N. Bunnell

10/09/2020

ECE-40153- Introduction to Embedded Systems

Section 151126, Fall 2020

Lesson 2 assignment 1

1. I’ve always thought of a microcontroller as “a microprocessor and...”, meaning that an MCU is the core architecture of some MPU design plus peripherals (GPIO, communication circuitry, RAM, flash, etc). There are times when either could be a solution to a design, however, as there are many criteria to consider on which would be the best when either would be acceptable.
2. As an example of the above, one *could* design a “general purpose” computer using only a microcontroller, say an AVR ATMega328 for example. However, that design would be severely limited in its capacity due to the limitations of onboard RAM and ROM, available GPIO and communication interfaces. No modern person would want to use a command line interface over a serial port to use their computer for whatever applications could be developed to fit and run under those constraints. Conversely, a modern x86-64 processor could be used in the design for a simple, limited/ dedicated function system, such as a pocket digital multimeter. The end product would likely be too cumbersome to be posket-sized, require too much power to be portable, and the costs for HW and SW design would be astronomical. Additionally, the extra power of a 64-bit based system versus an AVR in this use case would be wasted as the extra capacity for speed and precision quickly would jump into the territory of meaningless with such a tool.
3. Vehicle systems
   1. TPMS
      * An MCU with capacity for an ADC (pressure/ temp) and communication (some low power radio spec) back to a centralized controller
   2. Instrument Clusters
      * An MCU that can drive a display (digital systems), actuate a stepper or servo tied to a physical arm (analog systems), and some communication interface back to a centralized controller
   3. GPS
      * Could go either way, I would guess most GPU modules are either ASICs (kind of MCU-ish) or Systems On Chip (MPU-based). Regardless of the specific implementation, a GPS unit in an automotive context is still a subset of a larger system and needs to be able to communicate its processed data to rest of the appropriate system members (GPS, auto driving, tracking logs)
   4. Park/reverse assist
      * An MCU with some kind of sensor (light or sound based) and the ability to communicate back to a central controller
   5. Autonomous driving system
      * This could be an example of one of the controller systems mentioned above, that requires a large amount of processing capacity to both handle the amount of data from all of the above systems, as well as to prioritize one system’s feedback over another. For example, one tire being 1psi lower than the others should fall under a lower prioritization than an imminent collision being detected. Some MCUs on the market might have the capacity for this application but I would lean generally lean towards an MPU
4. With no design criteria other than what is listed in the question, I would lean towards a system of distributed MCUs with HW and SW specifically applicable towards the task they are assigned to accomplish. Additionally, I would intend the HW design to be modular, that is, in event of component failure within specific sections of the system, that task area’s constituent parts could be replaced without negatively affecting functionality to the rest of the system. For example, if I have an industrial control system monitoring inputs, controlling outputs, and communicating with a display, I would design so that my input capacity could be increased by swapping that section of the system for a different module while the rest of the system continues to function as-is
5. Many of the STM32 series chips, including the L475 on our IOT board, incorporate a CAN interface. Safety issues aside, I have always thought it would be very cool to interface with a vehicles CAN bus to pull various date pints such as engine temp, RPM, tire pressure, etc. and incorporate them into some kind of HUD projected on to the vehicle windshield. The STM32 IOT board has more than enough onboard horsepower for this while still being low powered enough that it could tap in to one of the unused 1A spots on the vehicle fuse box