Electrical & Computer ENGINEERING

Carnegie Mellon

Projects In Brief

18-349/14-642: *Introduction to Embedded Systems*
Spring 2020

Lab0 – PCB Design and Manufacturing

In this lab, students will 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.

Lab1 – Assembly and Bootloaders

This lab will introduce students to assembly programming and the toolchain used in the class. Students will implement a bootloader for the STM32 along with a set of assembly optimization challenges.

Lab2 - Peripherals

Lab 2 will teach students how to use Memory Mapped I/O (MMIO) to build peripheral drivers. Students will implement a UART driver, an I2C driver and build an acoustic "clap" detector that runs on the STM32. The UART driver will allow for the STM32 board to print to the user's console on their PC. The implementation of I2C will allow them to use the 7-segment display found on their custom PCB shield. The clap detector will the onboard ADC, to take data from the microphone and the light sensor. The value of both sensors will be shown on the 7-segment display. When a clap is detected, a message will be printed to the screen.

Lab3 – Timers and Interrupts

This lab explores timers and interrupts. Students will build a syscall that can control a servo using pulse width modulation. This lab will also introduce a user mode, which means student will need to implement syscalls, which are special interrupts, to access kernel level tasks.

Lab4 – Real-Time Kernel

In this lab, students will be architect and building their own multi-threaded Real Time Operating System (RTOS). They implement context switching, mutexes, and enforced fixed priority scheduling.

Lab5 – Interfacing to Linux

In this lab, students will design and implement a PID controller in order to control the position of a wheel attached to a motor. The system will be architected as 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. The RPI will interface with the microcontroller using a Linux kernel module that communicates over SPI.