# Lab 7 – Encrypted Communication

In this lab you will be using the Wireless Tx and Rx transmitter and receiver pair to send messaged between Arduino pairs. Once you have done the tutorial on how to send and receive string messages between Arduinos, you can attempt the lab at the end of this document. This will require you to encrypt your message prior to transmitting it, and decrypt it on the other side. This way no one can read your messages!

To do this lab:

1. Do the tutorials on the Wireless Tx and Rx radio transmitter and Receiver.
2. Do the Lab on encryption and decryption

## Tutorial Wireless Tx & Rx

This tutorial is on how to drive the wireless tx & rx communication module. Technically these are two separate modules which you can drive with two Arduinos in order to perform wireless communication. For both modules you should connect the VCC to 5v on the Arduino and GND to the ground on the Arduino. For the transmitter module, you should attach the tx pin to digital pin 12, and on another Arduino you should connect the receiver’s rx pin to digital 12. (Note: if you want to connect both modules on the same Arduino, or if you want to use pin 12 for something else, you can freely choose an alternative digital pin.

### Rx, The Receiver

Here we go through the code to drive the receiver module. In this program we will setup the Rx module and continually read messages sent via the transmitter (next tutorial). Here we show the code to setup the Rx module:

//Connect Rx on digital pin 12

#include <VirtualWire.h>

void setup()

{

vw\_set\_ptt\_inverted(true); //required for receiver

vw\_set\_rx\_pin(12);

vw\_setup(4000); //set bits per second

vw\_rx\_start(); //start the receiver

Serial.begin(9600);

}

Make sure the VCC is connected to 5v on the Arduino and GND is connected to GND on the Arduino. In the code for the receiver we are including VirtualWire and in our setup function we run some code to setup the receiver. First we configure the push to talk polarity with the vw\_set\_ptt\_inverted() function. We then set digital pin 12 to be the rx pin using the vw\_set\_rx\_pin() function. Then we setup the number of bits to send per second with the vw\_setup() function. Then the vw\_rx\_start() function is used to start up the receiver. Finally we setup the Serial console I/O object.

void loop()

{

uint8\_t str[VW\_MAX\_MESSAGE\_LEN];

uint8\_t len;

if (vw\_get\_message(str, &len))

Serial.println(str);

}

In the code above, we show the Arduino loop() function. We create an 8-bit char buffer set to the maximum length the receiver can receive. We also create a length variable (len) for storing the size of the received string in. the vw\_get\_message() function returns true if there was a message received (it is non-blocking aka the function returns right away whether a message was received or not). We input the buffer (str) with a pointer to the length function, if a message is available the message is stored in str, and the length is put into len. Next we simply print out the string to the Arduino console on the desktop.

### Tx, The Transmitter

The transmitter module also requires use of the VirtualWire library. In the setup() function for the Transmitter module we again configure the push to talk polarity with the vw\_set\_ptt\_inverted() function. Then we set the tx pin to digital pin 12 and set the data transfer speed.

//Connect Tx on digital pin 12

#include <VirtualWire.h>

void setup()

{

vw\_set\_ptt\_inverted(true); //

vw\_set\_tx\_pin(12);

vw\_setup(4000);// speed of data transfer Kbps

}

In our loop function we first create a message “hello there” as a c-string. Then using the vw\_send() function we send it. This function requires a pointer to the 8-bit char data buffer (msg) and the number of bytes to send (the length of the message). Then we wait for the message to be sent using the vw\_wait\_tx() function before finally calling delay() to wait for 2000 milliseconds (2 seconds) before repeating the transmission again (via the Arduino loop function being called repeatedly).

void loop()

{

char msg = "hello there";

vw\_send((uint8\_t\*)msg, strlen(msg));

vw\_wait\_tx(); // wait until the message is sent

delay(2000);

}

## Lab 7: Encrypted Communication

In this lab you will be required to create your own encryption algorithm to send secret messages between two Arduinos. To complete this lab you should complete the Wireless communication module tutorial. Here you must use the modules to transmit a secret message to your partner (or your other Arduino). You must first perform two types of encryption on this message prior to sending. One is the Caesar Cipher and the other is the shift cipher. In the Caesar cipher, you must shift the each character along the alphabet by a certain amount and substitute each character for the corresponding one in the shift. For example, apple shifted along by 2 would be output as: crrng. The problem with this is the secret message may still be found out due to statistical analysis. For example, there may only be a subset of words which have double letters (pp in apple = rr in crrng) and hackers may exploit these statistics with other secret messages sent. Therefore it is a good idea to implement a shift cipher along with the Caesar (substitution cipher). For example, we may shift the letters by a recurring list of shift values, for example 0,1,2,0,1,2... may be used to shift "apple start program" to "aqrlf utbtt qtohtan" where a in apple was shifted 0 times to a, the second p in apple was shifted once to q and the third apple was shifted twice to r and so on and so forth.

In this lab you must implement the shift-substitution cipher, use pointers and a function enc() to encrypt a message. This function must take an input string along with an input array where each element in the array is a shift value (the shift pattern in the array is repeated eg. 0,1,2;0,1,2;0,1,2...). You must also write a function called dec() to decrypt a string. These functions must have the following signatures:

void enc(char \* input, char \* output, int \* shiftPattern, int shiftPatternLength);

void dec(char \* input, char \* output, int \* shiftPattern, int shiftPatternLength);

Implement these functions and use them with your friends to send secret messages with eachother. Make sure to rendezvous with oneanother to exchange shift patterns. Note: this type of cipher may still be cracked. But even today, the best encryption methods are based on substitution and pattern shifting.