```
#include <LiquidCrystal.h>
#include <OneMsTaskTimer.h>

bool nukeing = 0;
int nukePin = PUSH2;
xy OldNukePosition;

void setupNuke()
{
  pinMode(nukePin, INPUT_PULLUP);
  nuke.active = 0;
  nuke.position.x = 16;
  nuke.position.y = 0;
  OldNukePosition = nuke.position;
  Serial.begin(9600);
  attachInterrupt(digitalPinToInterrupt(nukePin), nukeISR, FALLING);
}

void loopNuke()
{
  while(NukeThreadFlag == 0) //almost everything is on this clock
  {
    delay(10);
  }
  NukeThreadFlag = 0;

  //Serial.print("(Nuke Thread Wokring) ");

  if(PAS!=gameOver && PAS!=Gamestart && PAS!=GameInit) // if the game is not over and not at the starting screen
  {
    if(nukeing == 1 && nukeCount > 0 && nuke.active == 0) //the bottum has beem pressed and the player has resources left
    {
      nukeing = 0;
      nukeCount--;

      createNuke(); //logics here are similar to bonus

      Serial.print("Creating Nuke");
    }
    else if(nukeing == 1 && nukeCount <= 0 && nuke.active == 0)
    {
      Serial.print("Error: Not Enough Nukes");
    }

    if(nuke.active == 1)
    {
      advanceNuke(); //move the nuke across the screen
      deleteShield1(); //detect collision on each step
      deleteNuke(); //deleting the nuke when it is off the screen
    }

    nukeing = 0; //the flag should still be reset even if nothing happens
  }
}

void nukeISR()
{
  Serial.println("ISR - Nuke");
  nukeing = 1;
}


void createNuke()
{
  if(nuke.active == 0) //only runs when the nuke isn't already on the screen
  {
    nuke.position.x = HeroLocation.x + 1; //this might cause timing issues, but I'm too
```

```
      tired to fix it
      nuke.position.y = HeroLocation.y; //Set the nuke to appear one unit infront of the PC
      nuke.active = 1;
      OldNukePosition = nuke.position;

      lcd.setCursor(nuke.position.x, nuke.position.y);
      lcd.write(byte(4));
    }
  }

  void advanceNuke()
  {
    eraseNuke();
    drawNuke();
  }

  void eraseNuke()
  {
    lcd.setCursor(OldNukePosition.x, OldNukePosition.y); //these code are given
    lcd.print(" ");
  }

  void drawNuke()
  {
      nuke.position.x = nuke.position.x + 1; //march towards the right
      lcd.setCursor(nuke.position.x, nuke.position.y);
      lcd.write(byte(4));
      OldNukePosition = nuke.position;
  }

  void deleteNuke()
  {
    if(nuke.position.x < 0  || nuke.position.x > 15) //if the player wants to fire it off
    on the far right for some reason
      {
        eraseNuke();

        nuke.active = 0; //make the nuke inactive
        nuke.position.x = 16; //reset position
      }
  }

  void deleteShield1()
  {
    for(int i=0; i<obstcount; i++) //check every element of the array
    {
      if(((obsticals[i].position.x == nuke.position.x) && (obsticals[i].position.y ==
      nuke.position.y)) || ((obsticals[i].position.x == (nuke.position.x-1)) &&
      (obsticals[i].position.y == nuke.position.y))) //if the obstical meets the nuke,
      the obstical gets deleted
      {
        /*Serial.print("Deactivating element: ");
        Serial.println(i);*/

        eraseShield(i);

        if(obsticals[i].position.x <= 15 && nuke.position.x <= 15) //prevent collision
        off screen from messing with things
        {
          shieldsInUse--;
          Serial.print("Shield Count Updated: ");
          Serial.println(shieldsInUse);
        }

        obsticals[i].active = 0; //make the obstical inactive
        obsticals[i].position.x = 16; //reset position
        obsticals[i].checked = 0;
      }
    }
```

```
131     }
132
```