the email if a logic 0 is detected
    //Serial.println("We are in: Active Mode");
249    //Software Debouncing
    currentStatus = debounceRead(filamentSwitchPin);
251    //If we just detected a negative edge
```

```
    if(currentStatus == LOW && lastStatus == HIGH){
253   //Sound the alarm by sending the email
      Serial.println("Filament Gone: Sending the email/sounding alarm!");
255   sendEmail();
      //We are moving out of active mode
257   activeMode = false;
    }
259 //Otherwise the current status becomes the previous
    lastStatus = currentStatus;
261 }

263 }
```

3.3.2 Client Only

```
1 /*
  * Filename: wemos_filament_sense.ino
3 * Author: Tristan Luther
  * Date: 3/23/2020
5 * Purpose: Email Client to alert user of empty 3D printer.
  */
7

/*************** Libraries ********************/
9 #include "Arduino.h"
  #include <EMailSender.h>
11 #include <ESP8266WiFi.h>

13 /************** Macros/Globals ***************/

15 //Email using Gmail use port 465 (SSL) and SMTP Server smtp.gmail.com
  //The sending account needs the less secure app option https://myaccount.google.
    com/lesssecureapps?pli=1
17 #define emailSenderAccount    "SENDER_EMAIL_USERNAME" //Username of the account
    sending the email
  #define emailSenderPassword   "SENDER_EMAIL_PASSWORD" //Password of the account
    sending the email
19 #define emailRecipient        "RECIEVER_EMAIL_USERNAME" //Who is receiving the
    email
```

```
#define smtpServer              "smtp.gmail.com" //Location of SMTP being used
#define smtpServerPort          465 //gmail SSL SMTP Port number
#define emailSubject            "Lulzbot TAZ6: Moarstruder | Out of Filament" //
    Subject line of the email to be sent
#define emailMessage            "Lulzbot TAZ6: Moarstruder has run out of fillament
     (filament sensor has been triggered).\n- Wemos D1 Mini Filament Sensor" //
    Message in the email to be sent

//The two variables below must be replaced with local network credentials
const char* ssid = "WIFI_SSID";
const char* password = "WIFI_PASSWORD";

//Sending data object contains config and data to send
EMailSender emailSend(emailSenderAccount, emailSenderPassword);
//The deafult values though the EMailSender class expect a SSL STMP Gmail
    connection/account
//Can be changed with the commands below:
//EMailSender emailSend(emailSenderAccount, emailSenderPassword,
    emailSenderAccount, smtpServer, smtpServerPort);

bool activeMode = false; //State machine variable for determining if we are
    looking for positive edge (filament loaded) or negative edge (filament gone)
uint8_t filamentSwitchPin = 4; //Pin that the filament switch is hooked up to (
    Wemos pin D2)
bool currentStatus = false; //Used for debouncing of switch
bool lastStatus = false; //Used for debouncing of switch

/***************** Functions *****************/

//Function to connect to WiFi with the given AP SSID & Password
void WiFiConnect(){
  Serial.println(); //Clear the line
  Serial.print("Connecting"); //Print the status of the WiFi connection
  //Create the WiFi connection
  WiFi.begin(ssid, password);
  //Until connected, wait with prints to the terminal
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print(".");
    delay(200);
```

```
  }
  //Print the status of the WiFi connection
  Serial.println();
  Serial.println("WiFi connected.");
  Serial.println();
  return;
}

//Prepare the email to be sent
void sendEmail(){
  //Connect to the SMTP server
  EMailSender::EMailMessage message; //Delcare struct for SMTP
  //Modify the SMTP Struct based on the settings given in the Macros
  message.subject = emailSubject;
  message.message = emailMessage;
  //Fill the email response struct from the send
  EMailSender::Response resp = emailSend.send(emailRecipient, message);
  //Update the debug
  Serial.println("Sending status: ");
  //Print to debug terminal
  Serial.println(resp.status);
  Serial.println(resp.code);
  Serial.println(resp.desc);
  return;
}

//Software debouncing function
bool debounceRead(bool last){
  //Read the current state of the button
  bool current = digitalRead(filamentSwitchPin);
  //Check if the last recorded state is the same as the current
  if(last != current){
    //Delay for 5ms to let the bouncing stop
    delay(5);
    //Read the actual state
    current = digitalRead(filamentSwitchPin);
  }
  //Return the status
  return current;
```

```
91 }

93 /****************** Set-up ******************/
   void setup(){
95   Serial.begin(115200); //Initialize UART connection at 115200 baud rate for
       debugging

97   //Set-up the pinouts
     pinMode(INPUT, filamentSwitchPin);
99
     //Set-up the WiFi connection
101  WiFiConnect();
   }
103
   /****************** Loop ********************/
105 void loop(){
    //If the button is low then wait for it to be high
107  if(!activeMode){
       //Serial.println("We are in: Non-active Mode");
109    //This block is the non-active mode. The non-active mode waits for a logic 1
      to enter the active mode
       //Software Debouncing
111    currentStatus = debounceRead(filamentSwitchPin);
       //If we just detected a positive edge
113    if(currentStatus == HIGH && lastStatus == LOW){
         //We are moving into active mode
115      activeMode = true;
         //Turn the buzzer off
117      Serial.println("Filament Loaded: Turning off alarm!");
       }
119    //Otherwise the current status becomes the previous
       lastStatus = currentStatus;
121  }
     else{
123    //This block is the active mode. The filament has been loaded and will send
      the email if a logic 0 is detected
       //Serial.println("We are in: Active Mode");
125    //Software Debouncing
       currentStatus = debounceRead(filamentSwitchPin);
```

```
127     //If we just detected a negative edge
        if(currentStatus == LOW && lastStatus == HIGH){
129       //Sound the alarm by sending the email
          Serial.println("Filament Gone: Sending the email/sounding alarm!");
131       sendEmail();
          //We are moving out of active mode
133       activeMode = false;
        }
135     //Otherwise the current status becomes the previous
        lastStatus = currentStatus;
137   }

139 }
```

3.3.3 No Internet

```
1  /*
    * Filename: wemos_filament_sense.ino
3   * Author: Tristan Luther
    * Date: 3/23/2020
5   * Purpose: Wemos alert user of empty 3D printer.
    */
7
   /*************** Libraries ******************/
9  #include "Arduino.h" //Tinfoil hat

11 /************** Macros/Globals ***************/

13 bool activeMode = false; //State machine variable for determining if we are
       looking for positive edge (filament loaded) or negative edge (filament gone)
   uint8_t filamentSwitchPin = 4; //Pin that the filament switch is hooked up to (
       Wemos pin D2)
15 bool currentStatus = false; //Used for debouncing of switch
   bool lastStatus = false; //Used for debouncing of switch
17
   /***************** Functions *****************/
19
   //Software debouncing function
```

```
21 bool debounceRead(bool last){
   //Read the current state of the button
23 bool current = digitalRead(filamentSwitchPin);
   //Check if the last recorded state is the same as the current
25 if(last != current){
     //Delay for 5ms to let the bouncing stop
27   delay(5);
     //Read the actual state
29   current = digitalRead(filamentSwitchPin);
   }
31 //Return the status
   return current;
33 }

35 /****************** Set-up ******************/
   void setup(){
37 Serial.begin(115200); //Initialize UART connection at 115200 baud rate for
     debugging

39 //Set-up the pinouts
   pinMode(INPUT, filamentSwitchPin);
41 }

43 /****************** Loop ********************/
   void loop(){
45  //If the button is low then wait for it to be high
    if(!activeMode){
47    //Serial.println("We are in: Non-active Mode");
      //This block is the non-active mode. The non-active mode waits for a logic 1
     to enter the active mode
49    //Software Debouncing
      currentStatus = debounceRead(filamentSwitchPin);
51    //If we just detected a positive edge
      if(currentStatus == HIGH && lastStatus == LOW){
53      //We are moving into active mode
        activeMode = true;
55      //Turn the buzzer off
        Serial.println("Filament Loaded: Turning off alarm!");
57    }
```

```
    //Otherwise the current status becomes the previous
59    lastStatus = currentStatus;
  }
61  else{
    //This block is the active mode. The filament has been loaded and will send
    the email if a logic 0 is detected
63    //Serial.println("We are in: Active Mode");
    //Software Debouncing
65    currentStatus = debounceRead(filamentSwitchPin);
    //If we just detected a negative edge
67    if(currentStatus == LOW && lastStatus == HIGH){
      //Sound the alarm by sending the email
69      Serial.println("Filament Gone: Sounding alarm!");
      //We are moving out of active mode
71      activeMode = false;
    }
73    //Otherwise the current status becomes the previous
    lastStatus = currentStatus;
75  }

77 }
```

3.4 System Enclosure

The system enclosure is available on Thingiverse () and drawings are listed below.

3.5 Bill of Materials

Table 3.1: Bill of Materials

Item	Name	Qty.
1	Wemos D1 Mini	1
2	74HC14 (DIP-14)	1
3	2N4401 NPN (TO-92)	1
4	1k 1/4W Resistor	1
5	10k 1/4W Resistor	2
6	0.1uF Ceramic Capacitor	1
7	SPDT Mircoswitch	1
8	3-Pin JST Connector Pair	1
9	Control Box Base	1
10	Control Box Lid	1
11	Filament Sensor Case	1
12	Wemos Filament Sensor PCB	1
13	Zip Tie (WIT-18R-UVBM)	1
14	93365A122 Heat-Set Insert	8
15	90272A110 4-40 1/2” Long Phillips Head Screw	4
16	90272A106 4-40 1/4” Long Phillips Head Screw	4
17	90480A003 2-56 Hex Nut	3
18	91772A082 2-56 5/8” Long Phillips Head Screw	3