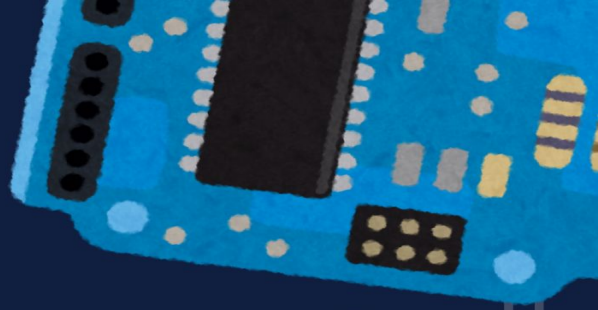
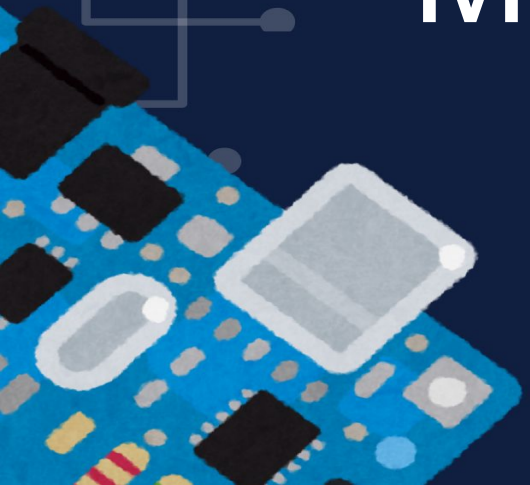


ACM Special Topics Meeting - Arduino

October 9th, 2024



Join our Discord!

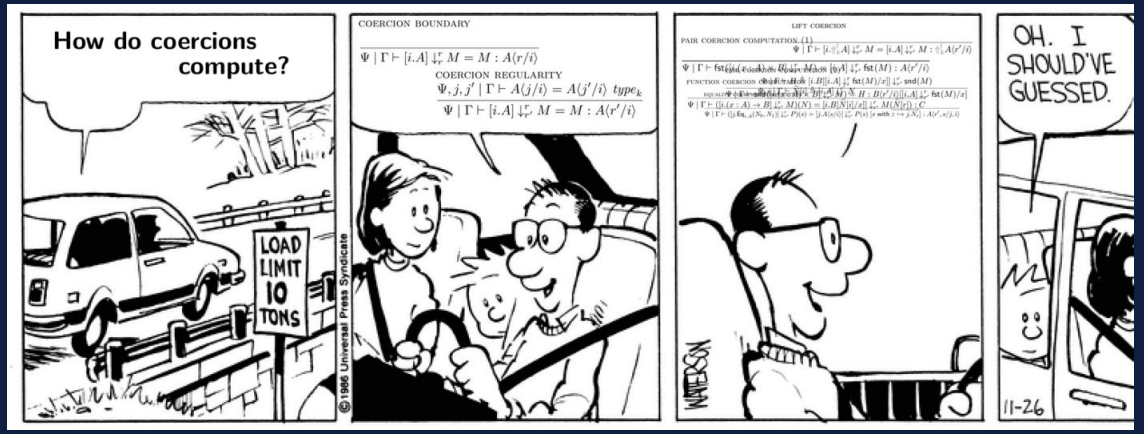
Find Research Opportunities!

Executive Board Introductions

- President: Ell Torek
- Vice President: Caden Allen
- Treasurer: Christina Foley
- Secretary: Ryon Peddapalli
- Communications Chair: Kate Fullero

Eli

- sophomore math/cs major
- cuhackit, icpc, and putnam
- undergrad ta for cpsc 1070 for 2 semesters
- interested in compilers + type theory



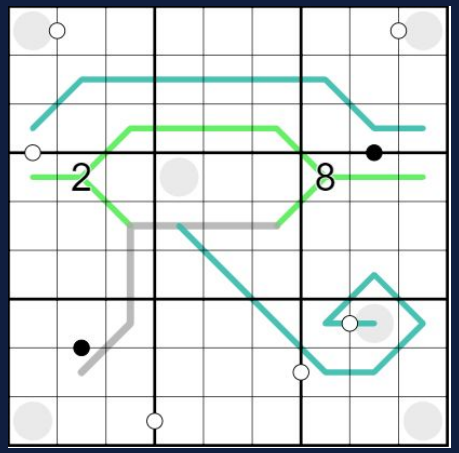
Caden the Nerd

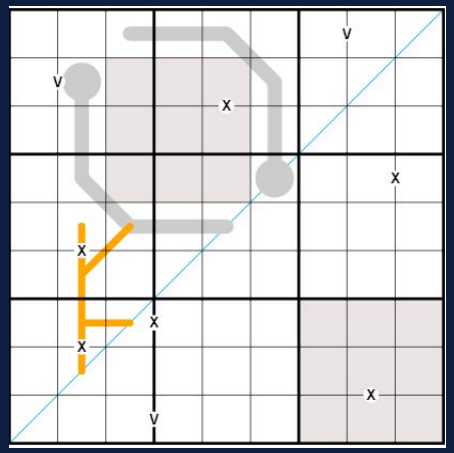
- 6 foot 9 ig
- Handsome, intelligent, athletic ig
- Nonchalant dreadhead ig
- (the paper towel roll is crazy ngl)

- 2 different embedded systems labs (PERSIST and TigerSec)
- Supreme Lord of Google (I can debug really well)
- Organizer for Clemson embedded systems CTF (please join)
- Will prob get Phd in embedded field
- Will do CyberBoat challenge (ask me after meeting)

Christina

- Computer Science undergrad
- CuHackit, College Loops, ICPC
- 1070 TA
- Grass Volleyball enthusiast
- Mildly obsessed with modded sudoku & mythology





Ryon

- Always eternally 1 inch taller than the tallest person in the room
- #1 hater of pineapple pizza
- Math / CS major
- #1 resident canposter of the CPSC discord
- I fence
- I’m banned from bajablast.com

Kate - CS (AI minor)

- Gifted & talented kid to burnt-out adult pipeline
- Side quests:
  - bobarista
  - modeled for wedding magazine & holiday spread
  - licensed nail tech
- Was forced to learn violin by parents
- 1070 expert
- Likes to cook & prob will make a cooking channel (side quest tbd)
- College Loops, SWE, CU Symphony, CU Quartet

Arduino Programming with Caden

1. Arduino Overview
2. Arduino IDE install
3. Blinky!
4. Secret Assignment ;)

What is Arduino?

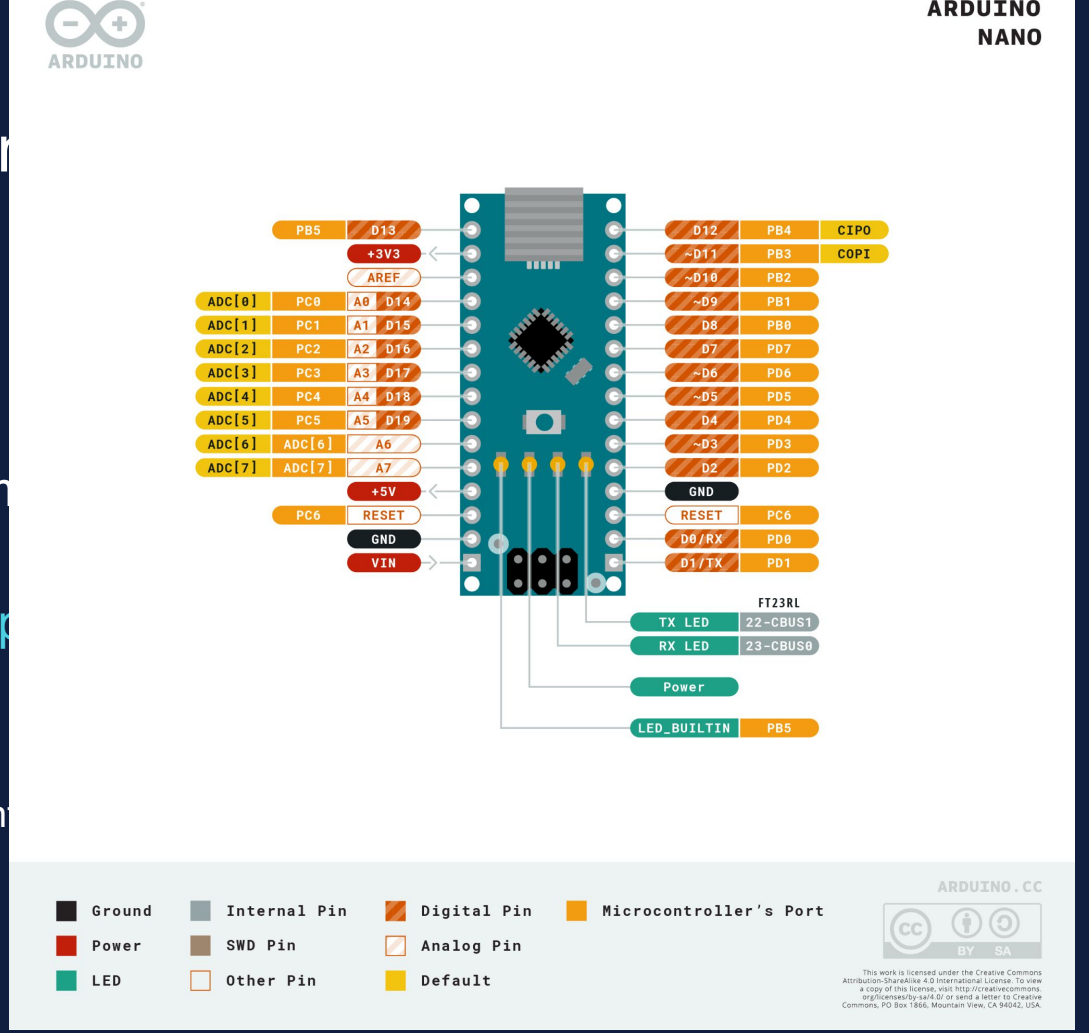
- Most popular hobbyist / multipurpose-production grade MCs and MPs
- Microcontrollers (MC) / Microprocessors (MP), simple to complex
- Very beginner friendly
- One of the most satisfying ways to learn C
- Extremely well-made documentation
- Extremely well-made pinout diagrams
- Very good for learning!!!

Parameter	Microprocessor	Microcontroller
Definition	Microprocessors can be understood as the heart of a computer system.	Microcontrollers can be understood as the heart of an embedded system.
What is it?	A microprocessor is a processor where the memory and I/O component are connected externally.	A microcontroller is a controlling device wherein the memory and I/O output component are present internally.
Circuit complexity	The circuit is complex due to external connection.	Microcontrollers are present on chip memory. The circuit is less complex.
Memory and I/O components	The memory and I/O components are to be connected externally.	The memory and I/O components are available.
Compact system compatibility	Microprocessors can't be used in compact system.	Microcontrollers can be used with a compact system.
Efficiency	Microprocessors are not efficient.	Microcontrollers are efficient.
Zero status flag	Microprocessors have a zero status flag.	Microcontroller doesn't have a zero status flag.
Number of registers	Microprocessors have less number of registers.	Microcontrollers have more number of registers.
Applications	Microprocessors are generally used in personal computers.	Microcontrollers are generally used in washing machines, and air conditioners.

source for image

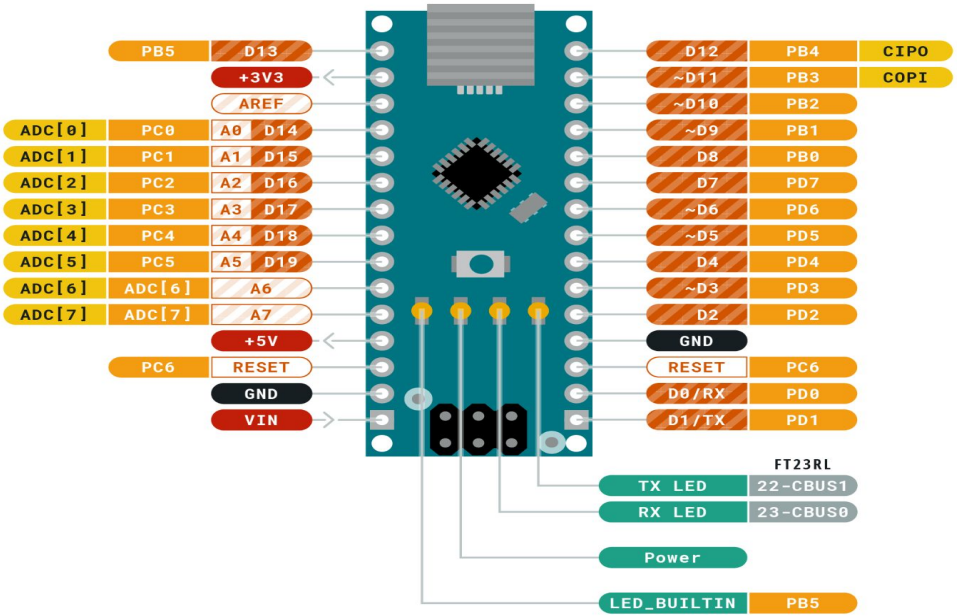
What board will we be usi

- link to board
- Arduino Nano ATmega328
- 14 digital pins
- Serial 0 and 1 for receive / transm
- LED (pin 13)
- More nerd crap that is out of scop
- We will stay simple
- How to read pinout?
  - Test what you can guess just from in



ARDUINO

ARDUINO NANO



Install Arduino IDE

- Compiling code on arduino is possible (and more efficient) without the IDE, but simple is good for now
  - I will not give away my secrets, not yet at least ;)
- Use this link to install IDE (or search it up, heh)
- Install the drivers they prompt you to (this is not spyware I promise)
- Now, plug in your usb to Arduino (be gentle)
- Select Arduino Nano for your board, and we can start!

Setting up Blinky!

- Blinky is the “Hello World!” of embedded, the first thing you do with ANY new MC or MP (if they have LED)
- Arduino is very beginner friendly, blinky is (strictly speaking) 3 lines of code!
- For more complex and “expert” centered boards, blinky could easily be up to 50 lines of code.
- Now, go to file > examples > basics > blinky
- Before anything, use intuition to understand the code
    - What do you understand? Not understand? Very important first step
- Now, time to compile and upload!

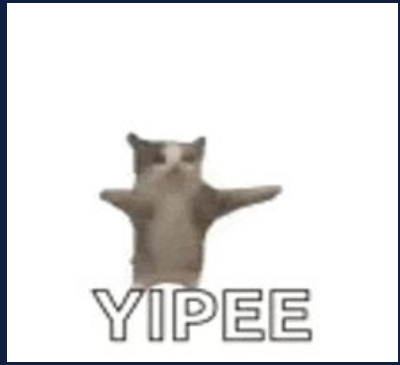
How to Debug

- You when the board doesn't blink →

- Why does this not work?
- Debugging can be very complicated, but let us simplify it with Google!
- Arduino is *the* most beginner friendly because of their amazing forums!
- Try some google searches (or on the forum) to figure out what went wrong
  - Hint: always use the board name in your queries
  - First to find out gets a cookie (not really)
  - If you are desperate, yell out "O wise Caden, what shall I do to appease the machine god?"
  - DO NOT LOOK AT NEXT SLIDE (idk if we posted the slides at this point or not)

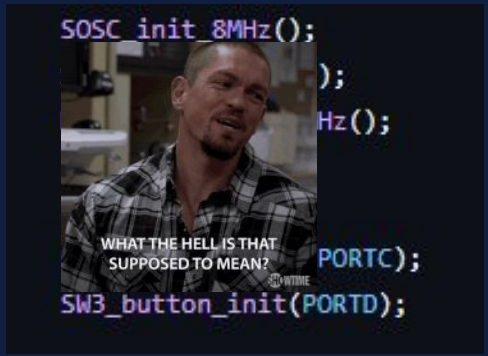
Problem Solved!

- Who solved the problem? What was the problem? How did you solve it?
- Learning to google answers is very, very important for learning embedded!
- You learn to understand others code, learn from their mistakes, and fix yours issues all at once
- Time spent on documentation is very rewarding
- I highly suggest being comfortable posting on forums



Understanding Blinky

- Arduino is very straightforward, and a
  lot of the more difficult aspects of blinl
  - For example, here is blinky on another MC
    (S32K144-Q100):



Highly technical nerd stuff, but know that this MC requires you to manually set up system clock. No clock, no ability to time instructions: how can it run code?

```
#include "clocks_and_modes.h"

void SOSC_init_8MHz(void) {
 SCG->SOSCDIV=0x00000101; /* SOSCDIV1 & SOSCDIV2 =1: divide by 1 */
 SCG->SOSCCFG=0x00000024; /* Range=2: Medium freq (SOSC between 1MHz-8MHz)*/
 /* HGO=0: Config xtal osc for low power */
/* EREFS=1: Input is external XTAL */
 while(SCG->SOSCCSR & SCG_SOSCCSR_LK_MASK); /* Ensure SOSCCSR unlocked */
 SCG->SOSCCSR=0x00000001; /* LK=0: SOSCCSR can be written */
 /* SOSCCMRE=0: OSC CLK monitor IRQ if enabled */
/* SOSCCM=0: OSC CLK monitor disabled */
/* SOSCERCLKEN=0: Sys OSC 3V ERCLK output clk disabled */
/* SOSCLPEN=0: Sys OSC disabled in VLP modes */
/* SOSCSTEN=0: Sys OSC disabled in Stop modes */
/* SOSCEN=1: Enable oscillator */
 while(!(SCG->SOSCCSR & SCG_SOSCCSR_SOSCVLD_MASK)); /* Wait for sys OSC clk valid */
}
void SPLL_init_160MHz(void) {
 while(SCG->SPLLCSR & SCG_SPLLCSR_LK_MASK); /* Ensure SPLLCSR unlocked */
 SCG->SPLLCSR = 0x00000000; /* SPLLEN=0: SPLL is disabled (default) */
 SCG->SPLLDIV = 0x00000302; /* SPLLDIV1 divide by 2; SPLLDIV2 divide by 4 */
 SCG->SPLLCFG = 0x00180000; /* PREDIV=0: Divide SOSC_CLK by 0+1=1 */
 /* MULT=24: Multiply sys pll by 4+24=40 */
/* SPLL_CLK = 8MHz / 1 * 40 / 2 = 160 MHz */
 while(SCG->SPLLCSR & SCG_SPLLCSR_LK_MASK); /* Ensure SPLLCSR unlocked */
 SCG->SPLLCSR = 0x00000001; /* LK=0: SPLLCSR can be written */
 /* SPLLCMRE=0: SPLL CLK monitor IRQ if enabled */
/* SPLLCM=0: SPLL CLK monitor disabled */
/* SPLLSTEN=0: SPLL disabled in Stop modes */
/* SPLLEN=1: Enable SPLL */
 while(!(SCG->SPLLCSR & SCG_SPLLCSR_SPLLVLD_MASK)); /* Wait for SPLL valid */
}
void NormalRUNmode_80MHz (void) { /* Change to normal RUN mode with 8MHz SOSC, 80 MHz PLL*/
 SCG_RCCR_SCS(6) /* PLL as clock source*/
 |SCG_RCCR_DIVCORE(0b01) /* DIVCORE=1, div. by 2: Core clock = 160/2 MHz = 80 MHz*/
 |SCG_RCCR_DIVBUS(0b01) /* DIVBUS=1, div. by 2: bus clock = 40 MHz*/
 |SCG_RCCR_DIVSLOW(0b10); /* DIVSLOW=2, div. by 3: SCG slow, flash clock= 26 2/3 MHz*/
 while (((SCG->CSR & SCG_CSR_SCS_MASK) >> SCG_CSR_SCS_SHIFT ) != 6) {}
 /* Wait for sys clk src = SPLL */
}
```

More code lmao

```
void SW3_button_init(PORT_Type *port){

        uint32_t port_index = 75;
        if(port == PORTA){
                port_index = 73;
        } else if (port == PORTB){
                port_index = 74;
        } else if (port == PORTC){
                port_index = 75;
        } else if (port == PORTD){
                port_index = 76;
        } else if (port == PORTE){
                port_index = 77;
        } else {
                port = PORTD;
        }

        PCC->PCCn[port_index] = PCC_PCCn_CGC_MASK;

        // Set PTC12 (SW3) as GPIO
        port->PCR[PTC12] = PORT_PCR_MUX(2);

        // Set PTC12 as input
        PTC->PDDR &= ~(1 << PTC12);

        // Enable pull-up resistor for SW3 button (active-low button)
        port->PCR[PTC12] |= PORT_PCR_PE_MASK | PORT_PCR_PS_MASK;
```

- This whole function is bad practice
- Can anyone guess the bad practices here?
- Or maybe what this function does?

More example files

- Now, I have specifically made myself as newbie as possible with this MC to demonstrate that *any* embedded experience can translate well to most boards
- Just like programming languages, the foundation is important
- Choose an example file you would like me to go over
- I will deny some files because they are not good for learning and some we do not have the equipment for
- Play around with some files and I can cover ones we think are interesting
- I can show some other work I have done on other MCs

Interested in embedded?

- Talk to me after meeting, can connect you with people and resources to get more experience
- A few events you can do for more experience
- Lend boards to you for personal learning
- We will support your learning

NASA SUITS Challenge

- 8-15 team members
- Undergrad + grad + faculty
- Amazing learning experience
- Go to Johnson Space Center, Houston, TX
- Test week: May 18-22, 2025

- NASA Spacesuit User Interface Technologies for Students (NASA SUITS) is a design challenge in which college students from across the country help design user interface solutions for future spaceflight needs.

ICPC!!!

- Teams of 3 members
- Undergrad & grad students
- Amazing learning experience
- Go to College of Charleston
- Regionals: November 16th
- Get outside and touch grass

International Collegiate Programming Contest is an algorithmic programming contest for college students. Teams of three, representing their university, work to solve the most real-world problems, fostering collaboration, creativity, innovation, and the ability to perform under pressure. Through training and competition, teams challenge each other to raise the bar on the possible. Quite simply, it is the oldest, largest, and most prestigious programming contest in the world.