**Burglar Control**

***(Using Arduino Uno, Wi-fi Shield, and an ultrasonic sensor)***

Burglar control is a system that alerts a person in case the safety of something valuable is compromised. We often are away from our homes at work or school, and there is always a danger of burglary. The proposed system is aimed at monitoring the status of the doors and windows of a house. Necessary action can be taken if something wrong is suspected, and mishaps can be prevented.

The assembly consists of an Arduino UNO rev 3 with a wi-fi shield (ESP8266 Serial wi-fi Shield) stacked upon it. The primary sensor will be an ultrasonic sensor to measure the distance between a door or window frame and the assembly. The assembly will be placed at a fixed pre-known distance from the object being guarded. Let us take the example of the door of a house. If the door is open, the distance read will obviously less than the fixed pre-known distance. It may be greater than that if some object is situated outside the door with the door open. In every case, the fixed distance (with a minimal deviation in measurement) is the safety standard. This sensor data is used up to recognize the scenario, i.e., whether the door is open or not in the absence of the owner. The code is written following the data from the sensor and is then manipulated to send a message to the owner with the help of the wi-fi shield. The shield is connected to the local wireless network(wi-fi) of the house, and a 9V battery will power the Arduino. The information can be obtained by just accessing the IP address of the shield from any browser.

Some details:

* The code is uploaded to the Arduino board, and the laptop is disconnected. Since the Arduino stores the previous code in its memory, it will execute the same code on powering it with an external power source (like a 9V battery). So, there is no need to keep the laptop plugged into the Arduino for long hours.
* The local wi-fi network will ensure that the shield is connected to the internet. The IP address of the shield (on the sticker) will be used to access the information. All the required information is implied in the code.
* The ultrasonic sensor determines the time duration that an ultrasonic pulse takes to return to the receiver from the transmitter and uses this duration and the speed of sound to determine the distance of the object from the sensor:

*Distance=((speed of sound)\*(duration))/2*

The division by two is evident in the fact that the sound pulse travels the distance twice.

* The assembly will be placed at a distance of 100cms (the pre-known distance) from the door, and there will be a minimal range of 2cms for the measurement of the distance to account for the fluctuations of the speed of sound with humidity and temperature.
* However, accuracy can be improved by using a DTH Sensor used to measure the humidity and temperature of the surroundings. From these data, the speed of sound can be calculated as follows:

Speed of sound=331.4+(0.606\*temperature) +(0.0124\*humidity)

Code:

#include <SPI.h>

#include <WiFi.h>

#include <dht.h>

char ssid[] = "yourNetwork";      // your network SSID (name)

char pass[] = "secretPassword";   // your network password

int keyIndex = 0;                 // your network key Index number (needed only for WEP)

#define DHT11\_PIN 7

const int trigPin = 9;

const int echoPin = 10;

long duration;

int distanceCm;

float speedS, hum, temp;

int status = WL\_IDLE\_STATUS;

WiFiServer server(80);

void setup() {

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

//Initialize serial and wait for port to open:

  Serial.begin(9600);

  while (!Serial) {

    ; // wait for serial port to connect. Needed for native USB port only

  }

  // check for the presence of the shield:

  if (WiFi.status() == WL\_NO\_SHIELD) {

    Serial.println("WiFi shield not present");

    // don't continue:

    while (true);

  }

  String fv = WiFi.firmwareVersion();

  if (fv != "1.1.0") {

    Serial.println("Please upgrade the firmware");

  }

  // attempt to connect to Wifi network:

  while (status != WL\_CONNECTED) {

    Serial.print("Attempting to connect to SSID: ");

    Serial.println(ssid);

    // Connect to WPA/WPA2 network. Change this line if using open or WEP network:

    status = WiFi.begin(ssid, pass);

    // wait 10 seconds for connection:

    delay(10000);

  }

  server.begin();

  // you're connected now, so print out the status:

  printWifiStatus();

}

void loop() {

   digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

hum=dht.readHumidity();

temp=dht.readTemperature();

speedS=331.4+(0.606\*temp)+(0.0124\*hum);

speedS=speedS/10000; //converting into cm/ms

duration = pulseIn(echoPin, HIGH); //duration in ms

distanceCm= (duration\*speedS)/2;//distance calculation from door

         int dmin=99;

         int dmax=101;

  // listen for incoming clients

  WiFiClient client = server.available();

  if (client) {

    Serial.println("new client");

    // an http request ends with a blank line

    bool currentLineIsBlank = true;

    while (client.connected()) {

      if (client.available()) {

        char c = client.read();

        Serial.write(c);

        // if you've gotten to the end of the line (received a newline

        // character) and the line is blank, the http request has ended,

        // so you can send a reply

        if (c == '\n' && currentLineIsBlank) {

          // send a standard http response header

          client.println("HTTP/1.1 200 OK");

          client.println("Content-Type: text/html");

          client.println("Connection: close");  // the connection will be closed after completion of the response

          client.println("Refresh: 5");  // refresh the page automatically every 5 sec

          client.println();

          client.println("<!DOCTYPE HTML>");

          client.println("<html>");

         if((distanceCm<=dmax)&&(distanceCm>=dmin))

         client.println("Everything is ok.");

         else

         client.println("Alert! The door has been opened.");

          client.println("</html>");

          break;

        }

        if (c == '\n') {

          // you're starting a new line

          currentLineIsBlank = true;

        } else if (c != '\r') {

          // you've gotten a character on the current line

          currentLineIsBlank = false;

        }

      }

    }

    // give the web browser time to receive the data

    delay(1);

    // close the connection:

    client.stop();

    Serial.println("client disonnected");

  }

}

void printWifiStatus() {

  // print the SSID of the network you're attached to:

  Serial.print("SSID: ");

  Serial.println(WiFi.SSID());

  // print your WiFi shield's IP address:

  IPAddress ip = WiFi.localIP();

  Serial.print("IP Address: ");

  Serial.println(ip);

  // print the received signal strength:

  long rssi = WiFi.RSSI();

  Serial.print("signal strength (RSSI):");

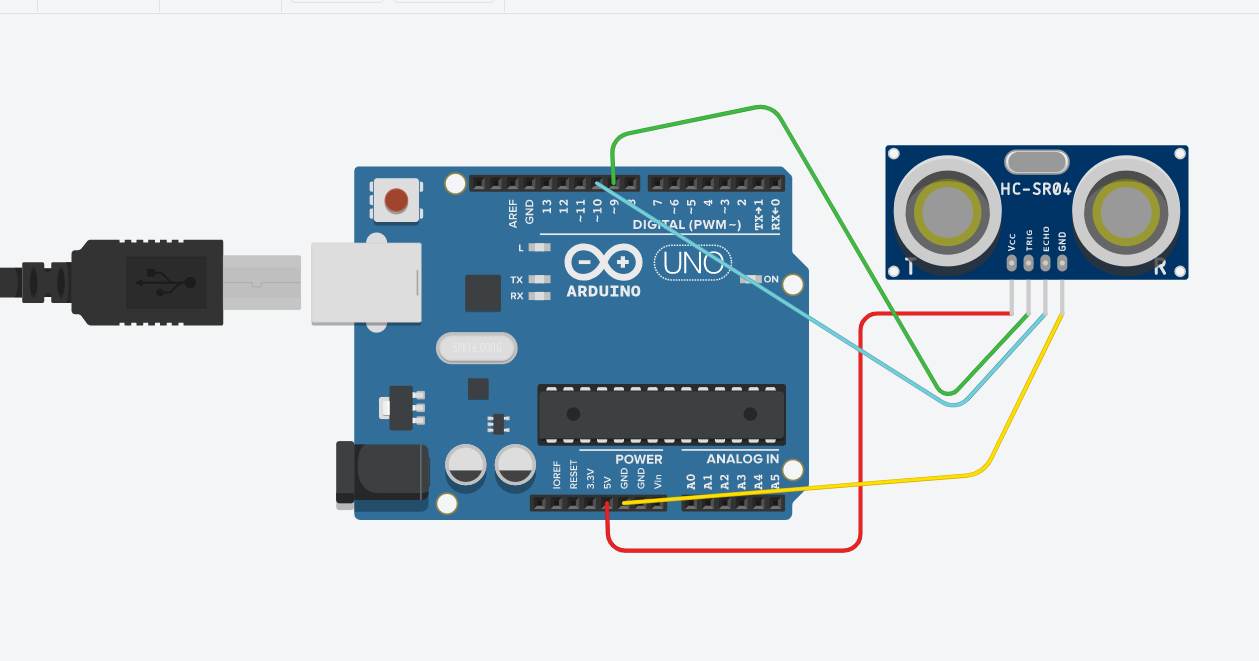
  Serial.print(rssi);

  Serial.println(" dBm");

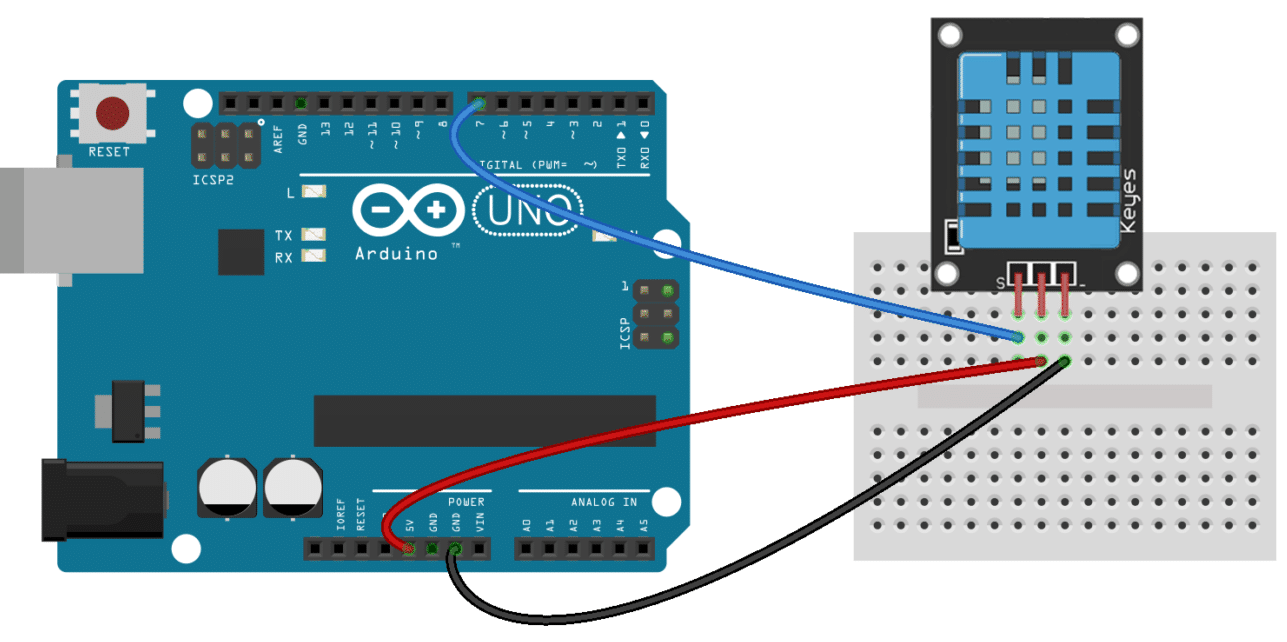
}

The following are the circuit connections of the respective sensors. As the Wi-fi shield is stacked over to the Arduino, all connections shown are made to the pins of the shield(equivalent to the circuit):



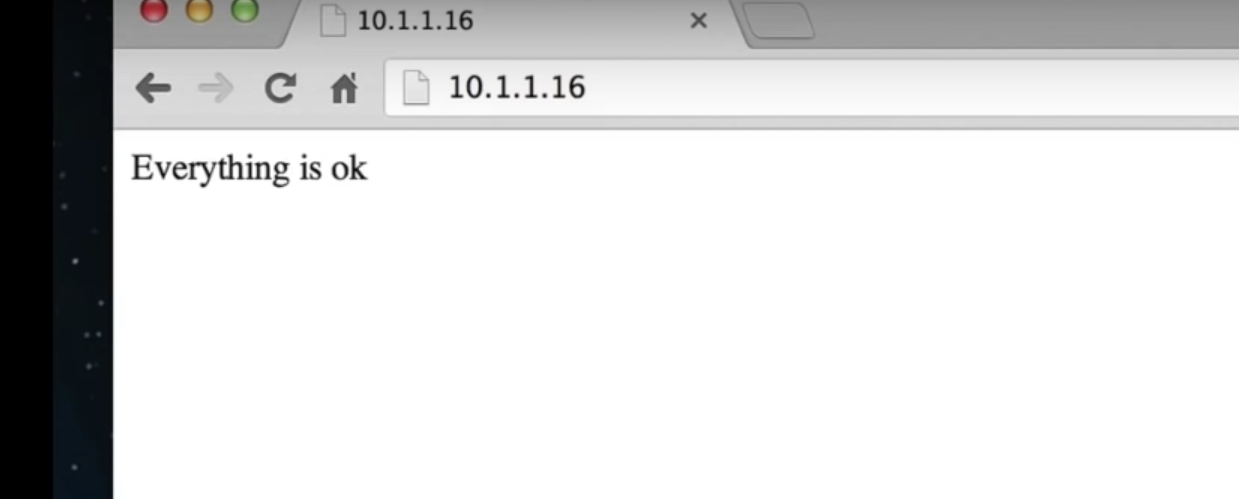


The ultrasonic sensor circuit

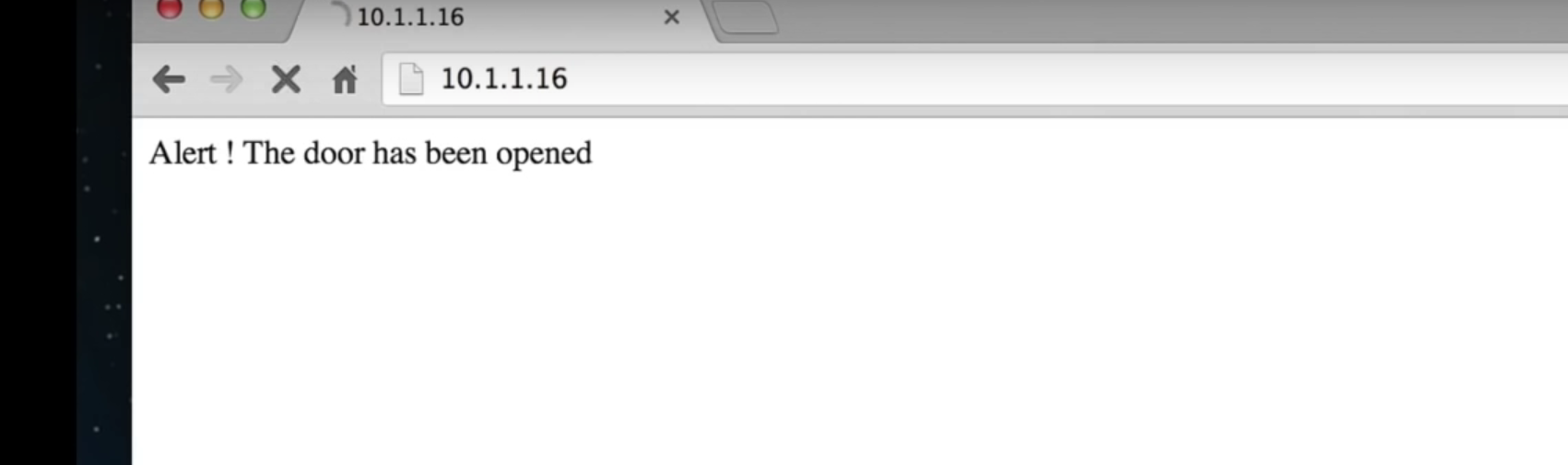


The DTH sensor circuit

The following messages will be shown on typing the IP address in a browser:



***When the door is closed***



***When the door is open***