✅ 1-Month Embedded & Systems Programming Kickstart Roadmap

Time commitment: 2–3 hours/day
Outcome: You'll be able to handle entry-level firmware/system roles, build small embedded projects, write multithreaded C programs, use IPC, sockets, RTOS, and makefiles.

📅 WEEK 1 — Embedded C + MCU Architecture + Linux Basics

🎯 *Goal:* Transition from C/C++ to Embedded C mindset & understand microcontrollers + toolchains.

DAY 1–2 — Embedded C Fundamentals

Learn what makes Embedded C different:

- Bitwise operations (critical)
- Volatile keyword
- Memory-mapped I/O
- Interrupt Service Routines
- Endianness
- Structs, unions & packed structs
- Fixed-width types (stdint.h)
- Bare-metal programming flow

Mini-tasks

- Implement blinking LED logic *without actual hardware* using register simulation in C.
- Write code showing why `volatile` matters with infinite loops or flags.

DAY 3–4 — Microcontroller Basics

Learn the overall MCU workflow:

- GPIO, Timers, UART, I2C, SPI
- Clock system, PLL
- Interrupt controller
- Memory layout (Flash, SRAM, Registers)
- Linker scripts (only conceptual for now)

Mini-project

- Pick an MCU family (STM32 or ESP32 recommended).
- Install tools:
  - STM32CubeIDE **or** ESP-IDF
  - OpenOCD
  - arm-none-eabi-gcc

DAY 5–7 — Linux OS + Build System Basics

Learn the environment where firmware/systems code runs before flashing.

Topics:

- Linux command line mastery
- File permissions, processes, signals
- Makefile basics
- GCC toolchain usage
- Cross-compiling basics
- GDB essentials (setting breakpoints, stepping, watchpoints)

Mini-projects

1. Write a Makefile to compile a simple C program with multiple `.c` and `.h` files.
2. Use GDB to debug a segfault program.

📅 WEEK 2 — Multithreading, Deadlocks, IPC, Synchronization

🎯 *Goal:* Become comfortable with concurrent programming on Linux.

DAY 8–10 — POSIX Threads (pthreads)

Learn:

- Creating threads
- Mutex, semaphores
- Condition variables
- Thread-safe programming
- Race conditions
- Deadlock detection & prevention

Mini-project

- Implement producer–consumer with mutex + condition variables.
- Write a program that intentionally deadlocks, then fix it.

DAY 11–12 — IPC Mechanisms

Learn:

- Pipes
- FIFOs
- Shared memory
- Message queues
- Signals
- mmap

Mini-project

- Build a parent–child communication system using a pipe AND shared memory.

DAY 13–14 — Multicore & Synchronization

Learn:

- Cache coherence basics
- Atomic operations
- Memory ordering
- Thread affinity (sched_setaffinity)

Mini-project

- Bind two threads to different cores and measure execution differences.

📅 WEEK 3 — Networking, Socket Programming, Protocols (TCP/UDP, RS232, I2C, SPI)

🎯 *Goal:* Build competence with networking + low-level communication protocols.

DAY 15–17 — Socket Programming

Learn:

- TCP vs UDP
- Creating server and client
- Blocking vs non-blocking sockets
- Select/poll/epoll basics

Mini-projects

1. A TCP chat program (client + server).
2. A UDP broadcaster + listener.

DAY 18 — Common Protocols (RS232, USB, I2C, SPI)

You don't need deep electrical knowledge initially. Learn:

- How UART works
- I2C addressing, SDA/SCL
- SPI master/slave, CPOL/CPHA
- USB layering
- WiFi stack (basic understanding)

Mini-project

- Simulate I2C/SPI transactions in C (struct-based, not hardware).

DAY 19–21 — Embedded Linux & RTOS Basics

Learn:

- Processes vs threads in Linux
- Real-time constraints
- RTOS concepts:
  - Tasks, queues, semaphores
  - Task scheduling
  - Priority inheritance
  - Timer tasks

Mini-project

- Run FreeRTOS on STM32 or ESP32.
- Implement two tasks + queue communication.

📅 WEEK 4 — Build Systems, Agile/SCRUM, Debugging, Integration Project

🎯 ***Goal:*** **Be interview-ready with an integrated project covering *all skills*.**

DAY 22–24 — Build Systems & Linux Environment

Learn:

- Writing advanced Makefiles
- Build rules, phony targets, environment variables
- Using CMake (optional)
- Using gcc flags (-O, -Wall, -Werror, -g)

Mini-project

- Convert a raw C project into one built via Makefile.
- Add debug and release build modes.

DAY 25–26 — Debugging Tools: GDB + trace32 Theory

You don't need full trace32 mastery — understand the concepts.

Focus on:

- Breakpoints, watchpoints
- Stack examination
- Backtraces
- Core dump debugging
- JTAG/SWD basics

Mini-project

- Debug a deadlocking multithread program.
- Debug a segmentation fault with gdb.

DAY 27–28 — Agile/SCRUM Basics

Learn:

- Sprint

- Standups
- User stories
- Backlogs
- Story points
- Definition of Done

Mini-project

- Create a mini sprint board for your Week 4 integrated project.

DAY 29–30 — Final Integration Project

Build a **complete embedded-system simulation project**:

Project: Multi-threaded Sensor Node Simulator

Includes:

- 3 threads (sensor, processor, logger)
- IPC via POSIX message queue
- TCP server broadcasting sensor data
- Makefile build
- Debug session using gdb
- Optional: RTOS version on ESP32/STM32

This integrates:
✓ Embedded C mindset
✓ Multithreading
✓ IPC
✓ Socket programming
✓ Build system
✓ Debugging
✓ Agile planning

Perfect for interview discussions.
