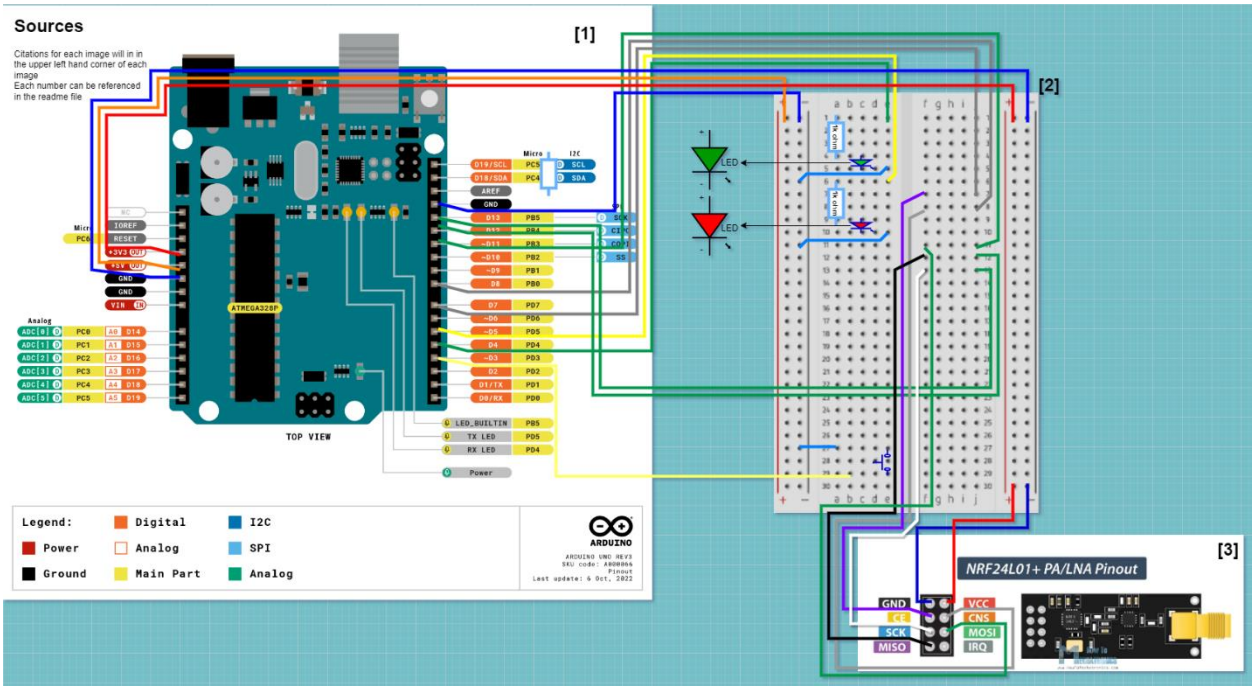
321 Term Project: 2.4 GHz Jammer

Patrick Wilkinson | prwilkin

University At Buffalo Undergraduate Computer Science

CSE 321: Realtime Embedded Systems

Hardware Component



NRF24L01 + PA/LNA Pins:

- GND -> R5.2-
- VCC -> R5.2+
- CE -> F7
- CNS -> F8
- SCK -> F13
- MOSI -> F11
- MISO ->F12

Arduino Uno Rev 3 Pins:

- PIN 3 -> 28B
- PIN 4 -> 1E
- PIN 5 -> 6E
- PIN 7 -> 7J
- PIN 8 -> 8J
- PIN 11 -> 11J
- PIN 12 -> 12J
- PIN 13 -> 13J
- 3.3V -> R1.1+
- 5V -> L1.1+
- LEFT GND -> R1.1-
- RIGHT GND -> L1.1-

Software Component

```
#include <SPI.h>
```

```
#include <nRF24L01.h>
#include <RF24.h>

RF24 radio(7, 8); // CE on Pin 7, CSN on Pin 8
const byte address[6] = "00001"; //meaningless but required
char payload[] = "Hello World"; //experiment with packet size
bool running = false;

void setup() {
  // debug with println
  Serial.begin(9600);
  Serial.println("Hello World");

  // Lights
  pinMode(4, OUTPUT); // Pin 4 green led
  pinMode(5, OUTPUT); // Pin 5 red led
  digitalWrite(5, HIGH); //red on
  Serial.println("Set Light: @");

  /* Button
  Button acts as a hardware interrupt. When pressed interruptHanlder is called to
process function.
  The interrupt is set to trggier on FALLING (FALLING for when the pin goes from
high to low.) since
  the pin is high on idle a press down triggers it. Other options:
  LOW to trigger the interrupt whenever the pin is low,
  CHANGE to trigger the interrupt whenever the pin changes value,
  RISING to trigger when the pin goes from low to high
  */
  pinMode(3, INPUT_PULLUP);  // Pin 3 button    High = 1 (True) Low = 0 (False)
  attachInterrupt(digitalPinToInterrupt(3), interruptHanlder, FALLING); //
Interrupt
  Serial.println("Set Button: @");

  // NRF24L01
  radio.begin();
  radio.openWritingPipe(address);
  radio.setPALevel(RF24_PA_MAX);
  radio.stopListening();
  radio.setAutoAck(false); // for debug only
  Serial.println("Set Radio: @");

  Serial.println("Set up complete");
}
```

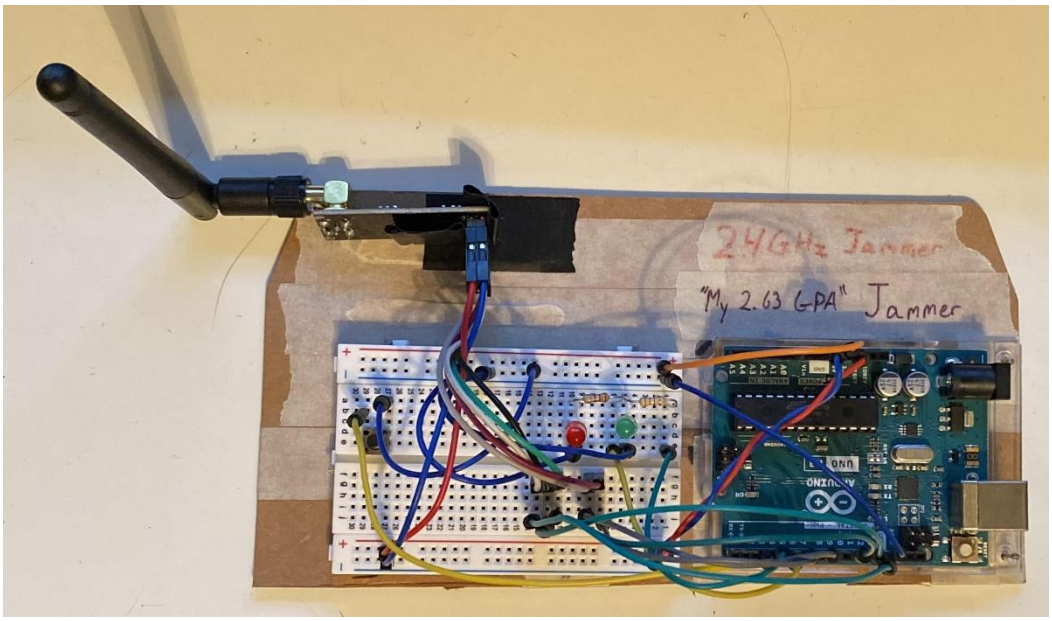
```
void loop() {
  if(running == true) {
    // www.everythingrf.com/community/2-4-ghz-wi-fi-802-11b-g-n-channels-and-
frequency-band
    // Center Freq - 2400 to get channel on NRF mod
    jam(12);  //WiFi channel 1    Bluetooth channel 4
    //jam(17);  //WiFi channel 2    Bluetooth channel 6-7
    jam(22);  //WiFi channel 3    Bluetooth channel 9
    //jam(27);  //WiFi channel 4    Bluetooth channel 11
    jam(32);  //WiFi channel 5    Bluetooth channel 13
    //jam(37);  //WiFi channel 6    Bluetooth channel 15-16
    jam(42);  //WiFi channel 7    Bluetooth channel 18
    //jam(47);  //WiFi channel 8    Bluetooth channel 20-21
    jam(52);  //WiFi channel 9    Bluetooth channel 23
    //jam(57);  //WiFi channel 10   Bluetooth channel 25-26
    jam(62);  //WiFi channel 11   Bluetooth channel 28
    //jam(67);  //WiFi channel 12   Bluetooth channel 30-31
    jam(72);  //WiFi channel 13   Bluetooth channel 33
    //jam(77);  //WiFi channel 14   Bluetooth channel 35-36
  }
}

void on() {
  digitalWrite(5, LOW); //red off
  digitalWrite(4, HIGH); // green on
}

void off() {
  digitalWrite(4, LOW); // green off
  digitalWrite(5, HIGH); //red on
}

void jam(int channel) {
  radio.setChannel(channel);
    const bool result = radio.write(&payload, sizeof(payload));
    // debug(result);
    if (!result) {
        error();
    }
}

void error() {
  Serial.println("Data sending failed");
  // flash red light twice
  off();
```

```
  delay(100);
  digitalWrite(3, LOW); //red off
  delay(100);
  digitalWrite(3, HIGH); //red on
  delay(100);
  on();
  delay(50);
}

void interruptHanlder() {
  if (running == false) {
    on();
    running = true;
    Serial.println("begin");
  }
  else if (running == true) {
    Serial.println("halted");
    off();
    running = false;
  }
  delay(300);
}

void debug(int result) {
  if (result) {
        Serial.println("Data sent successfully");
    } else {
        error();
    }
}
```

Testing

Testing was very difficult initially, there is not much that can be seen physically. My best solution was to evaluate in stages. Testing the LED's, the button, power supply, and NRF24 transmitter. Most tests where straight forward just testing connections and matching actual with expected results. The NRF24 transmitter was where things became tricky as there are stages to testing it. The first stage was to test the transmitter to think it was working and then second stage was making sure it was actually working. The debug() function was made for this and the 2nd stage used various methods finally settling on Bluetooth speakers and whether they were being jammed.

Photos



Observations

When testing the 2nd stage of the transmitter, essentially deploying the product, I originally believed that there was a decent chance the product could jam WiFi waves. While theoretically that's certainly possible, but in practice that's not unless there are fringe circumstances. The main problem with this is that the device is low in power. I only discovered the device was working when I set my phone down next unintentionally and the speaker my phone was attached to across the room began to stutter and cutout. The device simply is too low in power, however adding a higher power transmitter would fix this problem.

Citations

Sources

1. Arduino Uno Rev3 Pinout Diagram: https://content.arduino.cc/assets/A000066-full-pinout.pdf
2. Mini Breadboard Diagram: https://static.javatpoint.com/blog/images/breadboard.png
3. NRF24L01 + PA/LNA Pinout Graphic: https://howtomechatronics.com/wp-content/uploads/2017/02/NRF24L01-Pinout-NRF24L01-PA-LNA-.png

References

- TMRh20,Avamander RF24: A NRF24L01+ Library [Software]. https://github.com/nRF24/RF24
- Arduino Code Documentation: https://www.arduino.cc/reference/en/
- Editorial Team - Everything RF. October 6, 2018. 2.4 GHz Wi-Fi 802.11b/g/n Channels and Frequency Band. https://www.everythingrf.com/community/2-4-ghz-wi-fi-802-11b-g-n-channels-and-frequency-band