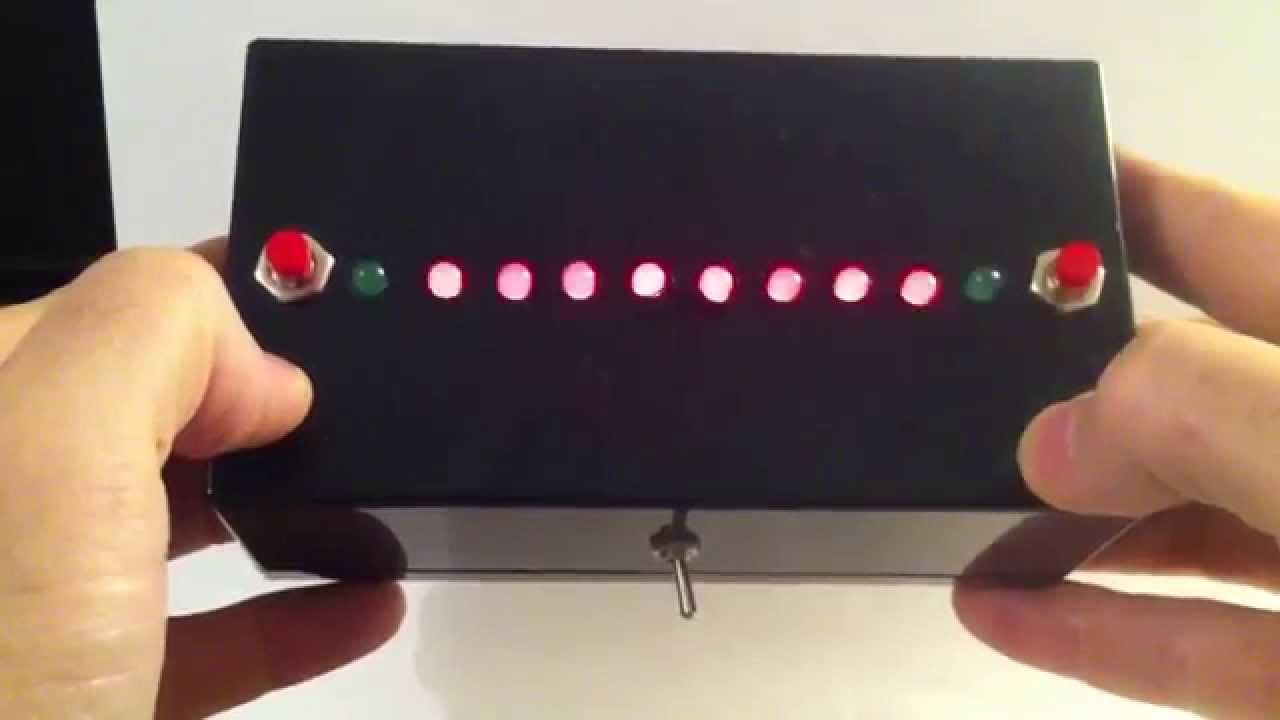
**NEW YORK CITY COLLEGE OF TECHNOLOGY**

THE CITY UNIVERSITY OF NEW YORK

**Department of Computer Engineering Technology**

*CET 3510 Microcomputers Systems Technology  
Section: E360***1K HACKADAY CONTEST**



**Table of Contents**

Official Rules……………………………………………………………………………………………………………………………………….pg 3

Roles Assigned…………………………………………………………………………………………………………………………………...pg 4

Potential Project Ideas……………………………………………………………………………………………………………………….pg 5

1-D Pong…………………………………………………………………………………………………………………………………………….pg 6

Materials Needed……………………………………………………………………………………………………………………………….pg 7

Initial Code Samples……………………………………………………………………………………………………………………………pg 8

**Official Rules**

* This contest is open to any microcontroller or microprocessor architecture.
* Projects must use 1 kB or less of code, including any initialized data tables, bootloaders, and executable code.
* Unavoidable Hardcoded bootloaders such as propeller, ESP8266, or i.mx series don't count toward the 1 kB limit. Entrants can use the bootloader code to start their processor up and get their own code executing. Once things are running though, any calls into bootloader or ROM tables count against you.
* Entrants must show how they came to the 1 kB calculation (map files, assembler/linker output, etc)
* Existing computer systems can be used (Example - Atari 2600) however, the size of any rom routines or data tables must be included in the 1 kB calculation.
* Mask Rom: If you are using a chip with masked rom code, you must include the size of any ROM routines you call in your final executable size calculation. If you wanted to write a program in SPIN for the Parallax Propeller, you’d need to include the SPIN interpreter ROM, which would put you over the 1 kB limit.
* Projects can use multiple processors as long as they are running the same 1 kB binary, or if the total size of the ROM being run is less than 1 kB. (Example - two micros running different binaries which are 512 bytes long each)
* Projects must fill out the description, details, components, and project logs.
* Projects must be open source. Source code, schematics, and board layouts for the project software must be either posted in the files section, or include a link to a public repository such as Github.
* Projects must use publicly accessible toolchains. Closed source toolchains (example - IAR or Keil) can be used as long as the demo versions will compile/assemble the contest code.

**Roles Assigned  
(not complete)**

**Project and Time Management:** Prof. Hamilton, Shiasia

**Documentation:** Nil, Elmer, Aayush

**Research & Development:** Emanuel, Patrick, Himu, Eric, Grisley

**Hardware:** Carlos, Himu

**Software (Coders):** Carlos(Flashing LEDs), Aldrick(Buttons w/ LEDs), Justin(Cycle LED & Charlieplexing?), Darius(Counters)

**Communications:** Or

**Quality Assurance:** Sam, Luis, William

?? -- Gyim,Emmanuel Jimenez,John

**Potential Project Ideas**

* Display
* Interface(s)/UI (user interface):
* Physical (button/s)
* Clap-on (sound activated/sound sensitive)
* Tiny machine
* "Useless" device
* CONTROLLER FOR...
* Power efficiency/power management
* Drones
* Mini projector
* Lights
* Smart house
* Voice recognition/text to speech (Echo)
* Launch apps
* Amazon Dash button
* Pill Minder
* Life Alert/Police call
* Noise filter (noise cancellation)
* Media server
* Enhanced Roku
* Random food ordering
* Real-time information
* 1-D Pong
  + - weather
    - radio report
    - transportation
    - market prices
    - Twitter

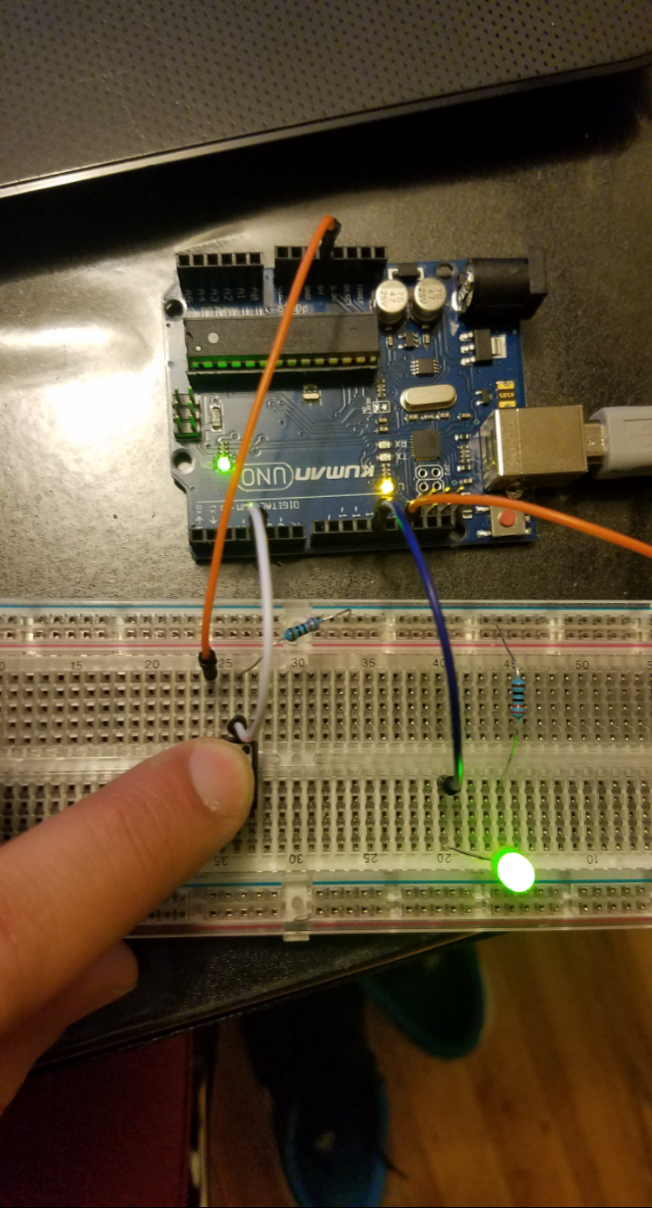
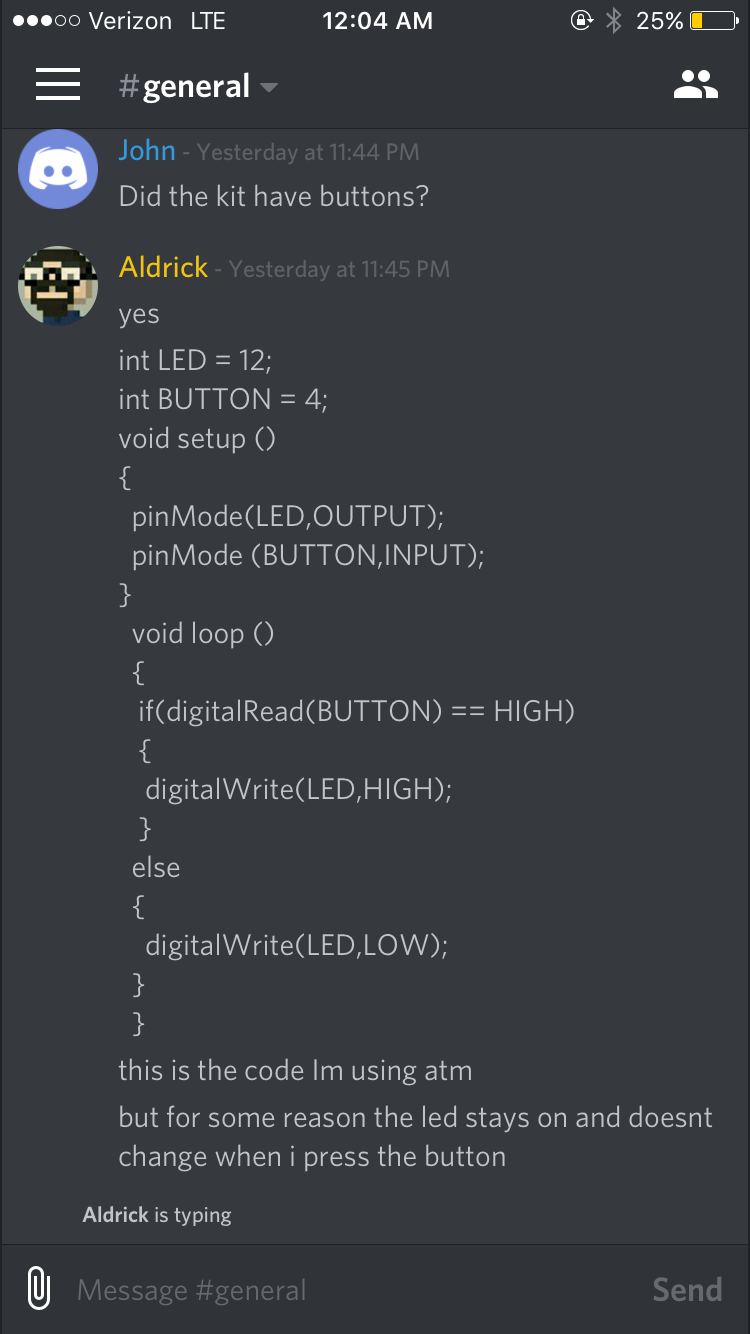
**1-D Pong**

As the class majority; we decided to build a 1-dimensional game of Pong, which will allow us to apply the skills we have learned in Microcomputers Systems Technology in regards to coding in assembly in different environments such as windows and Linux. The 1-dimensional pong is a classic game that requires the player to click one of the two buttons at the right time while the LEDs that are placed in series are used as a timer. The hardware of this game includes a **metal/wood/3d printed** casing which houses 11LEDs and two buttons all arranged in series with the two buttons on the ends, along with a power button. The players scores a point if and when the player clicks the button when the adjacent LEDs is lit; once the last LED is lit in the row the LEDs are then lit in the reverse order that they were initially lit; creating a “bouncing effect”. If the player manages to click the button in time the “ball” will start going back in the opponent’s direction and the cycle continues. To make this game challenging the speed at which the “ball” moves increases. If a player is unable to press the button in time, or presses it to early, their opponent is awarded a point. This is indicated by the LEDs adjacent to their button. After each point, their light will blink to signal the number of points they have. The game progresses until a player reaches five points, which is signaled by the winning player’s LED staying continuously lit. The game can be restarted by pressing off button for the power change. To complete the build, the components will be housed in a box.

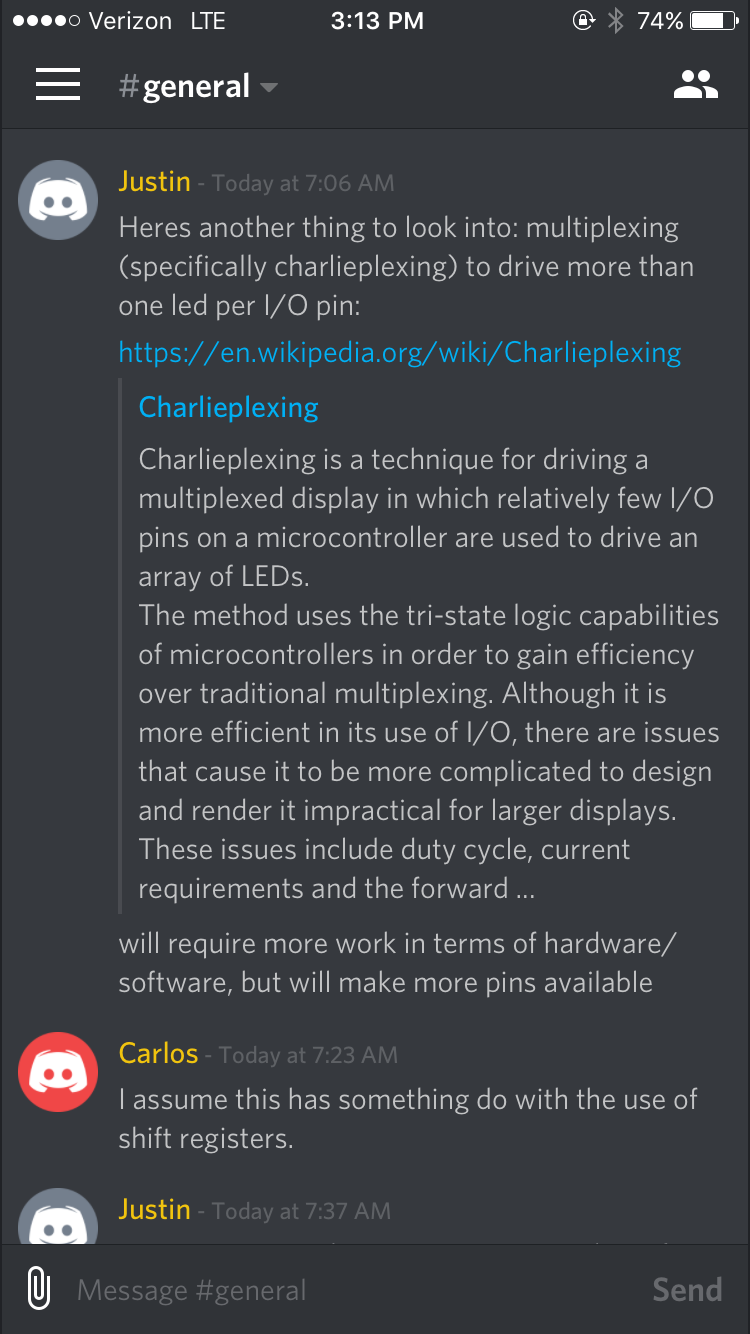
**Materials Required**

* Any version of Arduino Uno
* Around 9 green LED’s and 2 red LED’s --- **11 LEDs**
* 2 push buttons which act as the game controller
* 1 switch button as designated power button
* 2 7-segment counter displays
* Metal / Wood / 3d Printed Rectangular Box Enclosure
* Power Supply / Battery
* Breadboard
* Resistors
* Capacitors
* Sound - Buzzer??

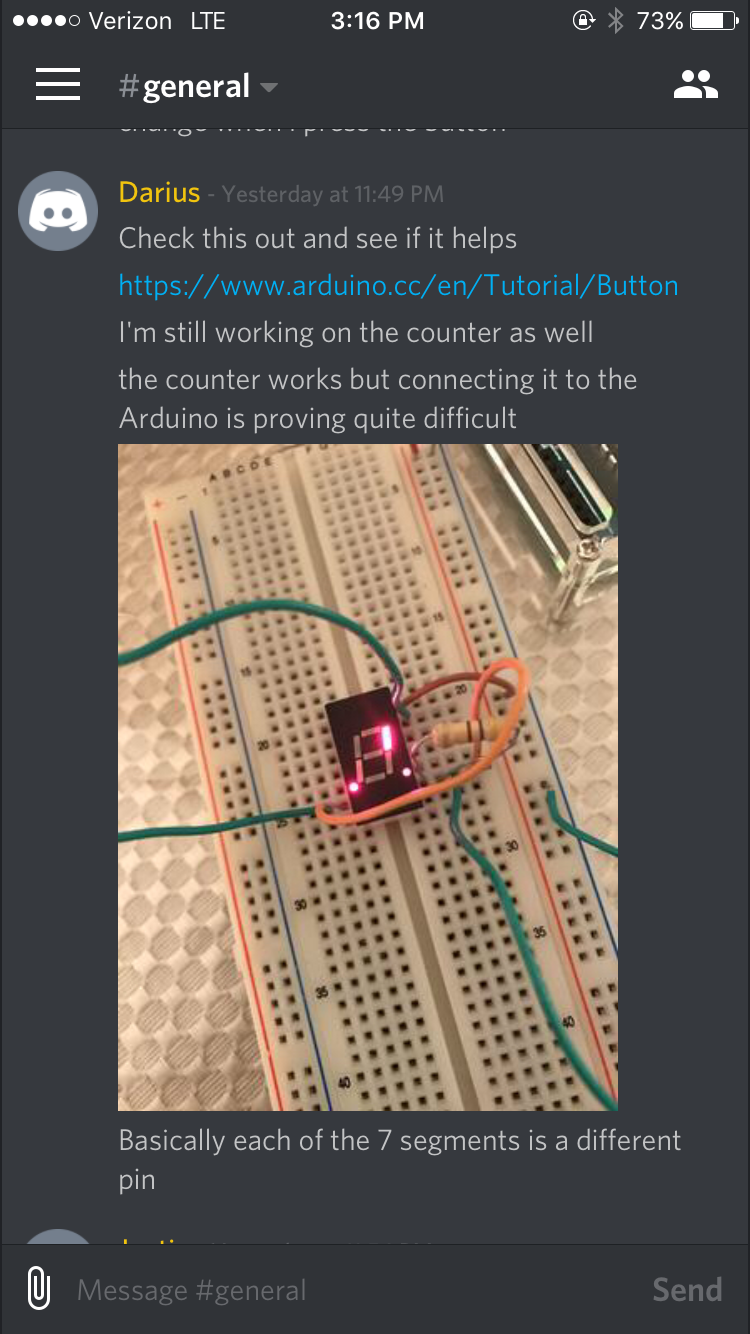
**Initial Code Samples**



Here one of the Coding group members, Aldrick, had trouble with getting an LED to work when a button is pressed. What he did to fix the problem was not related to the code, as a few other members had agreed that the code was not the problem. How he solved this problem was with a change in the circuit. Then he got the program working.



Here, Justin(part of the coding group) suggested that the team get into charlieplexing so that they could use the limited number of ports/pins on the arduino to their maximum capacity.



Here the coder Darius is working on the counter using a 7 segment display. He explains he is having difficulty getting the counter to work as we expect it to.

The code and video is done by aldrick.

Justin initially was the one who kick started this code, the code itself is written in AVR assembly as of Saturday 12:09 Am Dec 3rd. The video link is below and the code was uploaded to GitHub.

<https://cdn.discordapp.com/attachments/248984479744524289/254474294946430977/VIDEO0005.mp4>

Arduino IDE Code for led sliding back and forth:

int LED9 = 9;

int LED10 = 10;

int LED11 = 11;

int LED12 = 12;

int LED13 = 13;

int BUTTON = 4;

void setup ()

{

pinMode(LED9,OUTPUT);

pinMode (LED10,OUTPUT);

pinMode (LED11,OUTPUT);

pinMode (LED12,OUTPUT);

pinMode (LED13,OUTPUT);

pinMode (BUTTON,INPUT);

}

void loop ()

{

if(digitalRead (BUTTON) == HIGH){

digitalWrite (LED9,HIGH);

delay (100);

digitalWrite (LED10,HIGH);

delay (100);

digitalWrite (LED9,LOW);

delay (100);

digitalWrite (LED11,HIGH);

delay (100);

digitalWrite (LED10,LOW);

delay (100);

digitalWrite (LED12,HIGH);

delay (100);

digitalWrite (LED11,LOW);

delay (100);

digitalWrite (LED13,HIGH);

delay (100);

digitalWrite (LED12,LOW);

delay (100);

digitalWrite (LED13,LOW);

}

else{

digitalWrite (LED9,LOW);

digitalWrite (LED10,LOW);

digitalWrite (LED11,LOW);

digitalWrite (LED12,LOW);

digitalWrite (LED13,LOW);

}

}

The video link of it actually working <https://cdn.discordapp.com/attachments/248984479744524289/254487203244277764/20161203_005206.mp4>

Schematic for led sliding back and forth

