Variable Baud Rate AM/FM Wireless Digital Communication

Rhian Chavez & Jacob Swiezy

MIT 6.101 - Spring 2019

Abstract

The goal of this project was to create a set of RF transceivers capable of simultaneously transmitting the clock and data signals necessary for digital data communication. The RF transceivers encoded data bits in the transmitted signal using a method known as binary frequency shift keying. With this method, transmitting a signal at one frequency corresponds with a high data pulse, while transmitting a signal at a slightly different frequency corresponds to a low data pulse. The RF transceivers encoded the clock signal for the data using amplitude modulation. With this method, a square wave rode atop the data signal used as the carrier frequency. The frequency of the data clock for this setup was around 1kHz which was much lower than that of the frequencies used in the frequency modulation scheme outlined above. Using both frequency and amplitude modulation for RF communication, we were able to send high fidelity communication signals between at two nodes at an arbitrary baud rate. This communication scheme could be useful for encrypting data for wireless communication or even for communicating over a mesh network where nodes only respond to data sent at a prescribed data rate.

Contents

1 Introduction

This project can be broken into two major design blocks, the transmitting and receiving circuits. Within the transmitting circuit, the following design blocks will be considered: TX controller, frequency modulation of data signal, amplitude modulation of data clock signal, and RF transmission. Within the receiving circuit, the following design blocks will be considered: RF reception and amplification, frequency demodulation of data signal, amplitude demodulation of data clock signal, and RX controller.

2 Transmitter

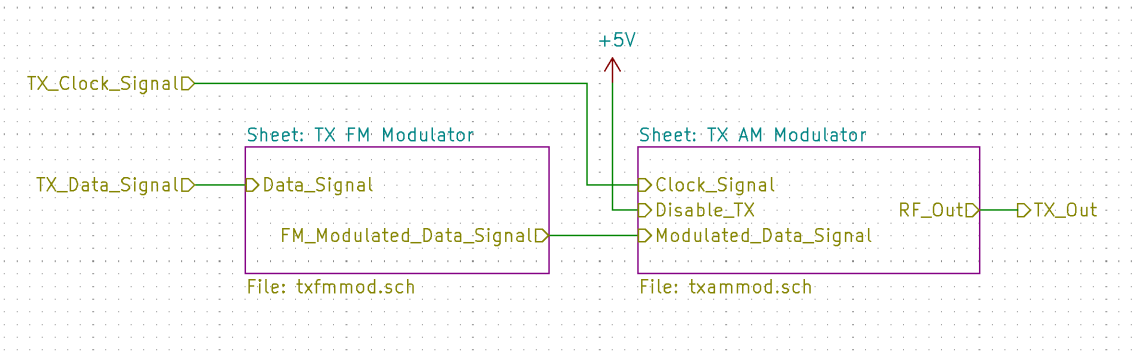

Figure 1: TX Modulation Block Diagram

2.1 TX Controller

In order to send and receive meaningful digital signals, our transmitter was controlled by a digital micro-controller capable of producing coherent signals for the receiving circuit to act on. We used the Teensy 3.2 micro-controller to perform this function. The Teensy was chosen because of its simple programming interface, availability in lab, easily accessible serial terminal, and ability to function as both the TX and RX controller for our circuit.

The software for controlling the TX Teensy is included in the appendix.

2.2 Frequency Modulation

The data signal produced by our micro-controller was transmitted using a method known as binary frequency shit keying (BFSK). A HIGH Signal corresponded to a frequency just below the resonant frequency of our transmitter, while a LOW signal corresponded to a frequency just above the resonant frequency of our transmitter. This method of modulation is demonstrated in figure 2.

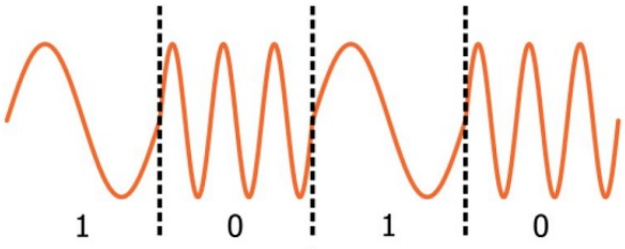

Figure 2: FM Diagram

The resonant frequency we selected for our RF communication system was 455kHz. In order to transmit distinguishable frequencies without sacrificing transmission power, we selected the following frequencies for our modulation scheme.

Table 1: Carrier Frequency Pairings

Digital Signal	Carrier Frequency	Difference from 455kHz
HIGH	440kHz	-15kHz
LOW	470kHz	+15kHz

The frequency modulation was achieved using a simple astable oscillator. The circuit for the variable frequency astable oscillator is shown in figure 3.

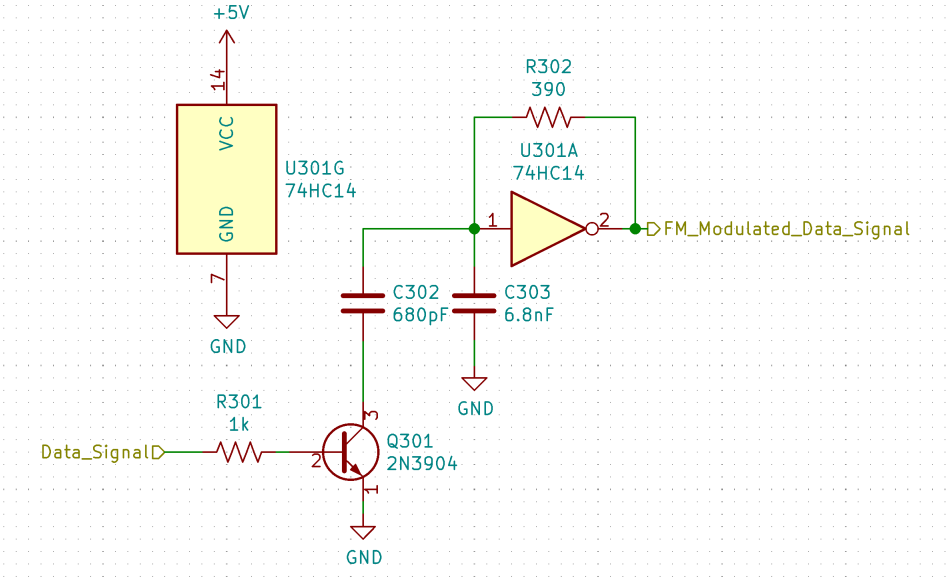

Figure 3: FM Transmitter Circuit Schematic

The astable oscillator is designed around an RC circuit connected to a schmitt trigger inverter (U_{301A}). The oscillator's frequency is a function of the values selected for the RC circuit as given below:

$$f_C = \frac{1}{0.8RC} \tag{1}$$

When a LOW signal is present on the output of the micro-controller, Q_{301} will be in a deactivated state, so only C_{303} and R_{302} will be present in the RC circuit. Thus the astable oscillator will have a frequency output of:

$$f_L = \frac{1}{0.8R_{302}C_{303}} \tag{2}$$

However, when a high signal is present on the output of the micro controller, Q_{301} will be in its active operating region, so C_{302} will be included in the RC circuit. Thus the astable oscillator will have a frequency output of:

$$f_H = \frac{1}{0.8R_{302}(C_{302}//C_{303})} \tag{3}$$

We started by selecting R_{302} to be a value of 390Ω which was in the recommended 100Ω-1kΩ range for this type of circuit. Then using the above formulae we selected C_{303} to be 6.8nF and C_{302} to be 680pF. This yielded our desired carrier frequencies listed above which were centered around the resonant frequency of our wireless circuit.

2.3 Amplitude Modulation

Amplitude modulation was used to transmit the digital clock signal produced by our microcontroller. The level of the clock signal was represented by the amplitude of the transmitted wave as seen in figure 4.

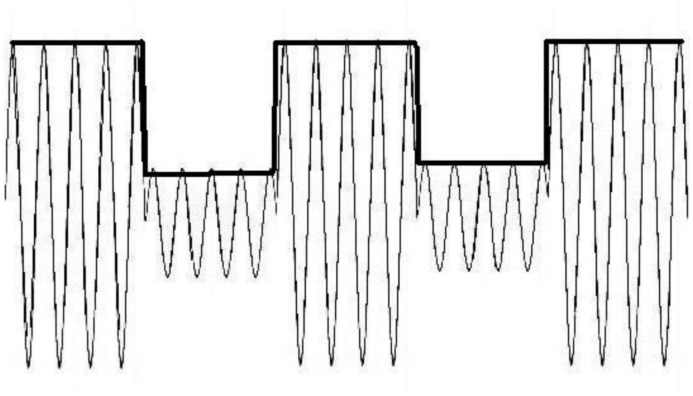

Figure 4: AM Diagram

This type of modulation was achieved using two emitter follower transistors in series as shown in the schematic in figure 5.

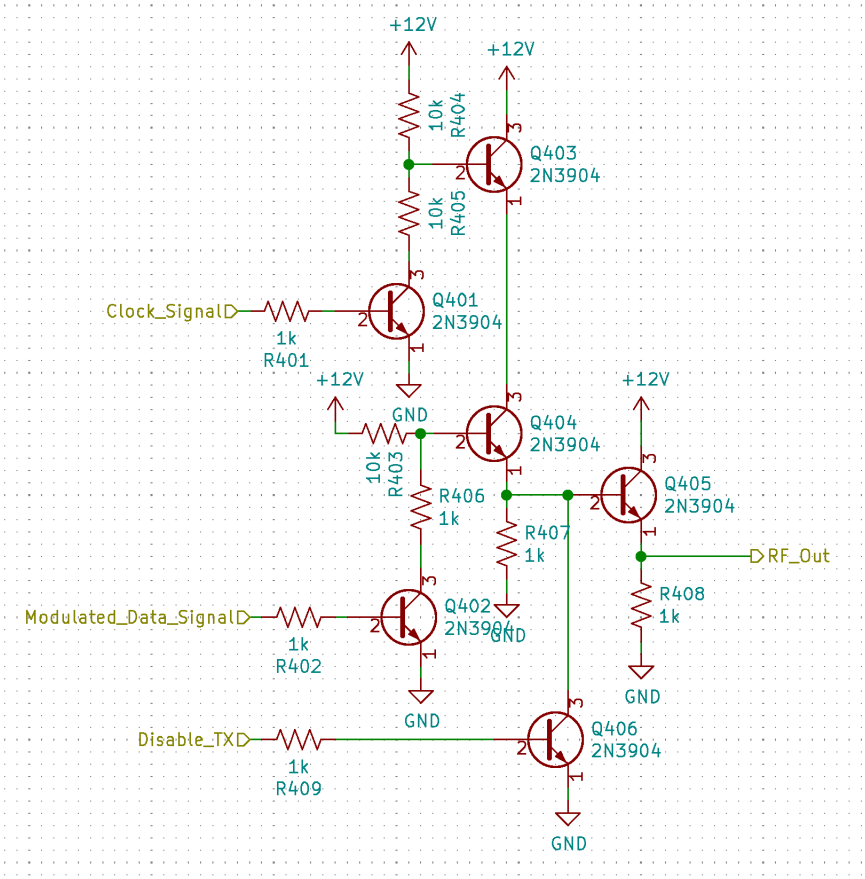

Figure 5: AM Transmitter Circuit Schematic

The bottom transistor (Q_{404}) in the circuit was controlled by the output of the frequency modula-

tion circuit. This transistor produced a square wave at the desired carrier frequency with an amplitude equal to the level of the emitter of the top transistor (Q_{403}). Because our frequency modulation circuit operated at a 5V logic level, transistor Q_{402} was added to allow Q_{404} to either be saturated or turned off by the oscillator signal. The addition of Q_{402} inverted the square wave produced by our oscillator, however, the frequency component of the wave was unaffected.

The top transistor (Q_{403}) allowed us to alter the amplitude of the wave produced by the bottom transistor (Q_{404}). This transistor was controlled by the data clock signal from the micro-controller. Because our clock signal was a digital signal meaning the signal only contained two discrete levels, we opted to only have two discrete levels for our amplitude modulation as demonstrated in figure 4. This was accomplished by biasing the base of the top transistor with two discrete voltage levels representing the highs and lows of our clock signal. A high clock signal from the Teensy would saturate Q_{401} creating a voltage divider at the base of Q_{403} with an output voltage of 4V. Meanwhile, a low clock signal from the Teensy would turn off Q_{401}, biasing the base of Q_{403} with 12V. This allowed us to create a square wave riding atop the carrier wave produced by our frequency modulation circuit. The combined output of the modulation schemes was present across R_{407}.

Q_{405} and R_{408} were configured as another emitter follower to help isolate the modulation circuits from the antenna load. Q_{406} was not ultimately used in the final implementation of the project, but it could allow the Teensy to completely disable the transmitting circuit. This would have been useful if we had implemented bidirectional communication where each node could both send and receive data.

2.4 Signal Transmission

The output of the AM modulation circuit was fed to the system's TX antenna via a 455kHz resonant LC circuit. The TX antenna was the same box antenna used in lab 1. It had an inductance of approximately 100uH. A 120pF capacitor was added in series with the antenna to achieve the desired 455kHz resonant frequency as shown in the circuit schematic in figure 6.

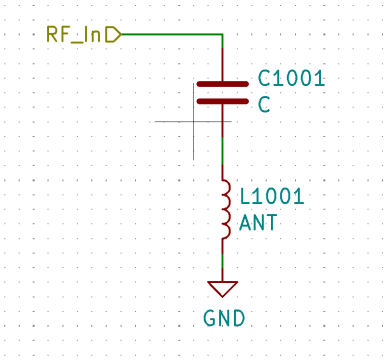

Figure 6: TX Antenna

3 Receiver

A top-level diagram of the receiving end demodulation system is presented below for clarity.

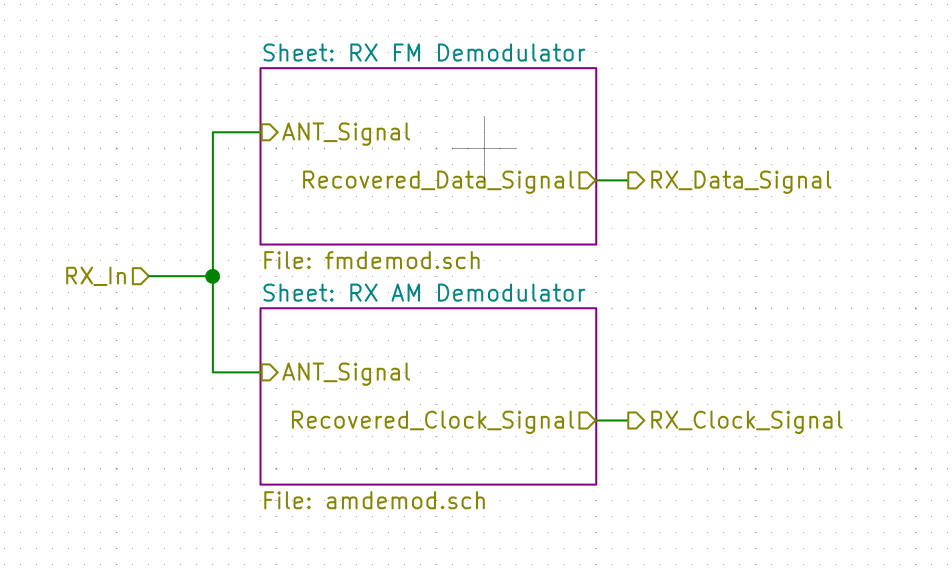

Figure 7: RX Demodulation Block Diagram

The RF signal produced by the transmitter scheme above was received by an antenna also tuned to a resonant frequency of 455kHz. The RX antenna was the same antenna/transformer combo used in lab 1. The antenna had an inductance value of approximately 720uH. A 1.7uF capacitor was placed in parallel with the antenna to achieve the desired resonant frequency. The signal was then amplified by a cascode amplifier similar to the one implemented in lab 1. The cascode amplifier acted like a current source into a resonant transformer tuned at a center frequency of 455kHz. The signal needed to be further amplified and was then passed into an emitter follower before being AC coupled into our demodulation circuitry. The entire receiving amplification and transformation circuitry is presented below in figure 8.

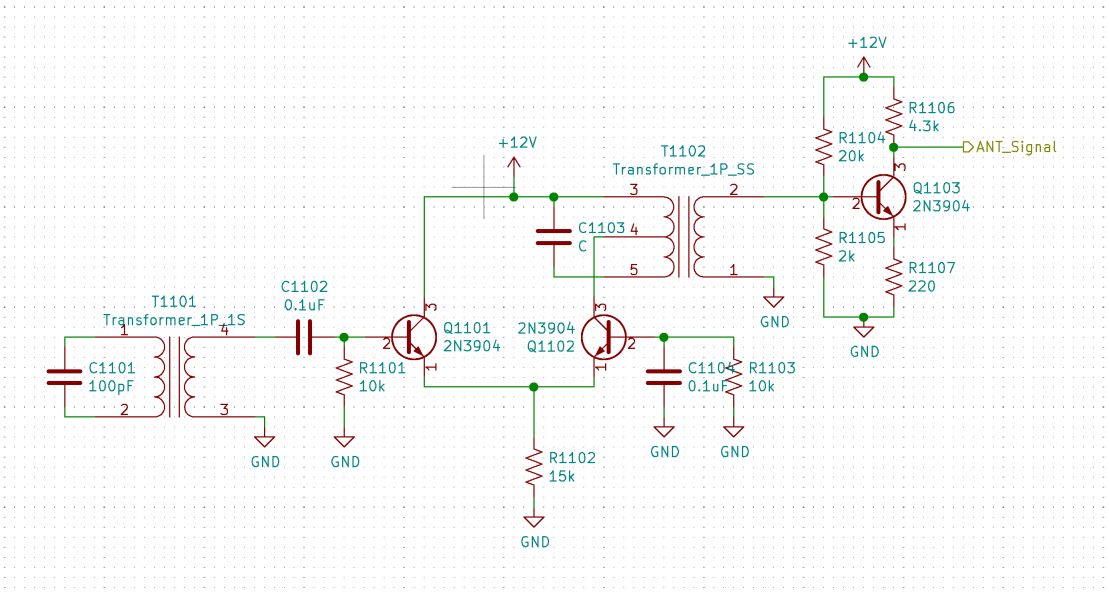

Figure 8: RX Amplifier

3.1 Frequency Demodulation

To determine the data sent by the transmitter, the carrier frequency of the received signal needed to be determined. Because the carrier frequency was one of two known values as outlined in table 1, the

carrier frequency of the received signal could be determined using a phase locked loop. The phase locked loop has many stages (frequency comparator, amplifier, and frequency generator). Typically, a phased locked loop would be used to create a constant frequency signal using negative feedback. Instead, we take advantage of the fact that within the system there is a pulse width modulation (PWM) post-amplifier where the duty cycle is correlated with the distance of the input frequency from the internal oscillator frequency. We set the internal oscillator frequency to 455kHz and can low pass filtered the PWM signal before passing the DC signal (which is proportional to the difference of the input frequency from 455kHz) to a properly tuned comparator. Using this method, we can successfully and reliably distinguish between our two digital data carrier frequencies.

We also modified the input signal to the phase locked loop by loading it onto two oppositely polarized diodes in series with a large resister. These diodes restrict the amplitude of the incoming signal and shape it into a square wave at reasonable frequencies.

The circuit diagram of the full frequency demodulation system is included below:

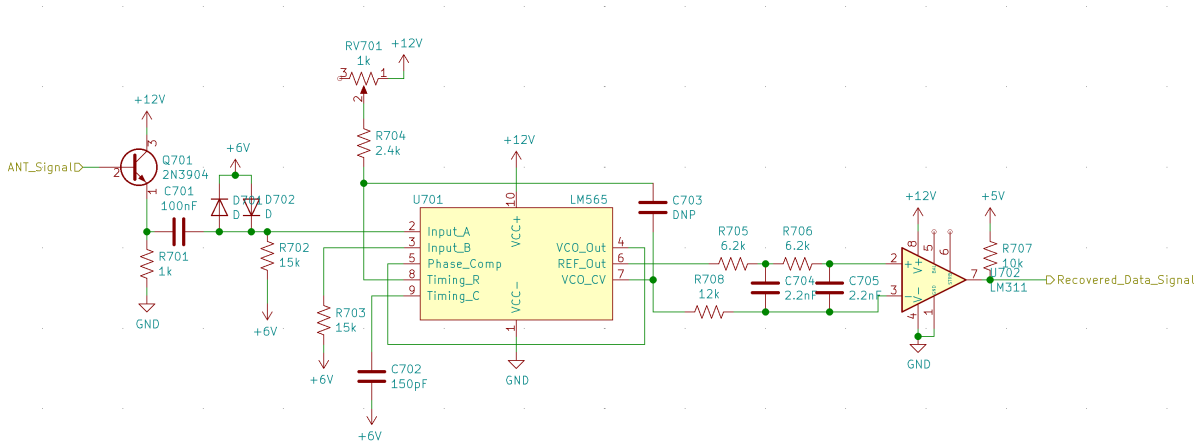

Figure 9: FM Demodulation

3.2 Amplitude Demodulation

The AM modulated clock signal was decoded using a diode and RC rectifier. The output of this simple circuit was fed into a comparator. The output of this was the square wave clock signal generated by the transmitter. The RC time constant of this circuit was set by RV_{802} and C_{801}. The time constant needed to be short enough to follow the transitions of the data clock, but long enough to filter out the high frequency carrier wave encoding the data signal. Unfortunately, this circuit was very sensitive to the transmission power. Things such as the physical proximity of the antennae greatly affected the required threshold of this circuit. Because of this, RV_{801}, which adjusted the threshold of the comparator, needed constant tuning.

A low-level circuit diagram of the amplitude demodulation circuit is presented in figure 10.

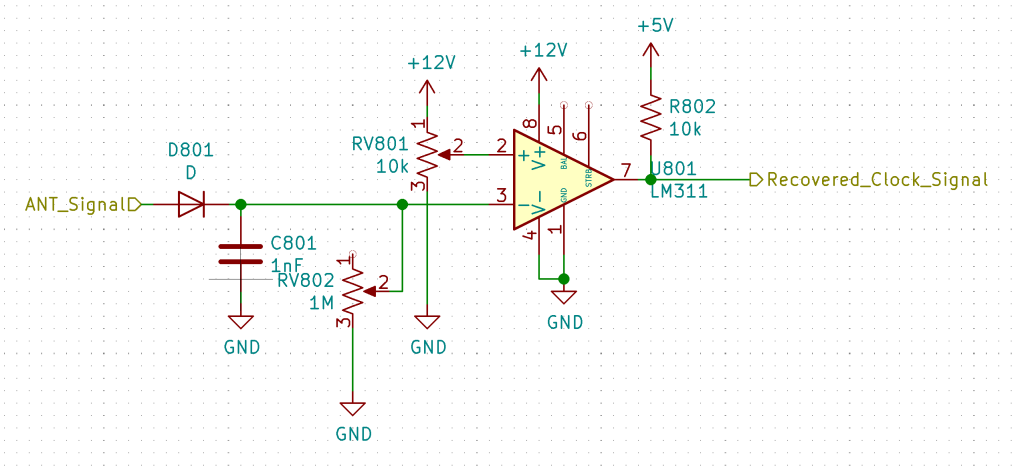

Figure 10: AM Demodulation

3.3 RX Controller

As mentioned before, we wanted to send and receive digital signals, thus the reconstructed clock and data signals produced by our receiver circuit were fed to a digital micro-controller, the Teensy 3.2, for interpretation. The Teensy decoded the messages sent and displayed the associated characters on the serial monitor. The RX Teensy was also capable of determining the exact baud rate used by the transmitting Teensy.

The software for controlling the receiving Teensy is included in the appendix.

4 Conclusion

Through this project we successfully established a means of variable baud rate wireless communication using both amplitude modulation and frequency modulation. While we achieved our objectives for the project, future iterations of this system would likely not use amplitude modulation or would implement it differently because it is very sensitive and requires precise turning. Other means of data modulation such as binary phase shift keying would likely be more appropriate. In its current form, this system could be taken advantage of for variable baud rate data encryption schemes and general close proximity wireless communication.

5 Appendix

Teensy controller code for TX and RX.

```
// pins used by transmitter
const int freqPin = 12;
const int ampPin = 13;
const int disableTXPin = 9;
// pins used by receiver
const int dataPin = 10;
const int clkPin = 11;

// baud rate range to select from
const unsigned int lowBaudrate = 1e3; //bps
const unsigned int highBaudrate = 1e4; //bps

// timeout values
const unsigned int rcvTimeout = (1e6/lowBaudrate)*2;

// function definitions
void setAmp(boolean level);
void setFreq(boolean level);

void sendChar(char value, unsigned int baudrate);

char receiveChar();
boolean waitForClkTransition();

void setup() {

  Serial.begin(9600);

  pinMode(freqPin, OUTPUT);
  pinMode(ampPin, OUTPUT);
  pinMode(disableTXPin, OUTPUT);

  setFreq(LOW);
  setAmp(LOW);
  enableTX(false);

  pinMode(dataPin, INPUT);
  pinMode(clkPin, INPUT);

  // for testing transmitter
  digitalWrite(clkPin, INPUT_PULLUP);

  Serial.println("Setup Complete!");

  // uncomment this line to get repeatable baudrates
  //randomSeed(0);

}

void loop() {
  // check if user is ready to send data
```

```
  if(Serial.available()){
    char input = Serial.read();
    switch(input){
      case '1':
        Serial.println("High Frequency Set");
        setFreq(HIGH);
        break;
      case '2':
        Serial.println("Low Frequency Set");
        setFreq(LOW);
        break;
      case '3':
        Serial.println("High Amplitude Set");
        setAmp(HIGH);
        break;
      case '4':
        Serial.println("Low Amplitude Set");
        setAmp(LOW);
        break;
      default:
        unsigned int baudrate =  random(lowBaudrate, highBaudrate);
        Serial.print("Sending: ");
        Serial.print(input);
        Serial.print(" at ");
        Serial.print(baudrate);
        Serial.println("bps");
        sendChar(input, baudrate);
        break;
    }
  // check if other transmitter is attempting to signal transmission
  }else if(!digitalRead(clkPin)){
    unsigned long transStartTime = micros();
    char data = receiveChar();
    if(data == '\0'){
      Serial.println("Attempted transmission error!");
    }else{
      // calculate approximate baudrate using length of transmission
      unsigned int transmissionBaudrate = 8*(1e6/(micros() - transStartTime));
      Serial.print("Received character: ");
      Serial.print(data);
      Serial.print(" at a baudrate of ");
      Serial.print(transmissionBaudrate);
      Serial.println("bps");
    }
  }

}

void setAmp(boolean level){
  digitalWrite(ampPin, level);
}

void setFreq(boolean level){
  digitalWrite(freqPin, !level);
}
```

```
void enableTX(boolean en){
  digitalWrite(disableTXPin, !en);
}

void sendChar(char value, unsigned int baudrate){
  // value - 8 bit character to send
  // baudrate - bits per second

  // calculate period (in usec) of data transmission from baudrate
  unsigned long period = 1e6 / baudrate;

  // signal receiver that data is coming by setting low amplitude
  // adding this extra half period will account for loss of half period at end
  // for baudrate calculation
  setAmp(LOW);
  setFreq(LOW);
  enableTX(true);
  delayMicroseconds(period/2);
  // send data serially LSB first
  for(int index = 0; index < 8; index++){
    setAmp(LOW);
    setFreq(bitRead(value, index));
    delayMicroseconds(period/2);
    setAmp(HIGH);
    delayMicroseconds(period/2);
  }
  // end of transmission
  // return to high amplitude state until new data is available to send
  setAmp(HIGH);
  setFreq(LOW);
  enableTX(false);
}

char receiveChar(){
  char data = '\0';
  for(int index = 0; index < 8; index++){
    // wait for clk transition, then write corresponding data bit
    if(waitForClkTransition()) bitWrite(data, index, digitalRead(dataPin));
    // if clk timed out, return null character
    else return '\0';
  }
  return data;
}

boolean waitForClkTransition(){
  // wait for clk line to transition from low>high
  unsigned long startTime = micros();
  while(!digitalRead(clkPin)){
      // if receive times out, return false
      if(micros()-startTime > rcvTimeout) return false;
  }
  // if receiver did not timeout, return true
  return true;
}
```