EECS 192: Mechatronics Design Lab

Discussion 1: Introduction

GSI: Ducky Lin, Derek Chou

January 22, 2020 (Week 1)

- Administrivia
- FRDM Board Intro
- Soldering
Welcome to EE192!
Project

- Project: build an autonomous track-following racecar given a stock chassis and microcontroller dev kit
- Teams should be 3 students
  - Combined skillset should include mechanical design / fabrication, electronics, programming
  - Controls experience helpful
- Teams formed by checkoff Friday
- Read the competition rules
  - NATCAR
Checkoffs

- One-hour time slot on Friday TBD to demonstrate that your project is where it should be
- At least one team member needs to show up to run your hardware
- These are graded, half credit if late

- First checkoff this Friday
  - Form project teams and check out cars
  - Trade a $300 deposit check for a car
  - Get private course GitHub repository
  - Details on website

Get your cars!
The Project in more detail
(Project Proposals Due February 7, 2020)
Planning & Reliability

- Get started early thinking about how to approach the project
- Measure once, cut twice, then hammer
- Measure twice, cut once
- Start thinking about high-level project plan
  - Plan ahead and examine feasibility
  - Get feedback on design ideas
- Reliability first, THEN performance
  - “Better is the enemy of the good enough”
  - Very fast car going into a wall (and breaking) gets you few points
  - Fast enough car hitting all of the checklist items gets you all the points

One weird trick to flunk ee192!

image from LOLCaption.com
Goal: Solid plan for mounting boards / other mechanisms
  - Level of detail: screw holes, dimensions
  - Paper and pencil is acceptable
    - Possibly even the most expedient solution
  - Cameraphone to take and annotate photos
  - Draw over picture in Paint or PowerPoint
    - Can even be physically accurate
  - SolidWorks only if you know how
    - Parametric CAD is really, really nice, but not worth learning for this

Use the provided diagrams!

Links: (top) (side) (CAD)
**Electrical**

- Goal: Motor driver design, power supply design and board layout plan
- Understand the boards we give you, and make your own!
- EDA tools
  - free & open source: (KiCAD, shown here)
  - freemium: (DipTrace) (EAGLE)
- Tutorials for each can be found on the internet
Car Critiques (Next Week)

- **Goal:** don’t reinvent the wheel
  - Assist in understanding all of the pieces that go into the project
  - Take design cues from those who came before you - recognize and use good ideas
  - Conversely, learn from others’ mistakes, so you don’t have to repeat them

- **Some design points to consider:**
  - Robustness
  - Maintainability
  - Design for Test
  - Graceful error handling
  - Anything else you want to add?

It’s been done before (don’t repeat it!)

© Fox
Hardware

- Beaglebone Blue (BBBL)
- Octavo Systems OSD3358 Processor
  - 1 GHz ARM Cortex-A8
  - 4 GB 8-bit eMMC flash storage
  - 512 MB RAM
- Program over network via USB or WiFi
- I/O connectors including
  - GPIO
  - 12-bit SAR ADC
  - USB2.0, I²C, SPI, and UART
- LEDs, Buttons, 9-axis IMU, Barometer
GPIO (general purpose input/output) pins (strawsondesign.com GPIO link)
- Output: sets pin voltage from software: either GND (0) or Vdd (1)
- Input: samples pin voltage: 0 (low) or 1 (high)

PWM (pulse-width modulation) module (strawsondesign.com PWM link)
- Every period, the pin is high based on the duty cycle, then low for the remainder
- Can digitally approximate analog outputs

Analog Inputs (ADC) (strawsondesign.com ADC link)
- Converts a continuous analog voltage (0-1.8v) to a 12-bit (0-4095) quantity
What would happen if the GPIO was 1?

The GPIO output would be 3.3v, and the LED lights up.

What if the GPIO was 0?

Nothing: the GPIO would be 0v, and no current flows across the LED.

What if the GPIO was PWMed?

Pulse Width Modulation (PWM) toggles an output between 0 and 1 "really fast", controlling the on and off ratio.

The LED would light at half intensity, but it may be (perceived as brighter).

"Analyze" this simple circuit.
Exercise: Hardware from Software

- What would happen if the GPIO was 1?
  - The GPIO output would be 3.3v, and the LED lights up
- What if the GPIO was 0?
  - Nothing: the GPIO would be 0v, and no current flows across the LED
- What if the GPIO was PWMed?
  - Pulse Width Modulation (PWM) toggles an output between 0 and 1 “really fast”, controlling the on and off ratio
  - The LED would light at half intensity, but it may be (perceived as brighter)

"Analyze" this simple circuit
Exercise: Hardware from Software

- What would happen if the GPIO was 1?
  - The GPIO output would be 3.3v, and the LED lights up

- What if the GPIO was 0?
  - Nothing: the GPIO would be 0v, and no current flows across the LED

- What if the GPIO was PWMed?
  Pulse Width Modulation (PWM) toggles an output between 0 and 1 "really fast", controlling the on and off ratio
Exercise: Hardware from Software

- What would happen if the GPIO was 1?
  - The GPIO output would be 3.3v, and the LED lights up

- What if the GPIO was 0?
  - Nothing: the GPIO would be 0v, and no current flows across the LED

- What if the GPIO was PWMed?
  Pulse Width Modulation (PWM) toggles an output between 0 and 1 "really fast", controlling the on and off ratio
  - The LED would light at half intensity, but it may be (perceived as brighter)
Exercise: Hardware from Software

- What would the GPIO read if the switch was pressed? (shorted)

"Analyze" this simple circuit

+3.3V

10kΩ

```
DigitalIn

-`
```
Exercise: Hardware from Software

- What would the GPIO read if the switch was pressed?
  (shorted)
  - The GPIO would read 0, because of the 0v at the pin
- What if the switch was not pressed?

"Analyze" this simple circuit
Exercise: Hardware from Software

- What would the GPIO read if the switch was pressed?
  - (shorted)
    - The GPIO would read 0, because of the 0v at the pin
- What if the switch was not pressed?
  - The GPIO would read 1, because of the 3.3v at the pin.
Robot Control Library

- C library hardware interface for Beaglebone Blue
- Examples and testing programs available
- Documentation:
  http://strawsondesign.com/docs/librobotcontrol/index.html
```
#include <stdio.h>
#include <robotcontrol.h>

int main()
{
    printf("Hello World!\n");
    rc_led_set(RC_LED_GREEN, 1);
    rc_usleep(500*1000);
    rc_led_set(RC_LED_GREEN, 0);
    rc_led_cleanup();
    return 0;
}
```
Detour: How to not kill your BeagleBoneBlue

- The supply of replacement boards is limited...
- Only power board from USB
- Do NOT use the barrel jack or the LiPo connector
  - because of a deficiency in the circuit design, this may fry the board

Don’t let this happen to you
Detour: How to not kill your BeagleBoneBlue

- Your body builds up static charge
  - ... just by walking, especially when dry
  - ... and up to several kV
    but under $\sim 2kV$ is imperceptible

- Chips are sensitive to high voltages:
  **may cause permanent damage**
  - read: board stops working
    “for no reason”

- Remember to ground (discharge) yourself before handling sensitive electronics
  - Touch the grounded lab bench surface
  - Use a ESD wriststrap
  - Avoid touching traces on boards

Don’t let this happen to you
Getting Started with your BeagleBone Blue

- We will be releasing a starter project soon...
- Two (not mutually exclusive) documented options for development environment:
  - Eclipse IDE on computer, cross-compiling to BBBL
  - Command-line compilation on BBBL
- Setup directions to come soon!
Live Demo!

connect to Beaglebone Blue, print "Hello World", toggle LEDs, GUI debug in Eclipse
So, how are you going to manage your code?

- main.cpp
- main_1.cpp
- main_final.cpp
- main_really_final.cpp
what a disorganized mess

- on a single team member’s laptop
what if their hard drive fails?
or they’re out sick during checkoff day?

- by email
another disorganized mess

- by email, with code in .doc files
I don’t even...

Don’t let this be your code. ©Fox
Use Git!

- **Git**: distributed version control software
  - Each commit: like complete snapshot
    - Full version history: you might not realize it now, but you’ll be glad you had it
  - Distributed: everyone has complete copy
    - Most operations local, periodically sync
  - Allows *branching* for concurrent work, which can be *merged*

- **Best Practices**
  - Small, logical, frequent commits
  - Write good commit messages
  - Keep master clean

Learn git here: try.github.io
GitHub Desktop Demo

Live Demo!

we wrote some code, we’re now going to commit it!

We recommend GitHub Desktop GUI for those new to Git.