

Basic Electronic Attack

This multi-week lab will explore the fundamental concepts to perform a basic electronic attack. There are 3 parts to this lab, with required deliverables listed for each section.

- Format for submission should follow the IEEE format (Found at: <https://www.ieee.org/conferences/publishing/templates.html>).
- **Any code should be submitted in Appendix.**

- PART 1 -

Determining Carrier Frequency and Modulation

Set-up.

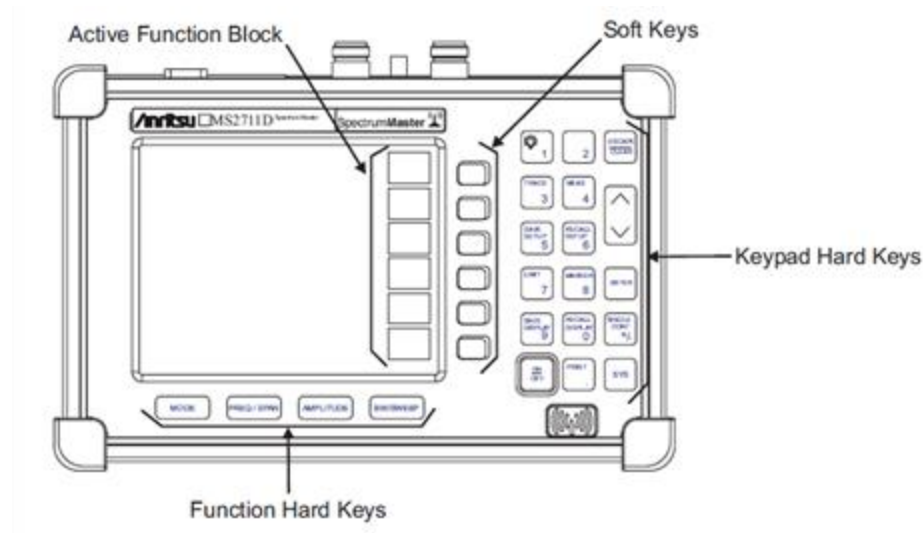
Equipment required:

- ☐ Anritsu MS2711D Spectrum Analyzer
- ☐ Agilent InfiniiVision MSO7032A Oscilloscope
- ☐ Telescoping Antenna w/ BNC connector
- ☐ RC Vehicles



Determine Carrier Frequency

To determine the carrier frequency of the provided RC vehicles, use the Anritsu MS2711D Spectrum Analyzer. To set up and operate the spectrum analyzer, refer to the “**Spectrum Analyzer Setup Instructions**” at your lab station.



Record your findings on Table 1 (see next page). **Note:** This table will serve as a Part I deliverable, so print it out and be neat!

SY310 Lab II, Table 1

RC Vehicle Description	f_c	Time-Domain Sketch	Modulation Type

O-Scope Time-domain Demo: _____

Time domain & Modulation

For a better idea of modulation scheme employed by each type of RC vehicle, we'll use the Agilent Technologies "InfiniiVision" MSO7032A 350 MHz Oscilloscope to look at transmitted signal in the time domain. For instructions on setting up and operating the O-Scope, refer to the "**Oscilloscope Setup Instructions**" posted at your lab station.

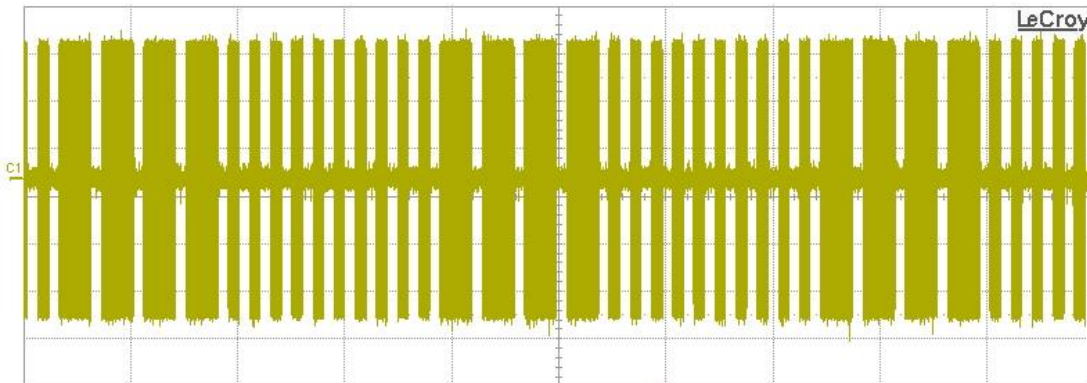
Once have the Channel configured, it's time to *capture the signal* using the Trigger section of the O-Scope.

- ☐ Press the **Edge** button
- ☐ Set Source = 1



- ☐ Ensure the antennas are extended on the O-Scope and the Remote if applicable
- ☐ Holding the RC car transmitter close to the O-Scope, send the Forward signal by driving the car forward.
- ☐ Capture the signal by pressing the Run/Stop button

If done correctly, your O-scope display should look similar* to this:



* Captured signal may vary – that's ok!

- ☐ Adjust the time scale to "expand" the captured waveform. What do you see? **Demonstrate to your instructor!**
- ☐ Sketch each vehicle's signal in the time-domain in the 3rd column of Table 1. Indicate the type of modulation each vehicle uses in column 4.

Deliverable for Part 1

1. Brief introduction and discussion of your procedure and findings regarding your RC vehicle control signals in time and frequency.
2. *Neatly* complete all parts of Table 1 and obtain instructor initials.

- PART 2 -
Control Signal Design Discovery

Set-up.

Equipment required:

- ☐ Agilent InfiniiVision MSO7032A Oscilloscope
- ☐ Telescoping Antenna w/ BNC connector
- ☐ RC Vehicles

Based on your findings in Part 1, some of these vehicles should be easier to attack than others. In the interest of time, we're going to choose the "easy" road for Part 2. For our purposes, "easy" means we can readily identify the carrier frequency and the control signals that cause the vehicle to drive in a particular direction.

STOP! Determine which RC vehicle will be "easiest" to attack.
You must get approval from your instructor before continuing!

Think about the controls – how many different signals do you expect to control the car? We looked at the forward signal in Part 1; how does the transmitted signal change for reverse? In addition to driving forward, the car can operate in reverse, as well as turning left and right... and any combination thereof! There are actually 9 different combinations of signals (including a release/stop signal), and to control the car you need to identify which operation each transmitted signal represents!

Examine each transmitted signal by using the O-Scope and repeating the process from Part 1 to capture the signal:

- ☐ On "Trigger" section of O-Scope display, select "Normal"
- ☐ Transmit desired signal.
 - Forward
 - Reverse
 - Forward AND Right (This is different from the signal to pivot the wheels to the right only!)
 - Forward AND Left (This is different from the signal to pivot the wheels to the left only!)
 - Right
 - Left
 - Reverse AND Right
 - Reverse AND Left
 - Release
- ☐ When your signal is displayed on the screen, press "Stop" on Trigger menu.
- ☐ Adjust time scale as required.

- PART 3 -

Design the Electronic Attack

Set-up.

Equipment required:

- ☐ Your issued Laptop
- ☐ MATLAB Code `RCcode.m` and `getkey.m`
- ☐ RC Vehicle
- ☐ Signal Generator & accessories (Instructor will set up)
- ☐ Anritsu MS2711D Spectrum Analyzer

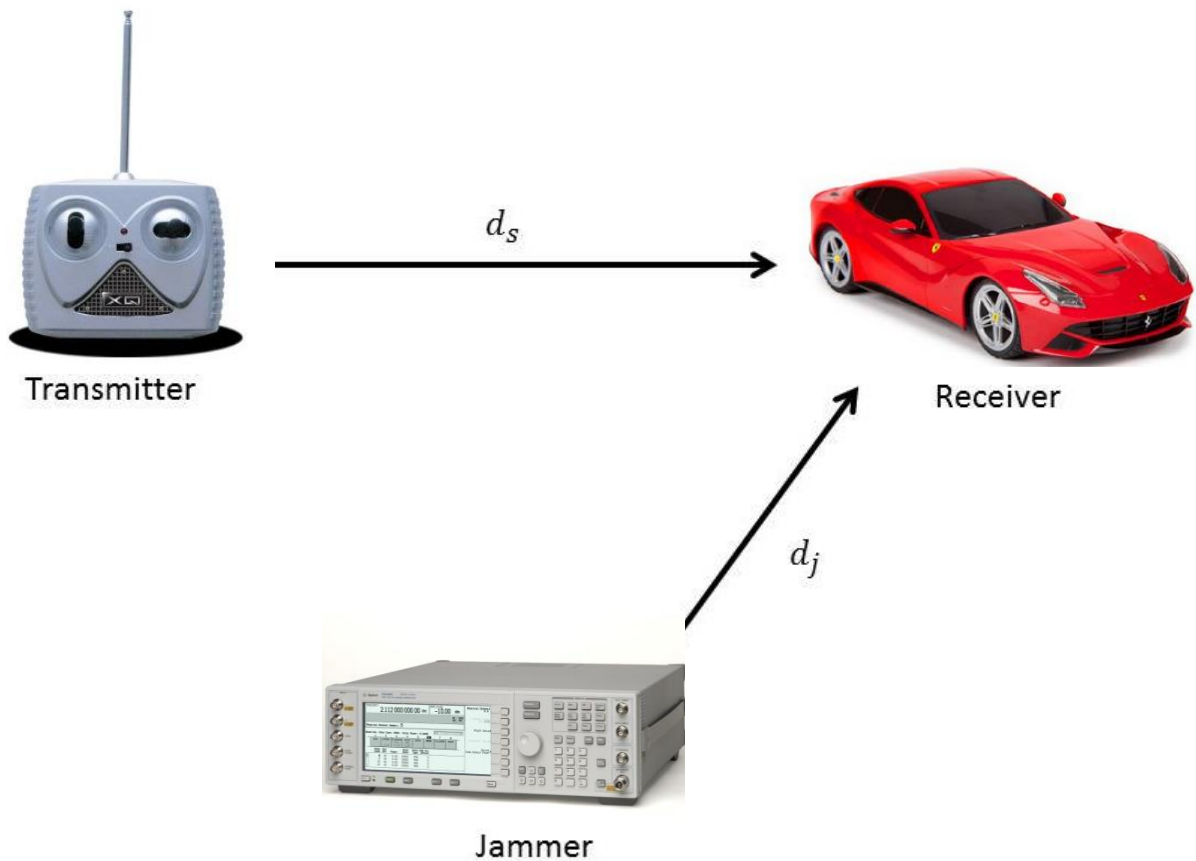


Now that we have some basic intel, think about what could happen if another signal was transmitted at the carrier frequency... The answer: It depends! We'll explore two scenarios.

Scenario 1: "Dumb" Jamming

In lecture, we learned that the effectiveness of EA/Jamming is dependent upon the Jamming to Signal Ratio (JSR). The JSR is dependent upon both the power of the jammer and the transmitter as well as the distance of the jammer and the transmitter from the receiver.

In this lab, our scenario looks like this:



The JSR depends on the *received* signal power at the car and the *received* jamming power at the car:

$$JSR = \frac{J}{S} = \frac{P_{J(W)}}{P_{S(W)}} = P_{J(dB)} - P_{S(dB)}$$

Generally, if the JSR is greater than 1 (or 0 dB), jamming will be effective.

- ☐ Play time! Drive your vehicle around the classroom. Your instructor will generate a 20 dBm FM signal at the carrier frequency. While your instructor is transmitting the jamming signal, experiment! Attempt to control the RC car with its transmitter at different distances from both the jammer and the RC car.
- ☐ Use the Anritsu MS2711D Spectrum Analyzer to draw the jamming signal in the frequency spectrum. How does this change if you transmit while standing next to the Spectrum Analyzer?
- ☐ How could you increase the range of the jammer?

Scenario 2: Reverse Engineering “Hack”

Let’s take a step up from “dumb” jamming. We know the carrier frequency. We know the signals that control the RC vehicle. We know some basic MATLAB. Let’s design a system to “take over” the RC car.

Examine the MATLAB `RCcode.m` file. The MATLAB code takes input from the arrow keys on your laptop and generates the baseband binary signals to control the RC vehicle. For simplicity’s sake, we’ll only use generate waveforms for 4 of the 9 possible operations, which means we’ll be slightly limited in the operation of our RC vehicle – we won’t be able to turn while operating in reverse.

- ☐ In MATLAB, update the “Setup Major Variables” section of your `RCcode.m` code (shown below) with the number of 1s in the “trail” (as recorded in Part 2 on Table 2) in preparation of taking over the RC vehicle.

```
#####  
% RC CAR CODE %  
#####  
  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%  
%   PRESS SPACE TO TERMINATE EXECUTION   %  
%  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
%  
% !!!!! NOTE !!!!!  
% If you do something wrong and Matlab terminates unexpectedly (you get a  
% lot of angry red Error messages) you will have to close out and restart  
% Matlab in order to clear out the sound card buffer!!!  
%  
% Forward = Up Arrow  
% Reverse = Down Arrow  
% Forward Right = Right Arrow  
% Forward Left = Left Arrow  
%  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
% Clear out memory and initialize default settings %  
% DO NOT CHANGE THIS SECTION %  
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%  
clear all
```

Change
This
Section!

```

close all
set(0, 'DefaultAxesFontSize', 14)
set(0, 'DefaultAxesFontWeight', 'Bold')
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Setup major variables
% CHANGE THIS SECTION ONLY!!! (FOLLOW LAB INSTRUCTIONS) %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
forward_ls = 01;
reverse_ls = 01;
right_fwd_ls = 01;
left_fwd_ls = 01;
} Insert Number of 1's from Table 2 here!
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sam_per_sym = 22; %fs/Rb = 44.1e3/(1/Tb), Tb ~ 500us
fs = 44.1e3; % Set sampling rate to sound card rate
Rb = fs./sam_per_sym;
fif = 10e3; % 10.0 kHz "baseband" (IF) Frequency
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Generate the original data to manipulate the car
% DO NOT CHANGE THIS SECTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
sync = [1 1 1 0 1 1 1 0 1 1 1 0 1 1 0];
forward = [sync repmat([1 0], 1, forward_ls)];
reverse = [sync repmat([1 0], 1, reverse_ls)];
right_fwd = [sync repmat([1 0], 1, right_fwd_ls)];
left_fwd = [sync repmat([1 0], 1, left_fwd_ls)];
pause = zeros(1,500);
key = 0; % Initial Keyboard Value

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Reads inputs once per second
% DO NOT CHANGE THIS SECTION
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
while key ~= 32 %Press space to stop

    key = getkey(1);


    if key == 30
        data = [forward forward forward forward forward forward forward forward];
    elseif key == 31
        data = [reverse reverse reverse reverse reverse reverse reverse reverse];
    elseif key == 29
        data = [right_fwd right_fwd right_fwd right_fwd right_fwd right_fwd right_fwd right_fwd];
    elseif key == 28
        data = [left_fwd left_fwd left_fwd left_fwd left_fwd left_fwd left_fwd left_fwd];
    else
        data = [pause];
    end

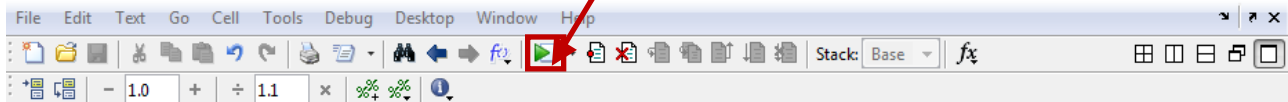
    % Generate Polar NRZ
    time_stop = length(data).*sam_per_sym;
    up_data = zeros(1,time_stop);
    time = linspace(0,(1/fs).*time_stop, length(up_data));

    % Upsample
    for i = 0:length(data)-1
        up_data(sam_per_sym.*i + 1 : sam_per_sym.*i + sam_per_sym) = data(i+1);
    end

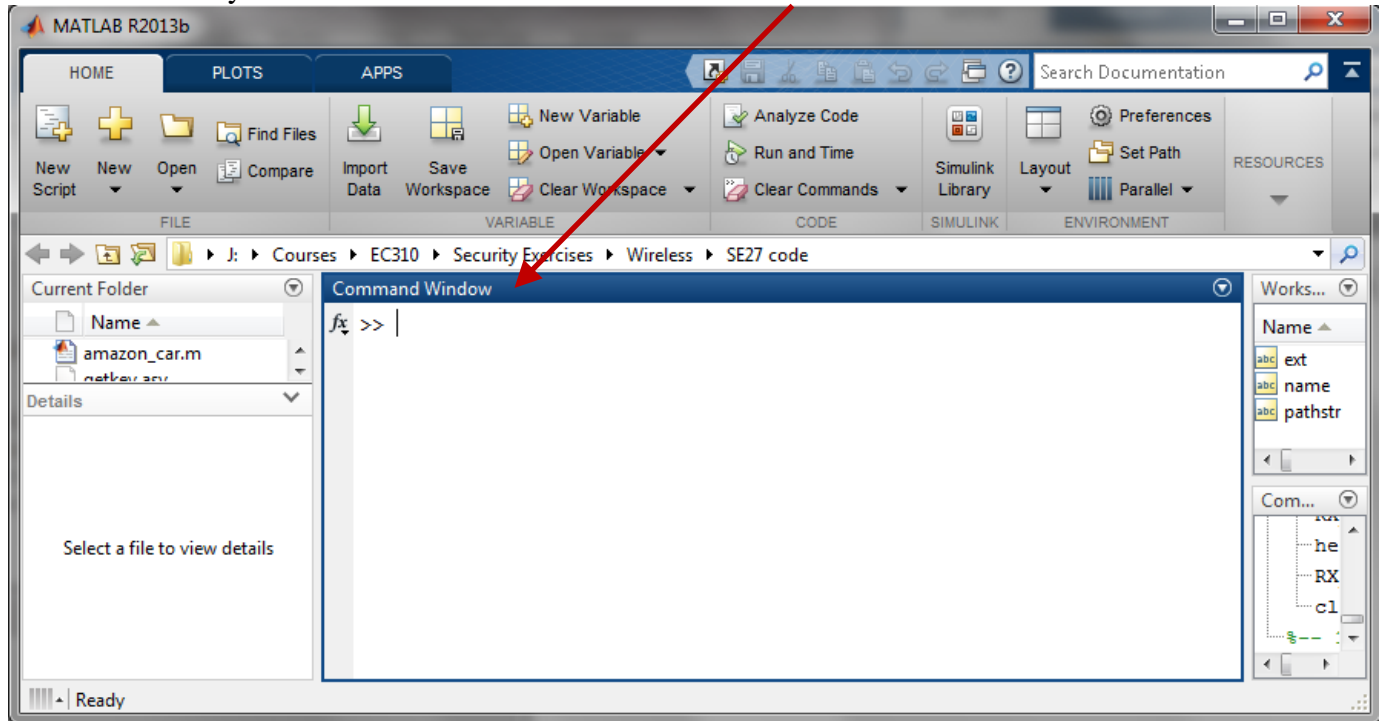
    % Generate the "baseband" (IF) waveform
    s_lo = cos(2.*pi.*fif.*time);
    s_if = s_lo.*up_data;
    soundsc(s_if,fs)
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

```


- ☐ When your code is updated, run it by pressing  . *Follow the next instruction carefully!*



- ☐ Double click your cursor in the MATLAB Command Window.



- ☐ Press your arrow keys to *simulate* driving your vehicle. What do you hear? What type of signal is being generated?

STOP! What can you do to transmit this signal so that the car can receive it?
You must get approval from your instructor before continuing!

- ☐ Design a system to “drive” the RC car from your laptop. (Not sure where to start? Try working through the daily beginning-of-class review!)
- ☐ Get your design approved by your instructor and prepare to drive!

Deliverables for Part 3

1. Summary of “dumb jamming” scenario, to include:
 - a. A sketch of the RC transmitter and jamming signals in the frequency domain
 - b. The range of the classroom jammer and brief discussion of how to increase the range.
2. Discussion and diagram of your system design to “drive” the RC car from your laptop.