Level 0x07

Structures
Pointers

Topics

- Events
- Makers
- C Stuff
  - Structures
  - Pointers
  - Combined!

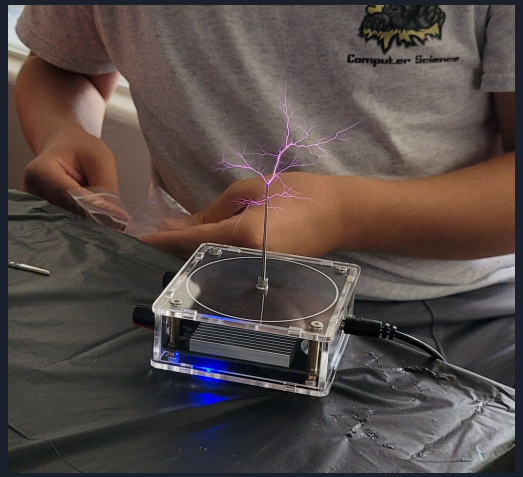
Upcoming Events

- SANS Holiday Hack Challenge - 2nd week of Dec
- TryHackMe's Advent of Cyber 2023 (Dec 1st - 25th)

- Eric Wastl's 2023 Advent of Code
  - 2015 started with 81 participants
  - 2022 has over 250,000 participants
  - How far can you get?
    - 2019 - Day 13
    - 2020 - All days!
    - 2021 - Day 21

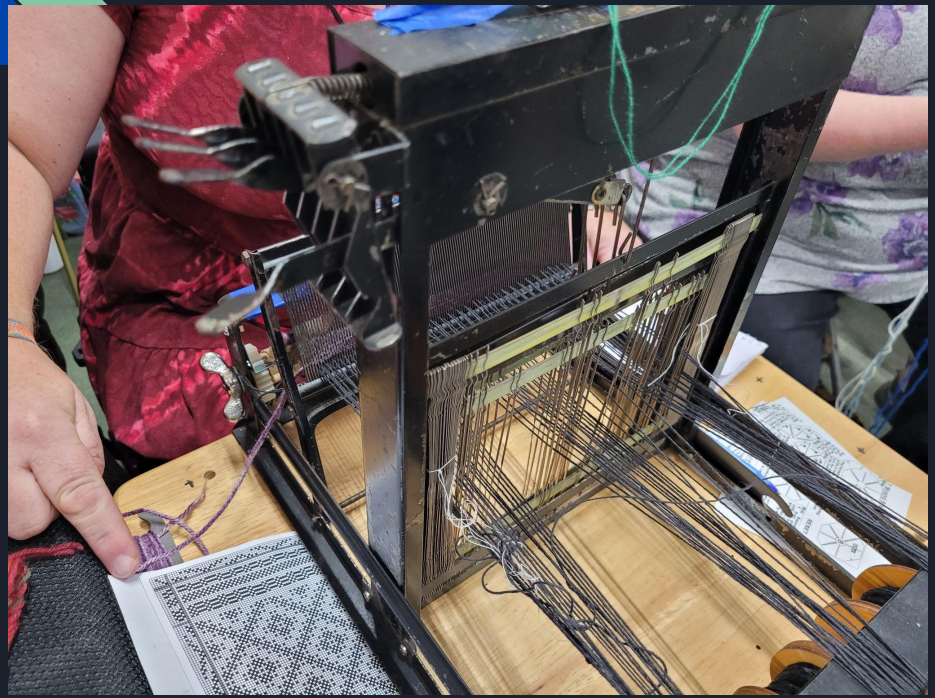
Central Florida Makers

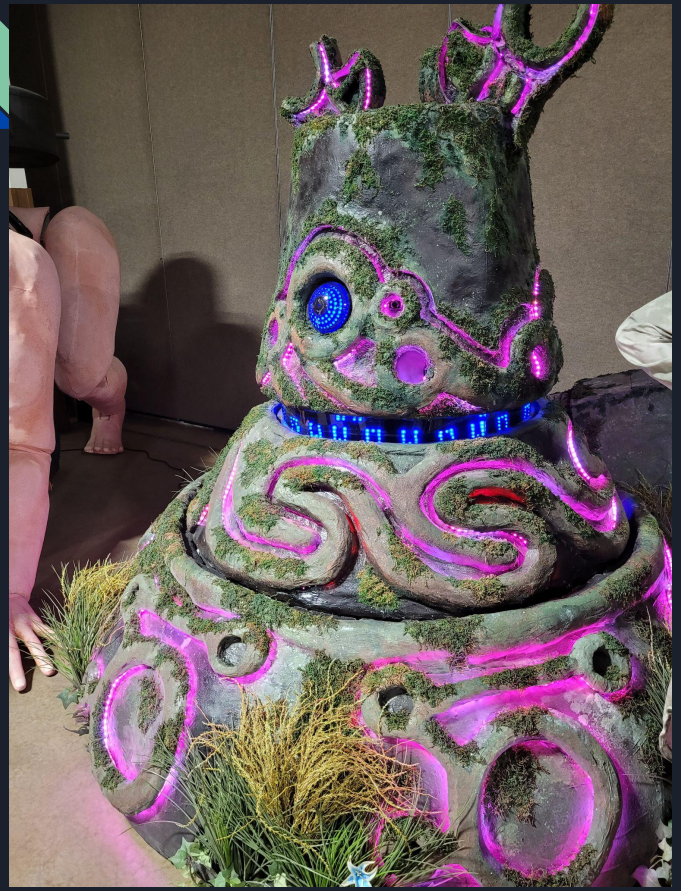
Tesla Coils

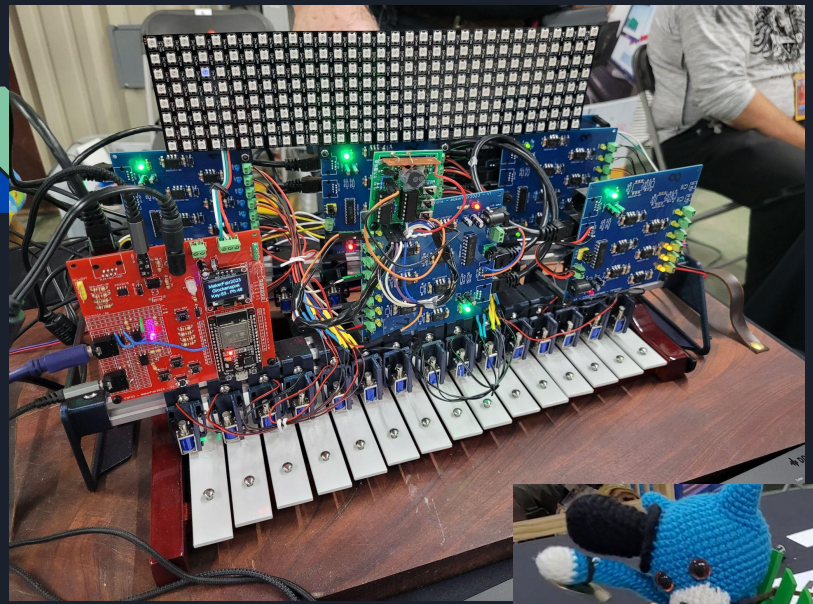


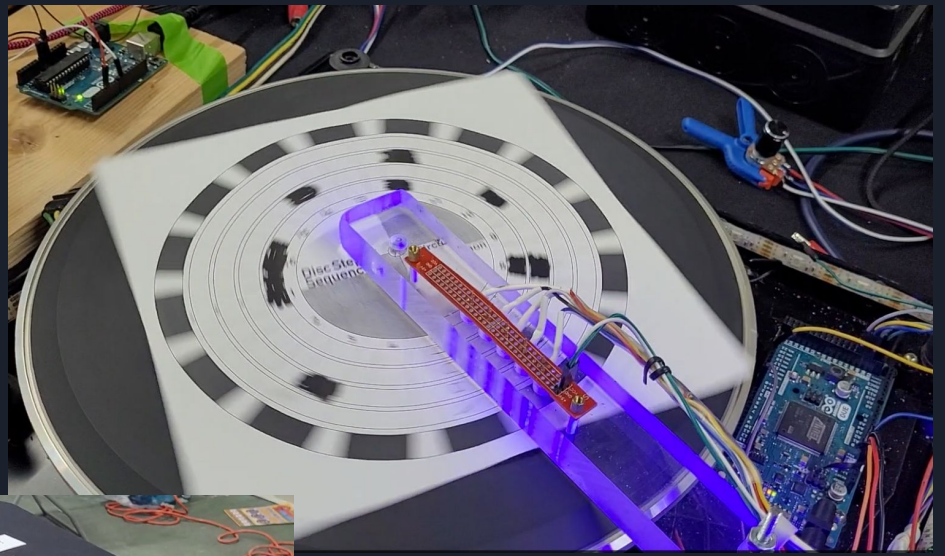
More Makers

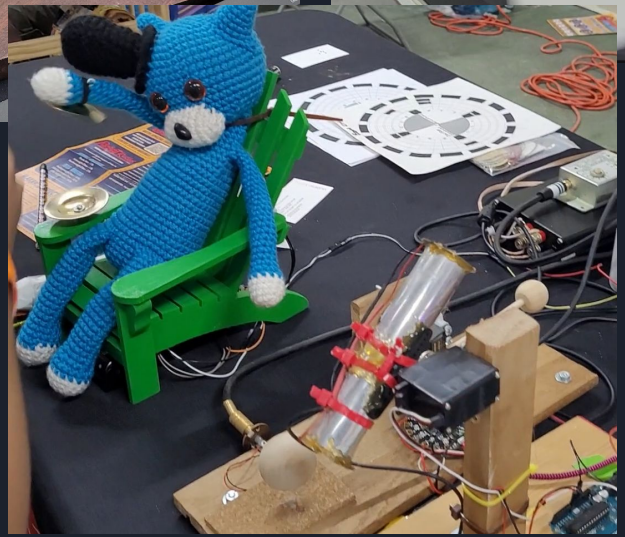
Wearable LEDs / circuits with conductive thread

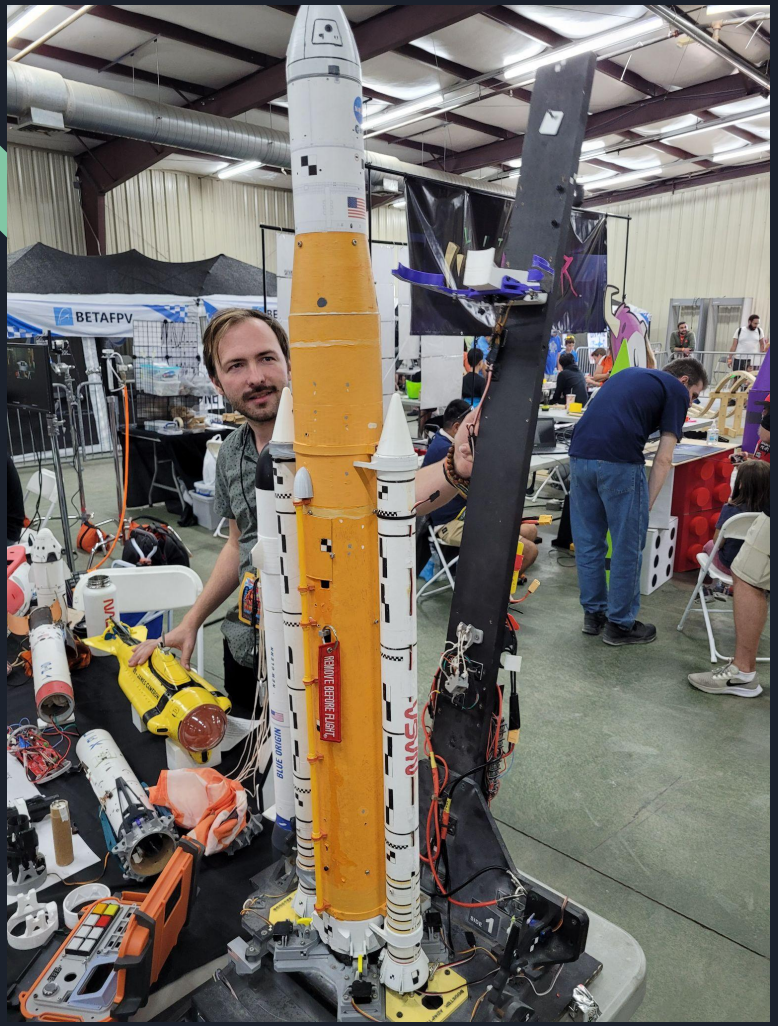


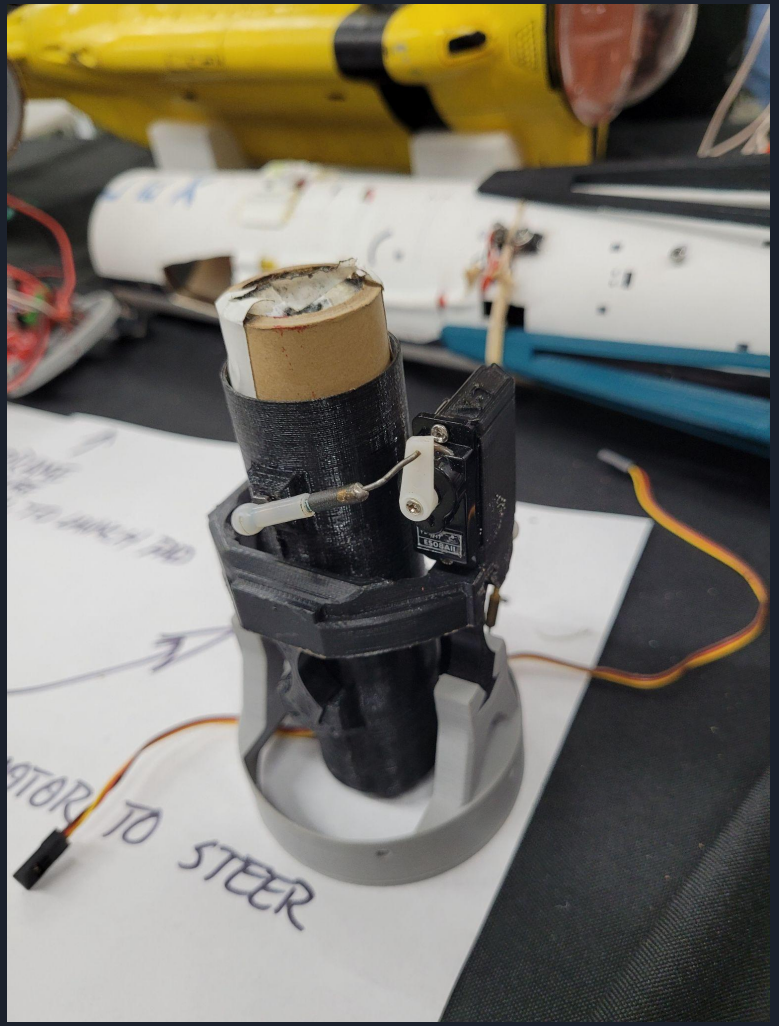








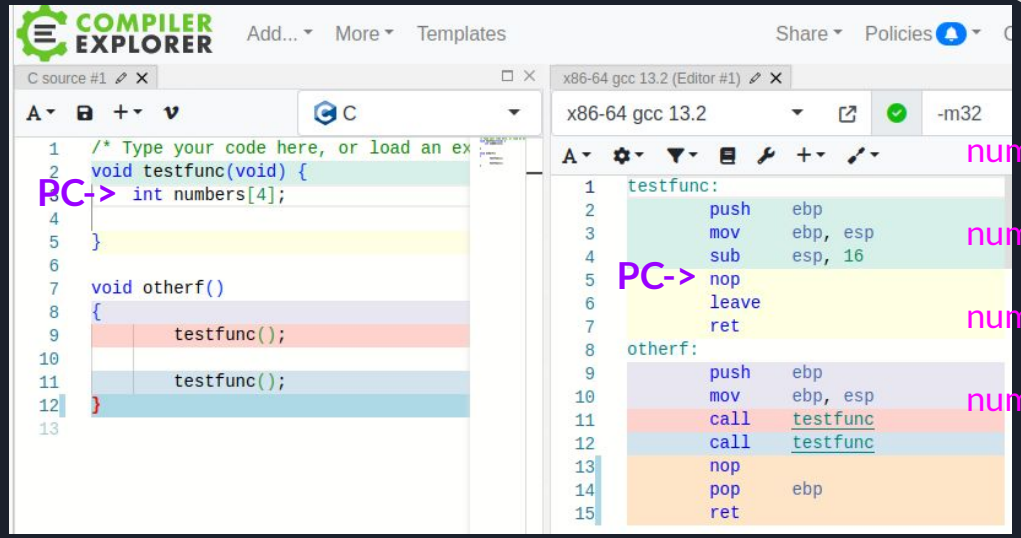

BETAFPV
REMOVE BEFORE FLIGHT!


TO STEER

Florida Tech Intro to Cyber CTF 1st Place

The Stack

SP moved up 0x10 or 16 bytes



Addresses	Memory Vals
0xf000 0030	
0xf000 0034	
0xf000 0038	
0xf000 003c	???
0xf000 0040	???
0xf000 0044	???
0xf000 0048	???
0xf000 004c	0xf000 0050
0xf000 0050	12

Structure

```
struct mathVector
{
  int angleDeg;
  int magnitude;
};

struct mathVector a;
a.angleDeg = 90;
a.magnitude = 2;

// Now I can pass a function very complicated parameters
```

Typedef Structure

```
typedef struct mathVectorStruct
{
  int angleDeg;
  int magnitude;
} MathVec;

MathVec a;
a.angleDeg = 90;
a.magnitude = 2;

// No idea why this isn't how they always work…
```

Pointers

```
int a = 1337;
int* b = &a;  // pointer b assigned the address of a

printf("Val of *b = %d\n", *b); // prints 1337

A = 999;

printf("Val of *b = %d\n", *b); // prints 999

int c = 42;
a = 100;
b = &c;

printf("Val of *b = %d\n", *b); // prints 42
```

Stack frame of last example

- Pointers are memory addresses

SP->

int a;

int* b;

int c;

BP->

Addresses	Memory Vals
0xf000 0030	
0xf000 0034	
0xf000 0038	
0xf000 003c	100
0xf000 0040	0xf000 0044
0xf000 0044	42
0xf000 0048	???
0xf000 004c	0xf000 0050
0xf000 0050	12

Structures and Pointers

```
typedef struct mathVectorStruct
{
  int angleDeg;
  int magnitude;
} MathVec;

MathVec a;
a.angleDeg = 90;
a.magnitude = 2;
MathVec* b = &a;

printf("angle = %d and mag = %d\n", b->angleDeg, b->magnitude);
```

Arrays of Structures

```
typedef struct MathVecStruct {
  int angleDeg;
  int magnitude;
  char name[0x20];
} MathVec;

  MathVec a[3] = { {90,  2, "first"},
                   {180, 3, "second"},
                   {0,   5, "third"}   };

  printf("Size of MathVec = %zd\n",sizeof(MathVec));

  for(int i = 0; i < 3; i++)
  {
    printf("Vec %s = %d @ %d deg\n", a[i].name, a[i].magnitude, a[i].angleDeg);
  }

// Size of MathVec = 40
// Vec first = 2 @ 90 deg
// Vec second = 3 @ 180 deg
// Vec third = 5 @ 0 deg
```

Arrays == Pointers...

```
#include <stdlib.h>  // malloc / free

// malloc gives you an address to memory, any size you want
MathVec* a = (MathVec*) malloc(userVar * sizeof(MathVec));

// ask user for data to populate array

for(int i = 0; i < userVar; i++)
{
  printf("Vec %s = %d @ %d deg\n",a[i].name, a[i].magnitude,
         a[i].angleDeg);
}

// I'm done with memory
free(a);
```

Links

: