# Physics 633, Microprocessors for Experimental Physics

**Course Description**

(3 S.H.) Students in the course make use of microcontrollers and/or single-board computers, eg Arduino, ATtiny, PIC, TI Launchpad, etc, to build physics lab equipment and solve design problems related to interfacing, control, and data acquisition. Topics include: Microcontroller structure, programming, digital I/O and interfacing, analog to digital conversion, and interrupts.  Students are encouraged to take Physics 632 and 633 concurrently.

Prerequisites: An introduction to Electrostatics (WSU Physics 202, 222, or similar) and Calculus (Math 213 or similar).  Instructor permission is also acceptable. Grade or P/NC. Offered every other year. The course is online with one or two optional in-person class meetings.

**Physics Subfield: Experimental Physics**

**Learning Outcomes**

1. Students will be able to program, compile, load, and run a microcontroller using C language.
2. The student will use the programming model of the microcontroller chip to write functional programs.
3. The student will use appropriate debugging tools to test and correct errors in the programs written in lab.
4. The student will connect a variety of circuits to the input and output ports of the microcontroller to measure and control external devices.
5. The student will maintain a laboratory notebook documenting the project requirements, hardware and software design, program outline and/or flowchart, and finished program code.

These goals will be assessed via weekly homework and occasional exams.

Additionally, students will:

1. Use a microcontroller to build an original data acquisition and control device for classroom or laboratory work

This goal will be assessed via a final project.

**Outline of Course Topics**

1. How can microprocessors make the lab more productive?
   1. Automation
   2. Response time
   3. Control
   4. Data Logging
2. Structure of microprocessors, microcontrollers, and single-board computers.
   1. Memory (Flash, SRAM, EEPROM)
   2. Clock
   3. Input/output (Analog, digital, parallel, serial, USB)
   4. Architecture families (AVR, ARM, PIC, x86)
   5. Power consumption
3. Programming (C programming of Atmel AVR ATmega328p)
   1. Variables (bool, int, float, char, arrays)
   2. Operations (mathematical, logical)
   3. Control structures (if/else/case, while, for)
   4. Finite-state machine architecture
   5. Functions
   6. Interrupts
   7. Libraries
   8. Compiling, debugging, and timing
   9. Considerations for small memories and slow clocks
4. Microcontroller I/O
   1. Digital out
      1. Using for communication, clocking
      2. Serial/parallel
      3. Using for timing
      4. Driving components via power transistors
   2. Digital in
      1. Use for communication
      2. Use for reading PWM
   3. Analog in
      1. Analog to digital conversion
      2. Fidelity (bit width)
      3. Sampling time/rate
   4. PWM out
      1. Duty cycle and carrier frequency
      2. Use for energy savings
      3. Use for communication
      4. Use for reproducing varying waveforms
   5. Interrupts
      1. Interrupts and Interrupt service routines
      2. External interrupt (pins)
      3. Clock/timer interrupts
5. Documentation
   1. Purpose
   2. Code comments
   3. Lab notebook
      1. overview
      2. Test conditions
      3. Errors/bugs
   4. Source/revision control
6. Labs to illustrate ideas:
   1. Blinking LED’s in parallel/serial (via shift register), controlling current with resistors, persistence of vision.
   2. Digital input switches as inputs that control a pattern of blinking lights, (Finite state machine)
   3. Analog to digital conversion with Linear (temperature) and non-linear (IR distance) sensors. Capacitors to smooth out data.
   4. Using internal timers/counters to measure the delay between input pulses (Geiger counter interfacing)
   5. Using libraries to write data to SD flash cards. Applications to remote data collection. Power considerations and strategic use of “sleep” states.
7. Example Projects
   1. Using a GPS antenna, make a location tracker (writing to sd card). Generate plots of velocity, acceleration, etc from locations.
   2. Using 3-axis accelerometer, make an instrument that records the acceleration vs time of an elevator.
   3. Using a Geiger counter, create an interrupt-based device that will record and display counts/second from the detector. Student designed final projects; including input, output, finite-states, and computation.
   4. Using a small PV panel (and a 3-axis accelerometer), take voltage & angle data from a ~1.5v PV panel. Record open circuit voltage vs time, and also find the optimal load resistance for power production via a transistor-actuated parallel ladder of drain resistors.