Microcontroller systems final report: RF live trap sensor

12/12/2024

Ben Mays

Components used:

arduino uno		2x
433 mhz RF transmitter	5Pcs 433MHz RF Wireless Transmitter and Receiver Module Kit for ARM/MCU	1x
433 mhz RF transmitter	5Pcs 433MHz RF Wireless Transmitter and Receiver Module Kit for ARM/MCU	1x
W5500	HiLetgo W5500 SPI to LAN Ethernet Network Module TCP IP STM32 Interface 3.3V 5V for Arduino WIZ820io RC5	1x
buzzer speaker	-	1x
LED(red)	-	1x
resistors: 220 ohm, 50k	-	1x, 2x
assorted wires and breadboards		

Introduction:

1) 433mhz RF transmission

The first step of this project was to ensure the RF link was functional. connection was relatively simple, using the radiohead.h library, however the signal was quite weak, and could only transmit over the distance of a few feet. connecting the system to a 9v battery and tuning the receiver produced some positive results, but the most beneficial change was the addition of a pair of 17cm wire antennas, ¼ the wavelength of the 433mhz signal. This expanded the range to allow the system to transmit over a distance of approximately 15m, which was deemed sufficient at this time.

To support this system, a button was added to the transmitter to act as the sensor for the trap closing, and a speaker and LED was added to the receiver to serve as a temporary alert.

figure 1: completed receiver. at time of completion, the ethernet could connect to the network but was still unable to send data

figure 2: completed transmitter. this section sends an "open" signal until the button is pressed, at which point it changes to "closed"

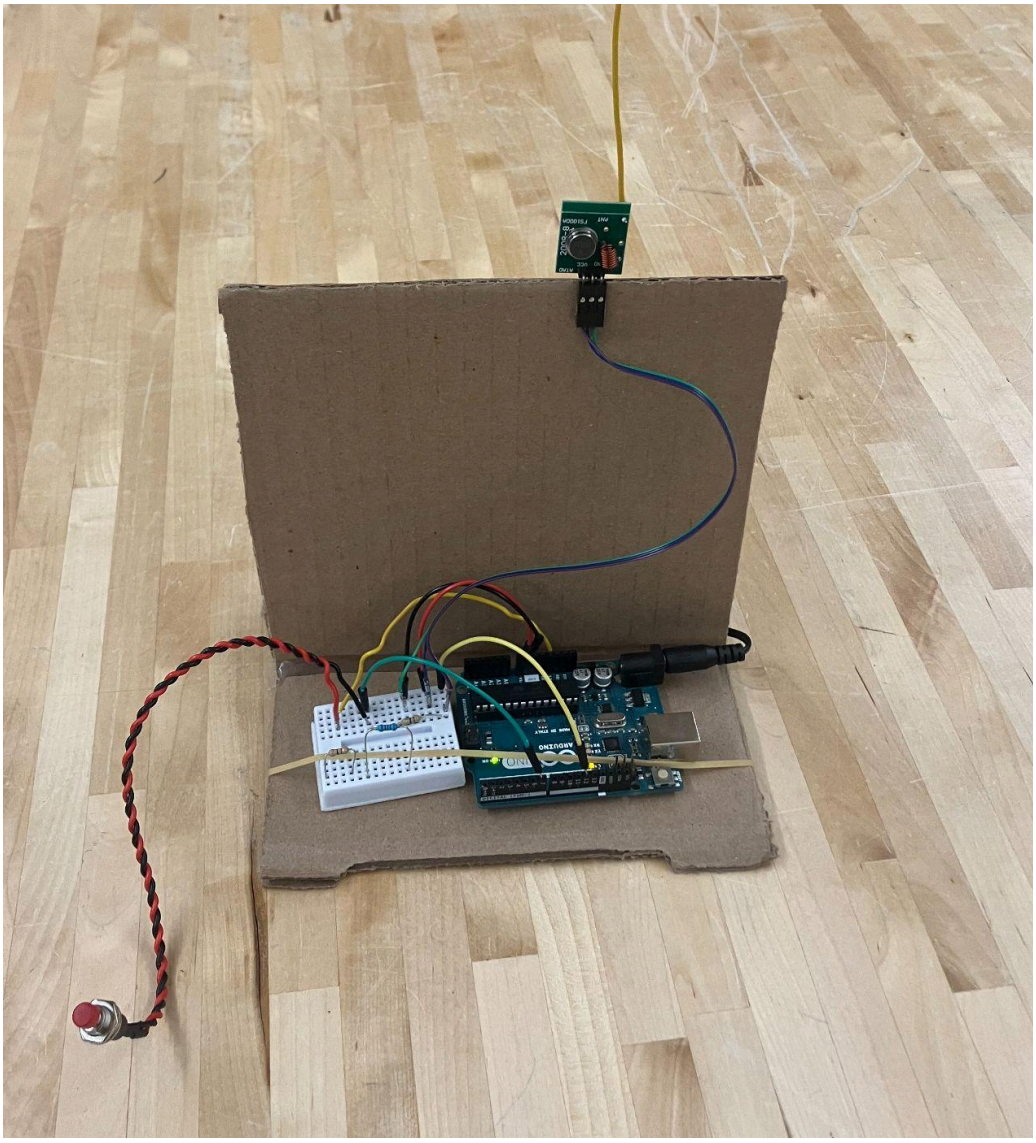
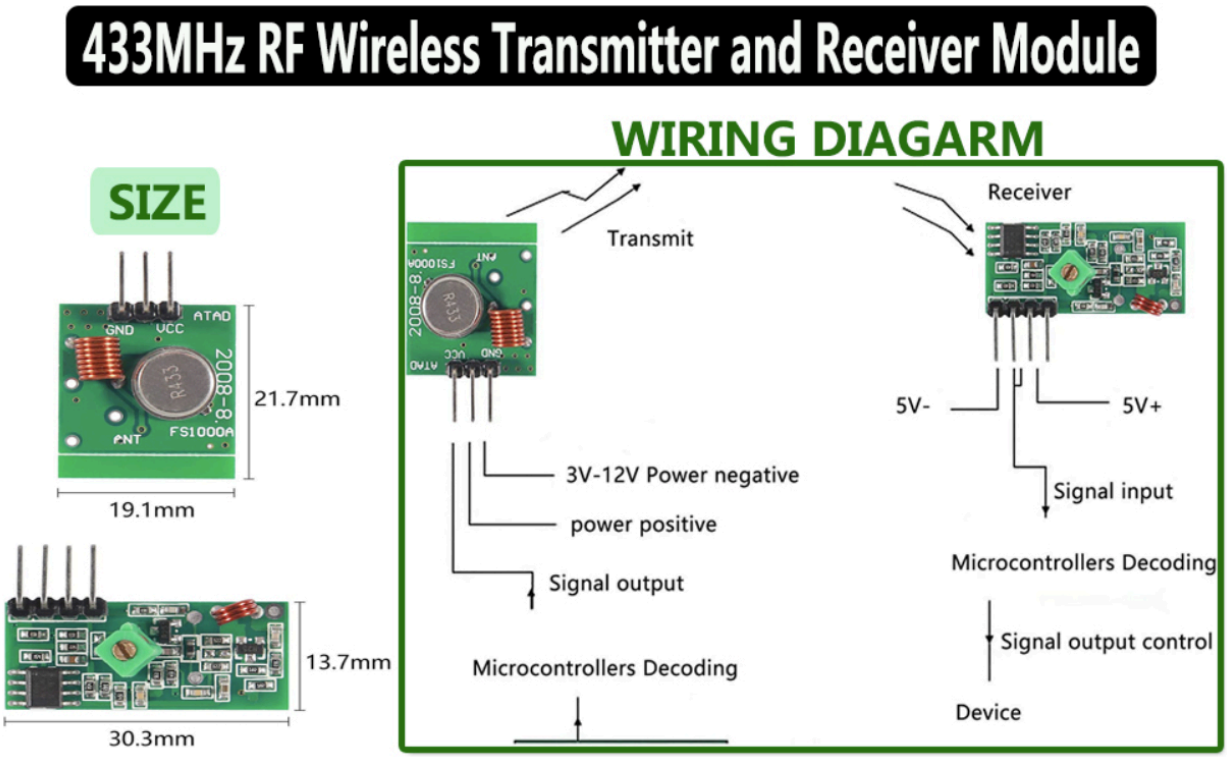


figure 3: details of RF link component



2) Ethernet connectivity

Internet connectivity proved to be significantly more challenging than expected. The first attempt involved using online resources such as IFTTT to connect and send a message, however many of these applications have been recently paywalled and were therefore considered inaccessible for this project. After no reasonable method of connecting to a resource capable of sending notifications, an attempt was made to connect to the local network to report status in a way that could be read by other devices, but this also failed, mostly due to time constraints. more work in the future could potentially make this work, or perhaps an upgrade to the board will allow the usage of a library such as EmailSender.h, which would work for this application but was unfortunately too large for the uno.

Conclusion:

While the failure of the online component is frustrating, the success of the RF link is undeniable, and has a variety of potential future applications as remote sensors. This project will likely be continued, with future goals focused around making a successful internet connection, and eliminating the need for an arduino in the transmitter to reduce costs and miniaturize.

Code:

```
// this code is for both the sender and receiver, comment out the undesired function
in loop and upload to each.
```

```
#include "Arduino.h"
#include <Ethernet.h>
#include <RH_ASK.h>
#include <SPI.h> // Not actually used but needed to compile



//========trying local internet!=======================
byte mac[]= {0x50, 0x7B, 0x9D, 0xEF,0x50, 0x7E};
//ip: uvm: 132.198.4.32
IPAddress ip(132, 198, 4, 32);
const int pinCS = 10;
EthernetServer server = EthernetServer(80);


//program for micro final
//transmits data between two arduinos using RF communication
RH_ASK driver(2000, 8, 12, 11);
bool isTriggered = false;
String message="open";
const char *opnmsg="opn";
const char *csdmsg="csd";

int recLedPin=5;
int recSpeakPin=6;
//email setup


void setup() {
  Serial.begin(9600);
  if (!driver.init())
  {
     Serial.println("init failed");
  }
  pinMode(7,INPUT);
  pinMode(5,OUTPUT);
  Serial.println("started");
```

```
  //email setup
  pinMode(pinCS, OUTPUT);
  digitalWrite(pinCS, LOW);

  Ethernet.begin(mac, ip);
  server.begin();
  Serial.print("server started on: ");
  Serial.println(Ethernet.localIP());

}

void loop() {


  //uncomment the one you want to send
  //sender();
  reciver();
  //maybe add in auto switcher? this is getting a little annoying

}



//put all the code for the receiver in here
//function waits and listens for signal from the sender
//if it receives a "csd" message, it will do nothing
//if it receives an "opn", it will send a message.
//on consecutive readings, it will simply flash a light and beep the
speaker
void reciver(){
  uint8_t buf[3];
  online();

  uint8_t buflen = sizeof(buf);
  if (driver.recv(buf, &buflen)) // Non-blocking
  {
    int i;
    // Message with a good checksum received, dump it.
    Serial.print("Message: ");
    Serial.println((char*)buf);
```

```
    Serial.println(sizeof(buf));

    //========turn on alert speaker===========
    //tone(6,200,200);

    if(buf[0] == opnmsg[0]){
      //don't send message/send all clear message
      isTriggered = false;
      message="Open";
      Serial.println("it's open");
      //also can be "reset"
    }
    if(buf[0] == csdmsg[0]){
      if(isTriggered==false){
        isTriggered=true;
        //first trigger: send message
        Serial.println("first trigger, sending message");
        tone(6,700,700);
        message="closed";
        //sendEmail();
      }
      else{
        //still triggered, don't send message but keep status
        Serial.println("still shut");
        digitalWrite(5,HIGH);
        tone(6,500,200);
        delay(200);
        digitalWrite(5,LOW);
      }
      //send message/
    }
  }

}

//put all code for sender in here
//function reads the switch on pin 7, and determines weither the trap is
open or not.
//it will then send either "opn" or "csd" through the transmitter, then
wait a second
```

```
void sender(){
  //char *msg;
  if(digitalRead(7)==HIGH){
  //*msg="opn";
  Serial.println("reading high");
  Serial.print("sending ");
  Serial.println((char *)csdmsg);
  driver.send((uint8_t *)csdmsg, 3);
  }
  else{
  //*msg="csd";
  Serial.println("reading low");
  Serial.print("sending ");
  Serial.println((char *)opnmsg);
  driver.send((uint8_t *)opnmsg, 3);
  }

  //driver.send((uint8_t *)msg, 6);
  driver.waitPacketSent();

  //Serial.println(strlen(opnmsg));

  delay(1000);

}

//nonfunctional
void online(){
  EthernetClient client = server.available();
  if(client){
    while(client.connected()){
      Serial.println("running");

      client.println("HTTP/1.1 200 OK");
      client.println("Content-Type: text/html");
      client.println();

      //body
      client.println("<html><body>");
      client.println("<h1>Holy crap it works</h1>");
```

```
      client.print("<p>your trap is:</p>");
      client.println(message);
    }
  }

}
```