USAF ACADEMY
DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING
LESSON 11 LAB - CUSTOM BLOCK
Lab Packet
Spring 2027

Name: _____ **Section:** _____ **Date:** _____

Learning Objective 5.1: I can create a GNU Radio out-of-tree (OOT) module and add a custom Python block to it.
Learning Objective 5.2: I can implement a custom signal-processing block from scratch and integrate it into a GNU Radio flowgraph.

This packet is the L11 Lab, in three parts, and you turn it in once, complete, before the midterm. Do **Part 1** on your own before the lab period: check your laptop's tools and work out the Morse timing you will check at the bench. **Part 2** is your own work in class: build and test the course module, set the Energy Detector's levels, and then write a block of your own, the **Key Timer**, in a module of your own. Do **Part 3** on your own after the lab.

Grading. One submission, graded against Objectives 5.1 and 5.2 on the course's 1 / 0 scale, with revision allowed under course policy. Paste screenshots and code where a question asks for them.

Nothing in this lab transmits: every flowgraph is a simulation on your laptop.

The files are on the L11 lesson page: `L11-gr-ece448.zip`, `L11-keyed-carrier.grc`, and `L11-keyed-carrier-epy.grc`. Every number below assumes the flowgraph's settings: `samp_rate` = 32 kHz, a tone of amplitude 1, and Morse "ECE" repeating. One Morse unit lasts $1.2/\text{WPM}$ seconds: a dit is 1 unit, a dah 3, the gap inside a letter 1, the gap between letters 3, and the gap after a word 7.

Routes. On Linux with `gnuradio-build-deps` installed, you build real modules; macOS should work the same way, but the course has not tested it. On Windows, first install WSL2 with Ubuntu 24.04 and follow the lab page's Ubuntu steps inside it. Without WSL2, take the Embedded Python Block route the lab page describes, and as a last resort build the two modules on a teammate's laptop for Questions 3(a) and 5.

Part 1: Pre-Lab (individual, before the lab period)

1. **Your tools.** At the radioconda prompt, run `gnuradio-config-info --version`, `gr_modtool --help`, and `cmake --version`.

   (a) Record your operating system, your GNU Radio version, and whether `gr_modtool` and `cmake` run. Name the route you will take.

   (b) Open and run `L11-keyed-carrier-epy.grc`, which needs nothing installed. Paste a screenshot of its window at the default settings.

2. Timing. You will check these at the bench.

(a) At 15 words per minute, find the length of a dit, a dah, the gap inside a letter, the gap between letters, and the gap after the word, and the number of samples in one unit at 32 kHz.

Dit	Dah	Inside a letter	Between letters	After the word	Samples per unit

(b) One sending of "ECE" is 26 units long, with its final word gap. How long does it last at 15 WPM, and how many points is that at the detector's 1 kHz output rate? You set the Time Sinks' Number of Points to this at the bench.

(c) Averaging 32 samples, how many detector outputs is a dit? Averaging 128, how long does one output last, and how many outputs is a dit? How many marks (key-downs) does one sending of "ECE" hold, and how many arrive in 30 s?

$N = 32$:

$N = 128$:

marks:

Part 2: At the Bench (in class)

3. **Build and test gr-ece448.** Follow the lab page's steps.

   (a) Paste the output of `ctest`, and the `gr_python_install` line of `python/ece448/CMakeLists.txt`.

   (b) Open `L11-keyed-carrier.grc`, which uses the installed block, and run it. Paste a screenshot of the Energy Detector's properties dialog.

4. **Set the levels.** In the running flowgraph, set the noise amplitude to 0.5. Then stop it, set `wpm` to 15 and both Time Sinks' Number of Points to one sending of ECE at 1 kHz, from your Question 2(b) answer, and run it again.

   (a) Read the floor and the carrier-on level from the power plot, and describe how much the floor jumps up and down.

   off: _____

   on: _____

   (b) Set the threshold to the floor's level with no hysteresis, and describe the key. Then choose your own threshold and hysteresis so that the key follows the keying for at least 20 s. Record your settings and paste a screenshot.

5. **Write the Key Timer.** Make a module of your own and write a block in it from the template, as the lab page lists. The block is a sink with one float input, the Energy Detector's key, and one parameter, `rate`, the key's sample rate. It counts how many samples each mark and each space lasts and keeps the length of the last mark and the last space, in ms, and the number of marks so far. Three methods, `last_mark()`, `last_space()`, and `mark_count()`, return them.

   (a) Paste your `key_timer.py`.

   (b) Paste your `qa_key_timer.py` test and the output of `ctest`. The test must feed a known key, such as five 0s, three 1s, two 0s, one 1, and four 0s at a rate of 1000, and check all three methods.

   (c) Paste your `grc/*.block.yml`, and the `gr_python_install` line, which must list `key_timer.py`.

6. Time the Morse. Save a copy of `L11-keyed-carrier.grc` with your settings from Question 4, add your Key Timer on the Energy Detector's key output with `rate` = `samp_rate / decim`, and show its three values on QT GUI Labels through Function Probes.

(a) Watch the labels for a few sendings and record the mark and space lengths you see. Compare them with Question 2(a).

	Dit	Dah	Inside a letter	Between letters	After the word
Measured (ms)					

(b) Paste a screenshot of your flowgraph's canvas and of its running window.

7. A noisier signal. Set the noise amplitude to 0.8.

(a) With 32 samples averaged, choose the best threshold and hysteresis you can find. Record them, then restart the flowgraph and record the mark count after 30 s.

(b) Stop the flowgraph, set `decim` to 128, and repeat. Describe how the floor on the power plot changed. Record your settings, the count after 30 s, and the dit length the Key Timer reports.

Part 3: Writeup (individual, after the lab)

8. **Hysteresis and timing.** At a noise amplitude of 0.5, with a -2.5 dB threshold, and 3 dB of hysteresis, some dits read 81 or 82 ms; with no hysteresis, they read 80 ms. Explain why, and what Question 7 says about choosing N.

9. **Module or Embedded Python Block?** Lesson 12 builds a Morse decoder that uses the Energy Detector and your Key Timer. Explain which of the two should live in an OOT module and why, and what the unit test caught or would catch that watching the flowgraph would not.

10. **From timing to letters.** The Key Timer reports the dit and dah lengths you measured in Question 6(a). Propose a rule a decoder can use to tell a dit from a dah, and a rule to tell the end of a letter from the gap inside one, that still works at 20 WPM.

Documentation: _____