Lecture 8.1
Intro to 8-bit Embedded Systems ECE-231
Spring 2024

Prof. D. McLaughlin

ECE Dept.

UMass Amherst

David McLaughlin, Professor ECE

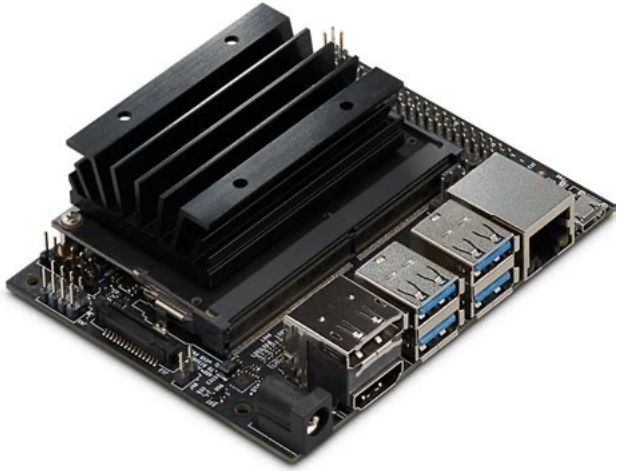
Pronouns: he/him

- Office: 211 Marcus Hall
- dmclaugh@umass.edu
- Bio
  - BS ECE, UMass 1984 (CSE then EE)
  - PhD EE 1989 (Radar Remote Sensing)
  - ECE Faculty Northeastern University ’89 – ’99
  - UMass ECE Faculty ‘99 till present (roles: Professor, Assoc Dean, Academic Dean, Center Director)
  - 10 years as Engineering Fellow (part time) at Raytheon Co.
- Interests: Hardware, Systems
- Recent Teaching:
  - ECE – 597 Innovation & Entrepreneurship (in collaboration with Prof. Bill Wooldridge)
  - ECE – 297 Queer Lights (in collaboration with Dr. Genny Beemyn)
  - ECE - 361 Fundamentals of EE (Electronics for non-ECE engineers)
  - ECE – 231 Intro Embedded Systems (in collaboration with Prof. Fatima Anwar)
  - ECE – 304 Junior Design Project
  - ECE – 697 Weather Radar Networks

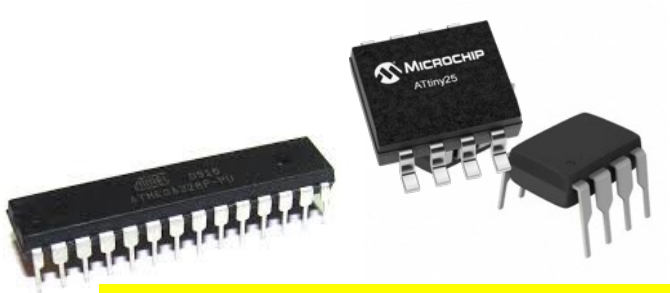
Some logistics:

- Attend all lectures & labs
- Bring your laptops to class, else you won't be able to see the small fonts
  - download the pdf's before class, follow along
- Please don't talk while I'm lecturing
- Use Piazza for peer & TA questions
- See me during office hours
- Contact me via email: dmclaugh@umass.edu
- If you need me: 413-896-8618

Embedded Computers



Small form factor 32 & 64 bit computers running Linux OS with native Python, C compilers, other apps



8 bit MCU system-on-a-chip running a single app in machine opcode(*)

Increasing size, weight, power, heat, interface flexibility, functionality

(*)app originally written in C on a host computer, then cross-compiled into the machine-understandable binary code of the target MCU.

8 bit embedded systems – 2nd half of course

Mac Book Pro, 64 bit intel core i9

- OS: Mac OS, Win 10, Linux
- Large Li-ion; daily charge
- $1-$2k purchase price
- 500GB SSD, 16GB RAM
- clock: 2.6 GHz
- 12 W idle, 35 W CAD
- Standard I/O: keyboard, monitor; mouse; trackpad; speaker; WiFi; light sensor; USB ports
- Separate microprocessor, RAM,ROM, hard disk, peripherals

8 bit Microcontroller Unit (MCU)

- No OS: bare-metal application
- Small battery, last months/years
- $1-$4 purchase price
- 32KB flash ROM, 2KB RAM
- clock: 1-20 MHz
- 5V, 0.1 mA = 0.5mW
- Headless: no monitor, keyboard, mouse; 3 GPIO ports, 6 ADC channels, 3 timers, serial port, SPI, I2C,
- Everything on a single IC

Recall:
General Purpose Computer – vs Microcontroller

Gen Purpose Computer

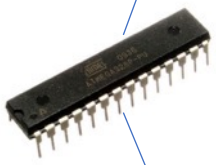
Microprocessor:
- Control Unit
- ALU
- Registers

RAM | ROM

Input | Output

Timers | Serial Comm

Microcontroller



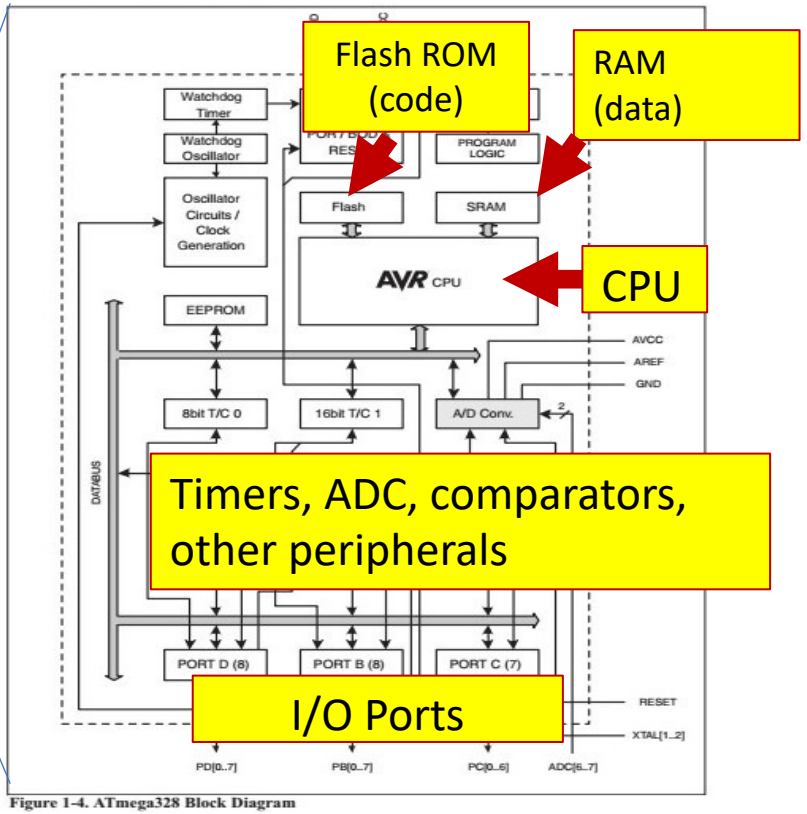
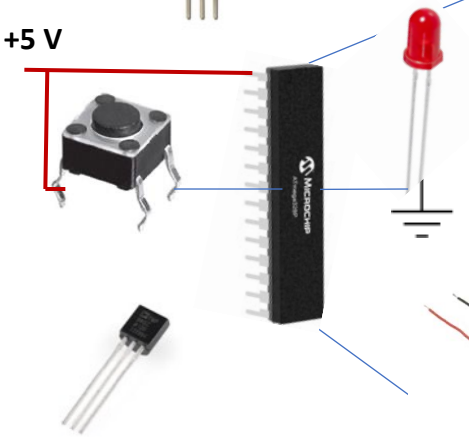
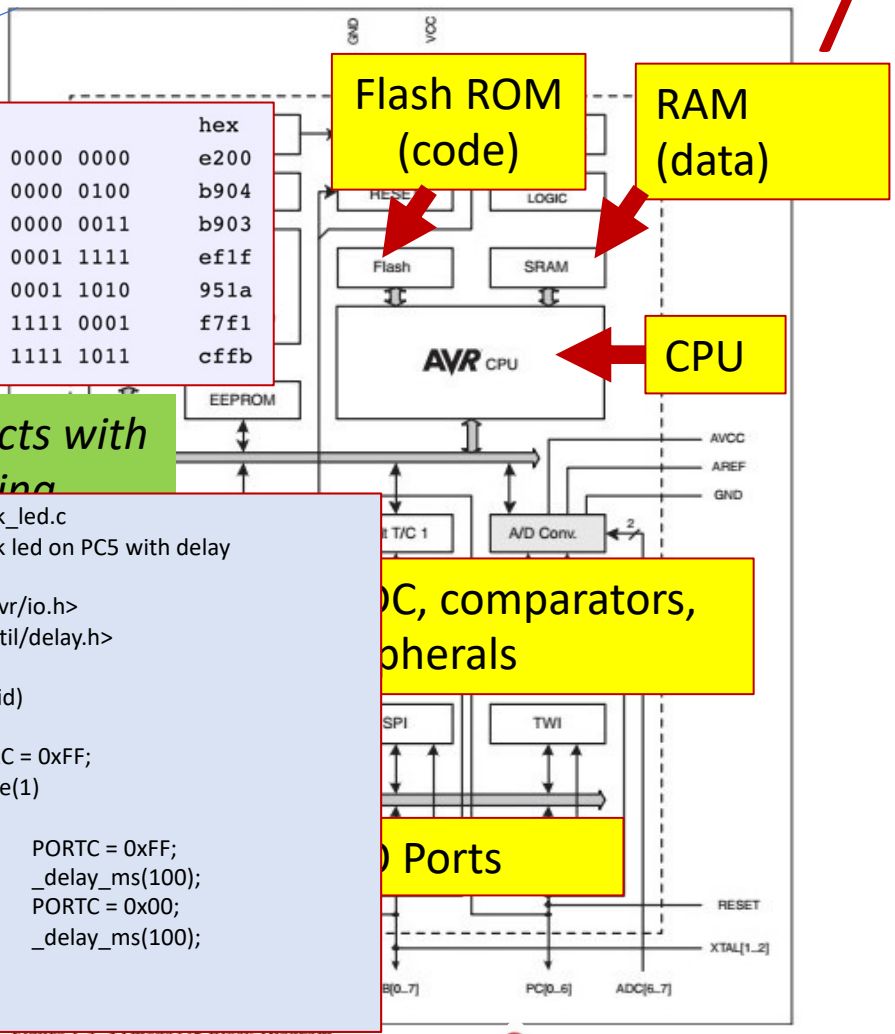


Figure 1-4. ATmega328 Block Diagram




```
bin                  hex
1110 0010 0000 0000  e200
1011 1001 0000 0100  b904
1011 1001 0000 0011  b903
1110 1111 0001 1111  ef1f
1001 0101 0001 1010  951a
1111 0011 1111 0001  f7f1
1100 1111 1111 1011  cffb
```



machine code interacts with physical world by doing reads/writes t locations (reg associated wi

```
/*      blink_led.c
        blink led on PC5 with delay
*/
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
        DDRC = 0xFF;
        while(1)
        {
                PORTC = 0xFF;
                _delay_ms(100);
                PORTC = 0x00;
                _delay_ms(100);
        }
}
```

C code

assembly/machine code

architecture (processor, memory, I/O registers)

logic gates, flip flops

transistors

voltages, currents

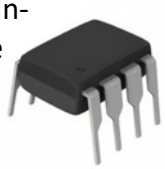
8 bit bare metal embedded programming involves:

- Reads & writes to memory locations (registers) in the I/O interfaces, to interact with sensors & actuators connected to ports
- Configuring peripherals (timers, input/output ports, ADCs, etc…)
- Loops (to repeat things)
- Conditional branching (if something xxx, else yyy)

ATtiny85 (left) & ATmega328P (right) MCUs

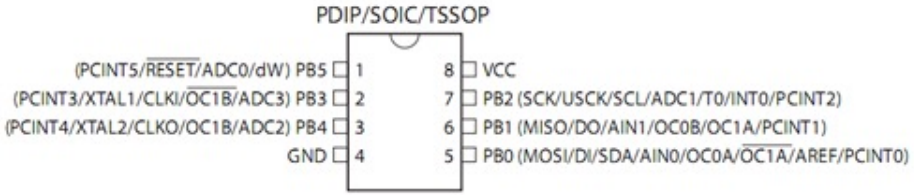
$2.49 from Jameco

Plastic Dual In-Line Package (PDIP)



Small Outline Integrated Circuit Package (SOIC)

Pinout ATtiny25/45/85



- 6 General Purpose I/O (GPIO) pins
- 4 Analog to Digital Converter (ADC) pins
- 2 Timers
- no communication

- 8K Byte Flash ROM for Code
- 128 Byte data RAM
- 128 Byte data EEPROM

$2.59 from Jameco

28 pin PDIP

32 pin Thin Quad Flat Package (TQFP)

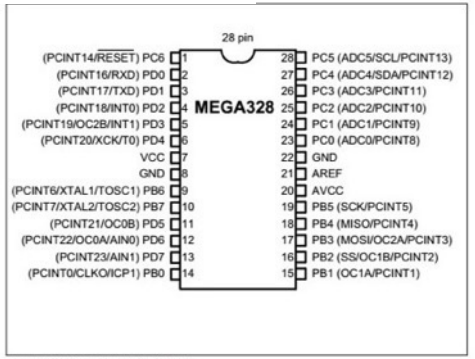
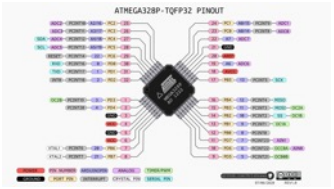


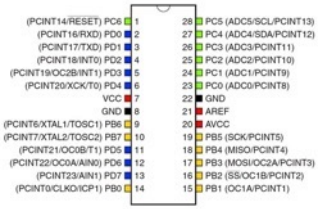
Figure 4-1. ATmega328 Pin Diagram

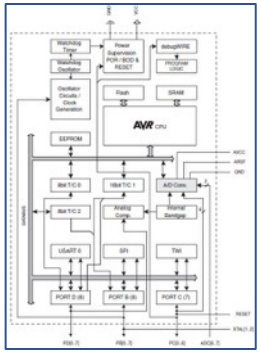
- 23 General Purpose I/O (GPIO) pins
- 8 Analog to Digital Converter (ADC) pins
- 3 Timers
- USART for serial communication

- 32K Byte Flash ROM for Code
- 2K Byte data RAM
- 1K Byte data EEPROM

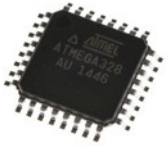
Programming the ATmega328P





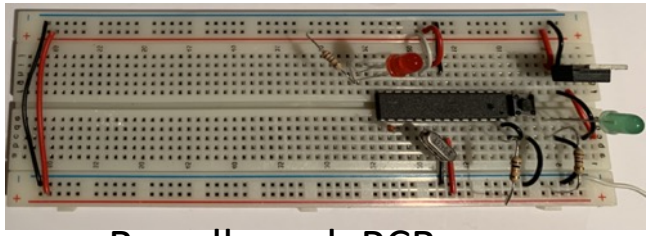


- Data Sheet (100's pages)
- Chip pinout
- Functional block diagram
- Registers
- Downloaded machine code
- External sensors, actuators
- Support circuitry: power, clock crystal
- Programming interface
- Debugging

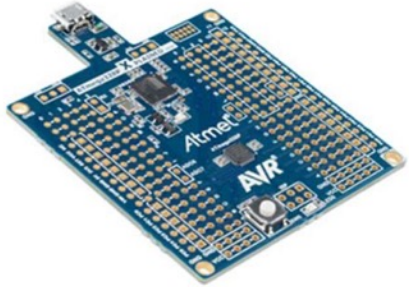


Thin Quad Flat Package (TQFP)

Dual In-line Package (DIP)



Breadboard, PCB



development boards

External programmers

Creating an Embedded System

Host Computer

Integrated Development Environment (IDE) – Microchip Studio 7 (Win 10)MPLAB X (Win, MacOS) coding text editor (Visual Studio Code)

1. Source code written in C

2. Code is compiled, linked with other objects to create an executable program for the target processor (ATMEGA328)

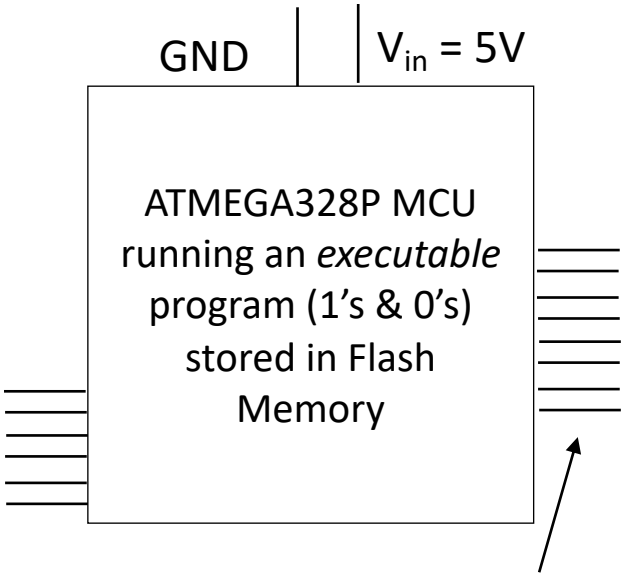
USB

3. MCU programmer flashes code to MCU

ISP

ATMEGA328P MCU running an *executable* program (1's & 0's) stored in Flash Memory

MCU Integrated Circuit



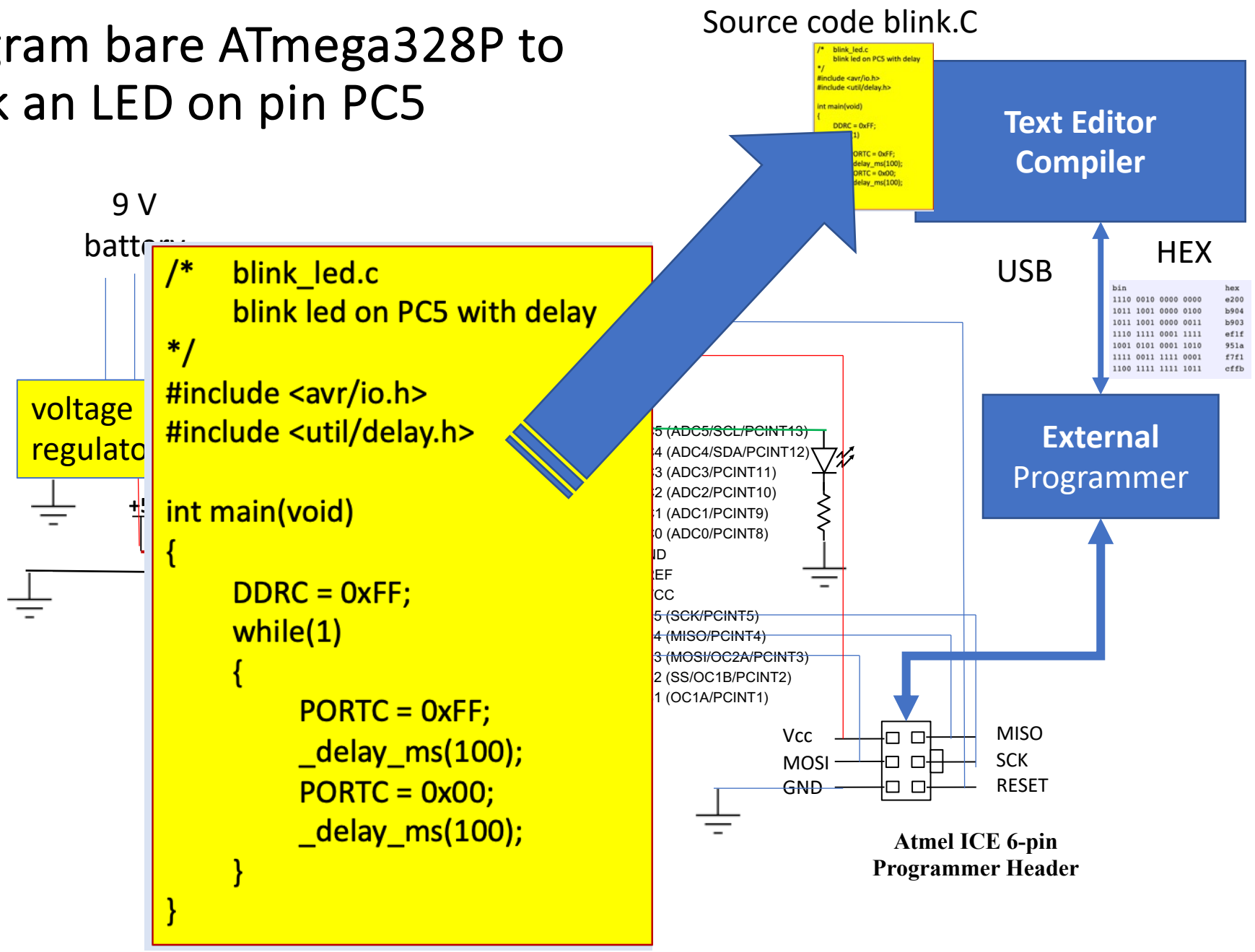
Analog Input (ADC) Pins

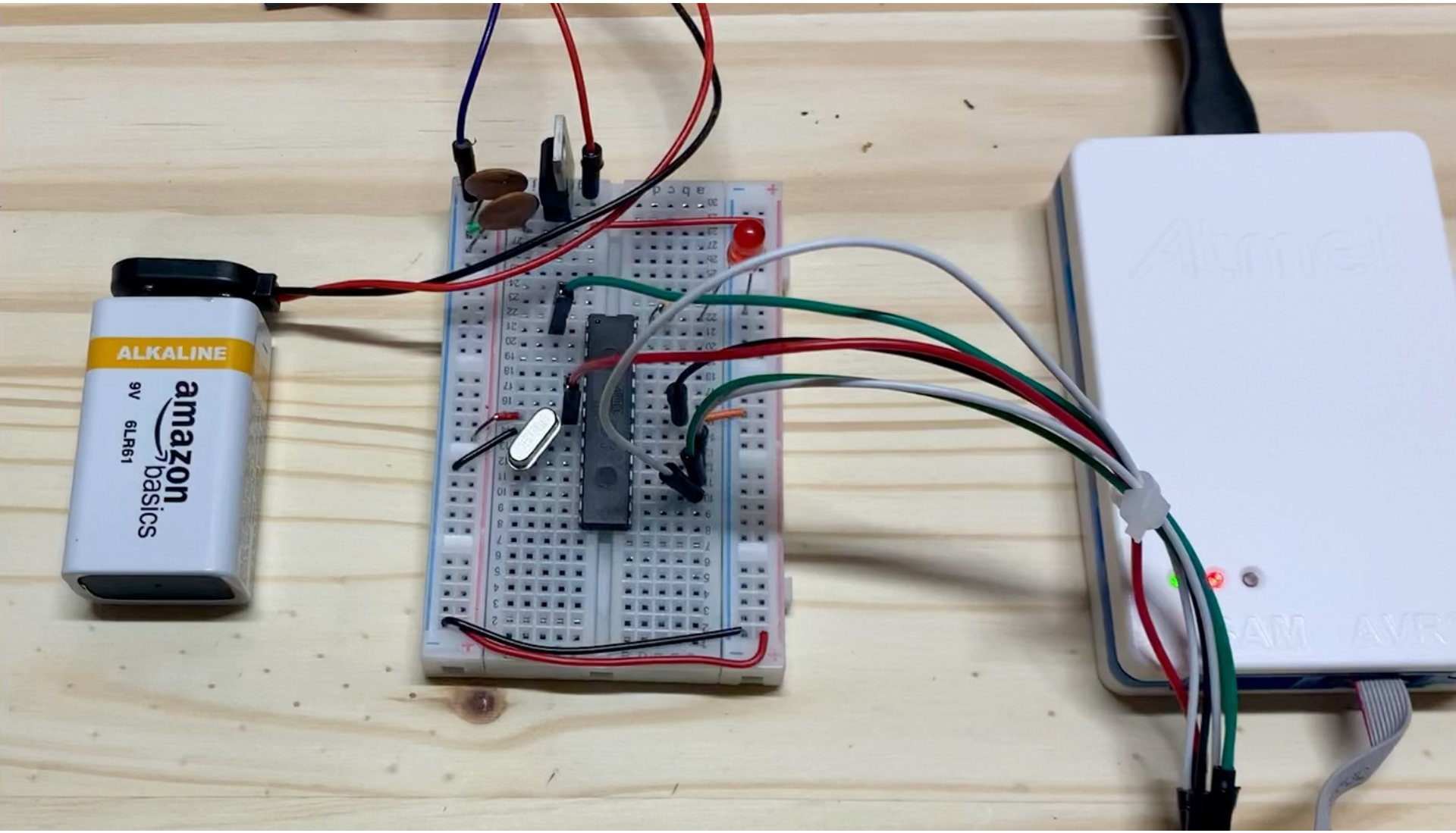
- Input voltage typically between 0 and 5V
- 10 bit representation of the input voltage (0 to 1023)
- 0V interpreted as 0
- 5V interpreted as 1023

General Purpose Input & Output (GPIO) Pins

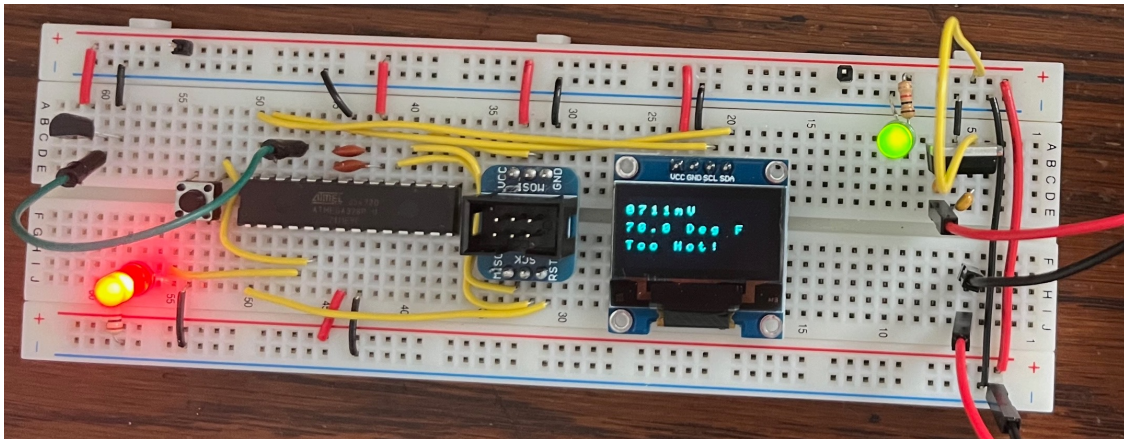
- logically 0 or 1
- electrically 0V or 5V
- Output pins: assert 0V (0) or 5V (1) on a pin, causing something external to happen
- Input pins: read 0V (0) or 5V (1) to interpret something external

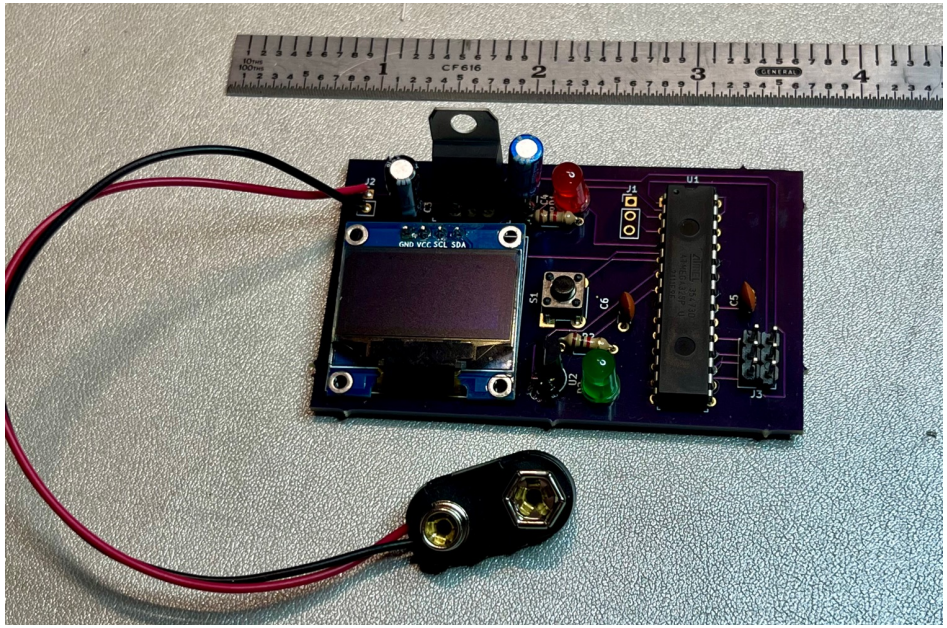
Program bare ATmega328P to blink an LED on pin PC5




ALKALINE
amazon basics
9V
6LR61

Example of a build for ECE-304: Digital Thermometer, “megaTemp”





Creating an Embedded System in 2nd half of ECE 231

Visual Studio Code (free editor) [Not Visual Studio!]

1. Source code written in C

2. Command line: Code is compiled, linked with other objects to create an executable program for the target processor (ATMEGA328)

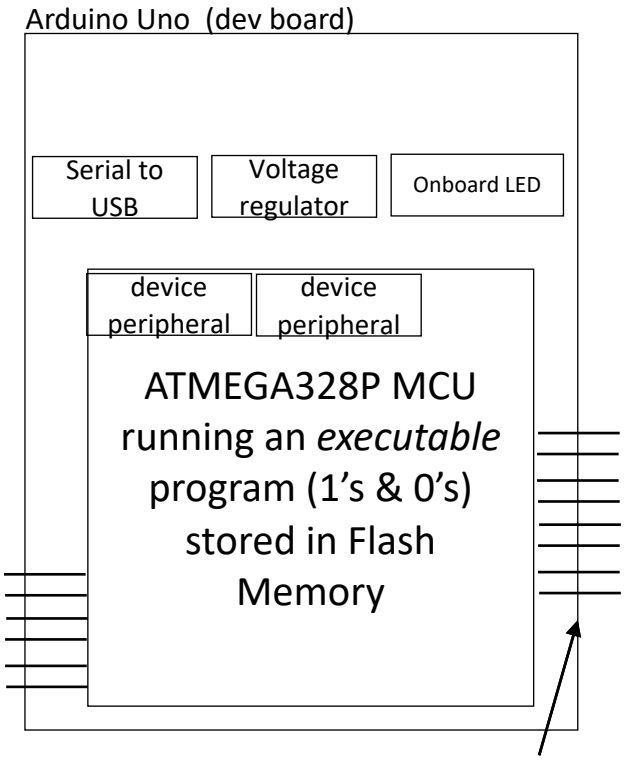
USB

Arduino Uno

Serial to USB converter

serial

ATMEGA328P MCU running an *executable* program (1's & 0's) stored in Flash Memory



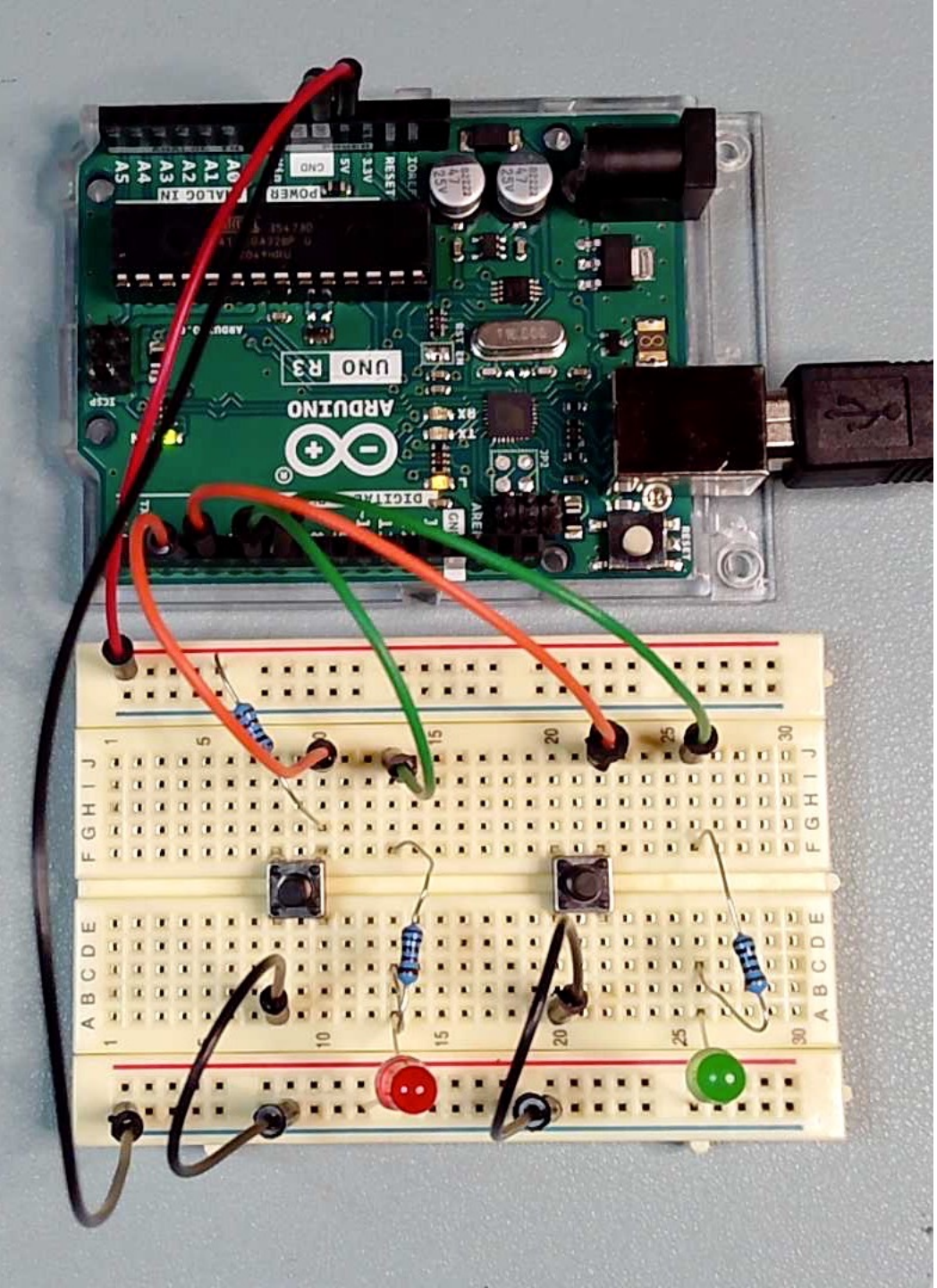
Analog Input (ADC) Pins

- Input voltage typically between 0 and 5V
- 10 bit representation of the input voltage (0 = 1023)
- 0V interpreted as 0
- 5V interpreted as 1023

General Purpose Input & Output (GPIO) Pins

- logically 0 or 1
- electrically 0V or 5V
- Output pins: assert 0V (0) or 5V (1) on a pin, causing something external to happen
- Input pins: read 0V (0) or 5V (1) to interpret something external

Internal LED (PB5) On & Off for 0.1 Second, repeat



Remember:

We’re not programming an Arduino.

We’re not programming an Arduino Uno

We are programming an ATmega328P microcontroller (MCU) that happens to be on an Arduino Uno development (dev) board.

ECE-231 8 bit Programming Environment

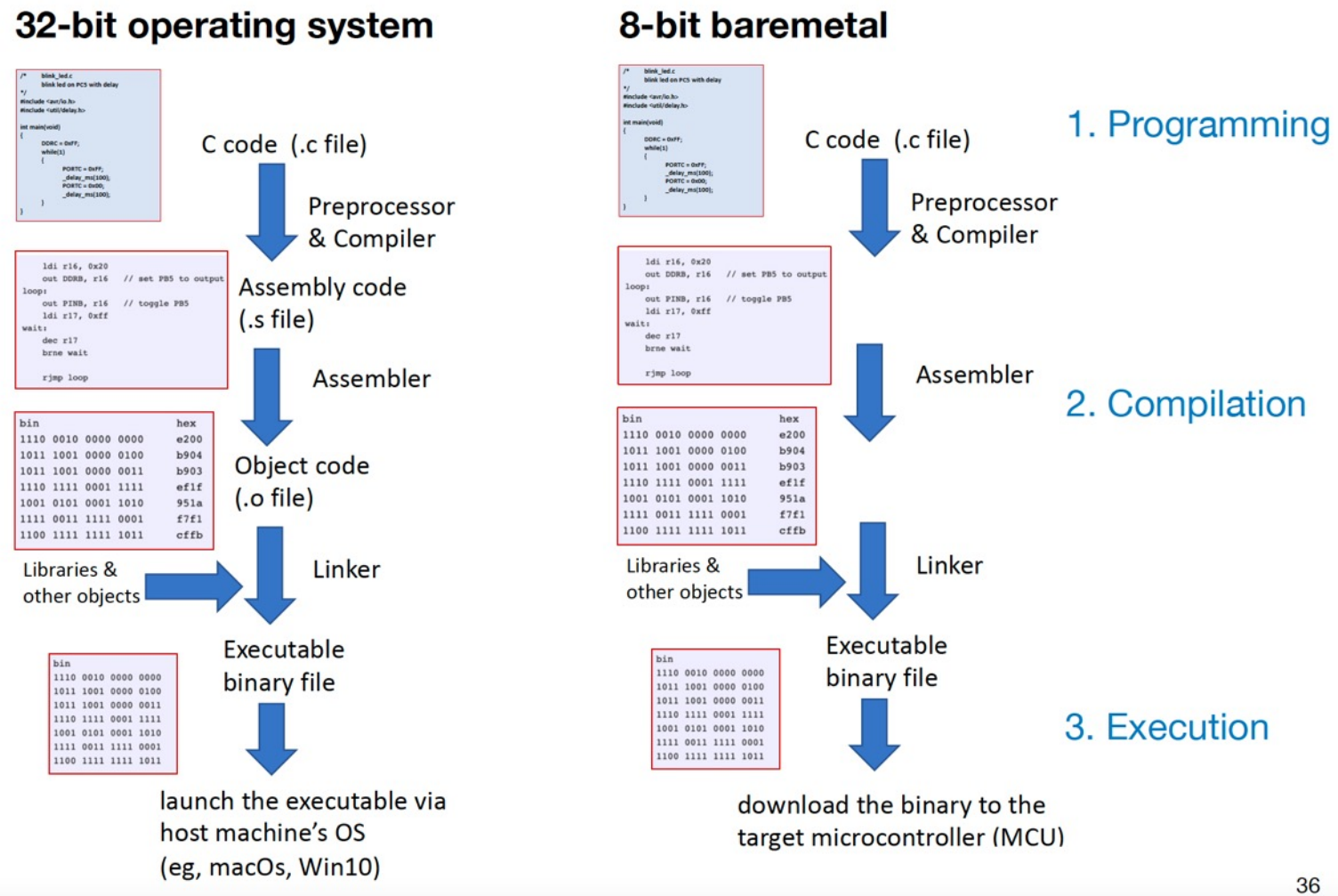
- Visual Studio Code (VS Code) – text editor
  - https://code.visualstudio.com
- avr-gcc toolchain
  - Windows: WinAVR-20100110-install.exe
    - https://sourceforge.net/projects/winavr/files/WinAVR/20100110/
  - Mac: AVR 8 bit toolchain (OSX) version 3.7.0
    - https://www.microchip.com/en-us/tools-resources/develop/microchip-studio/gcc-compilers
    - also: install homebrew, xcode developer tools (not the full xcode system), avrdude, and possibly make
- Command Line
  - Windows: command line
  - Mac: terminal

We will set up our programming environments & conduct first system build (blinking LED) in lab this week.

Creating an embedded C project

- First, create a project directory on your host computer
- create a file of C *source code* in a text editor, save with .c extension
- *compile* the source code into executable code that can be understood & run by your machine (ie, Intel processor, ARM processor, embedded processor, etc...)
- Deal with the inevitable series of compiler warnings and errors
- Wire up your hardware, check wiring
- Flash the binary executable to the target MCU, where it will then start running
- Debug hardware & software as needed.

Software Life Cycle



TENTATIVE SCHEDULE FOR ECE-231 PART 2, 8-BIT EMBEDDED SYSTEMS

VERSION 1.0 Spring 2024

Day	Date	Lecture	Lab
Mon	4/1/24	8.1 Intro	
Tues	4/2/24		Lab 8.0 Install tools & blink
Wed	4/3/24	8.2 GPIO	Lab 8.0 Install tools & blink
Thurs	4/4/24		Lab 8.0 Install tools & blink
Fri	4/5/24		Lab 8.0 Install tools & blink
Sat	4/6/24		*Lab 8.0 Due 4/6/24*
Sun	4/7/24		
Mon	4/8/24	8.3 GPIO (Taped)	
Tues	4/9/24		Lab 8.1 GPIO
Wed	4/10/24	8.4 GPIO	Lab 8.1 GPIO
Thurs	4/11/24		Lab 8.1 GPIO
Fri	4/12/24	8.5 UART *(Monday Schedule)*	Lab 8.1 GPIO (keep Fri Schedule)
Sat	4/13/24		*Lab 8.1 Due Sat 4/14/24*
Sun	4/14/24		
Mon	4/15/24	*Holiday*	
Tues	4/16/24		Lab 8.2 Dig Thermometer
Wed	4/17/24	8.6 LEDs	Lab 8.2 Dig Thermometer
Thurs	4/18/24		Lab 8.2 Dig Thermometer
Fri	4/19/24		Lab 8.2 Dig Thermometer
Sat	4/20/24		
Sun	4/21/24		
Mon	4/22/24	8.7 ADC	
Tues	4/23/24		Lab 8.2 Dig Thermometer
Wed	4/24/24	8.8 ADC, Timers	Lab 8.2 Dig Thermometer
Thurs	4/25/24		Lab 8.2 Dig Thermometer
Fri	4/26/24		Lab 8.2 Dig Thermometer
Sat	4/27/24		*Lab 8.2 Due Sat 4/27*
Sun	4/28/24		
Mon	4/29/24	8.9 DEMO	
Tues	4/30/24		Lab 8.3 Ultrasound
Wed	5/1/24	8.10 Infrasound, OLED	Lab 8.3 Ultrasound
Thurs	5/2/24		Lab 8.3 Ultrasound
Fri	5/3/24		Lab 8.3 Ultrasound
Sat	5/4/24		
Sun	5/5/24		
Mon	5/6/24	8.11 Additional timers	
Tues	5/7/24		Lab 8.3 Ultrasound
Wed	5/8/24	8 bit review (DM)	Lab 8.3 Ultrasound
Thurs	5/9/24		Lab 8.3 Ultrasound
Fri	5/10/24		Lab 8.3 Ultrasound
Sat	5/11/24		*Lab 8.3 due Sat 5/11*

ECE-231 Assignment #8.0
Assigned Monday 4/1/24
Due: 11:59 pm Saturday 4/6/24

There is nothing to turn in for this lab but you do need to demonstrate rapid blinking LED to your lab TA.

1. Visit https://github.com/ProfMcL/ECE231 and look over the ECE231 repository. We will be referring to code and documents in this repository throughout the rest of the semester. You can either download or clone the repo or visit it from time to time as needed.

2. Download and install the programming tools needed for your Windows of macOS machine. You'll find detailed instructions in the ECE231 repo.

3. Make a fast-blinking LED on your Arduino Uno board by following the above-mentioned instructions. Show it to your lab TA to demonstrate that you've successfully installed all necessary tools and that you're able to flash a simple code to the ATmega328P on your Arduino Uno dev board.

Version 1.0 March 31, 2024

Link: https://github.com/ProfMcL/ECE231