Electrical & Computer ENGINEERING

Carnegie Mellon

Projects In Brief

18-349/14-642: *Introduction to Embedded Systems*
Fall 2019

Lab 0 – PCB Design and Manufacturing

Students design and build a printed circuit board. The PCB is an expansion shield that interfaces an STM32 Nucleo board to various sensors and peripherals. Peripherals include an analog microphone, light sensors, 7-segment display, servo controller port, motor encoder and an H-Bridge circuit for driving motor outputs. The PCB also has a connection to a Raspberry Pi which will be used to communicate between the STM32 and the RPi. This allows students to explore big-little architectures that run a hybrid of Linux (on the RPi) and real-time firmware on the STM32.

Lab 1 – Assembly and Bootloaders

Students implement a bootloader for the STM32 along with completing a set of assembly optimization challenges. Students are introduced to assembly programming and the toolchain used in the rest of the lab set.

Lab 2 – Peripherals

Students use Memory Mapped I/O (MMIO) to build peripheral drivers. Students implement a UART driver, an I2C driver, and build an acoustic "clap" detector that runs on the STM32. The UART driver allows the STM32 board to print to the user's console on their PC. The implementation of I2C also allows the use of the 7-segment display. The clap detector along with the onboard ADC take data from the microphone and the light sensor. The value of both sensors is shown on the 7-segment display. When a clap is detected, a message is printed to the screen.

Continued on next page

Lab 3 – Timers and Interrupts

Students build a syscall that controls a servo using pulse width modulation. This lab also introduces a user mode, which enables student implement syscalls, which are special interrupts, to access protected resources.

Lab 4 – Real-Time Kernel

Students architect and building their own multi-threaded Real Time Operating System (RTOS). They implement context switching, mutexes, and enforced fixed priority scheduling.

Lab 5 – Interfacing to Linux

Students design and implement a PID controller to control the position of a wheel attached to a motor. A two-tier design with bottom half of a controller running on the RTOS built in lab 4 and the top-half of the controller running on an RPI is required. The RPI interfaces to the microcontroller using a Linux kernel module that communicates over SPI.