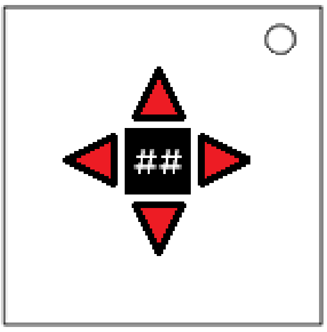
On the Subject of Not Red Arrows

Pointer.



On the module are 4 directional buttons, and a small screen in the middle.

If the buttons are not Red, or the screen in the middle is not showing a two digit number, you're looking at a different module.

On the screen is your Starting Number. The Up and Down buttons will change the number by 1, while the Left and Right will change the number by 10. To disarm the module, set the display to the number obtained from the instructions below, and tap the screen to submit. Submitting an incorrect number will generate a new Starting Number.

```
INNI:
int startNumber = [The Starting Number];
int key = startNumber;
int lastDigit = [Last Digit of Serial Number];
int tick = 0;
int totalCount = 0;
int correctNumber = 11;
int dunceCounter = 0;
bool trigger = TRUE;

START:
if (startNumber % 2 == 0) {
  JMP S_E;
} else {
  JMP S_O;
}

S_E:
if (key % 10 == lastDigit) {
  trigger == FALSE;
  key = key - lastDigit;
} else if (key % 10 > lastDigit) {
  key = (lastDigit + 2) * 7 ;
} else {
  key = key + lastDigit;
}
JMP CNT_OB;
```

S_O:

```
if (key % 10 == lastDigit) {
 key = (2 * key) % 100;
} else if (key % 10 > lastDigit) {
 key = (lastDigit - 2) * 5 ;
 trigger = FALSE;
} else {
 key = 50 - key;
}
JMP CNT_OB;
```

CNT_OB:

```
int n = key, cycleCount = 0, a;
while(n > 0)
    {
      a = n % 2;
      if(a == 1)
      {
        count++;
      }
      n = n / 2;
    }
totalCount = totalCount + count;
if (tick > 2) {
 JMP FORK_A;
 }
if (cycleCount % 2 == 0) {
 JMP O_V;
} else {
 JMP O_D;
}
JMP FAILSAFE;
```

O_V:

```
if (trigger == TRUE) {
 trigger == FALSE;
 key = key * (cycleCount + 1)
} else {
 key = key + cycleCount;
}
tick++;
JMP CNT_OB;
```

```
O_D:
if (trigger == TRUE) {
 key = key + (lastDigit * cycleCount);
} else {
 trigger == FALSE;
 key++;
}
tick++;
JMP CNT_OB;

FORK_A:
int x = 32;
x = x - totalCount
if (trigger == TRUE) {
 key++;
 JMP A_TR;
} else {
 key = key - 10;
 JMP A_FL;
 }

A_TR:
int atr = (cycleCount * lastDigit) + 3;
for (int i = 0; i < x; i++) {
 if (startNumber % atr == 0) {
  key = key + atr;
  continue;
 }
 atr++;
 key = key + (atr - i)
}
JMP PR_CH;

A_FL:
int afl = (totalCount - lastDigit) + 4;
for (int i = 0; i < x; i++) {
 if (startNumber % afl == 0) {
  key = key + afl;
  continue;
 }
 afl++;
 key = key + (afl - i)
}
JMP PR_CH;
```

PR_CH:
```
int q, r = 0, flag = 0;
  r = key/2;
  if (n == 0 || n== 1) {
   JMP KEY_CMP;
  } else {
   for (q = 2; q <= r; q++) {
    if ( key % r == 0 ) {
     JMP KEY_CMP;
     flag = 1;
     break;
    }
   }
   if (flag == 0) { JMP KEY_PR };
  }
}
}
JMP FAILSAFE;
```

KEY_CMP:
```
int b = 0;
int v = x % lastDigit;
v = v + 5;
        for (int i = 0; i < v; i++) {
            b = b + i;
            key = key + (totalCount % b)
        }
JMP FORK_B
```

KEY_PR:
```
int b = 2;
int v = totalCount % lastDigit;
v = v + 2;
        for (int i = 0; i < v; i++) {
            b = b + i;
            key = key + (x % b)
        }
JMP FORK_B
```

FORK_B:

```
if (trigger == TRUE && flag == TRUE) {
 JMP ANS_A;
} else if (trigger == TRUE) {
 JMP ANS_B;
} else if (flag == TRUE) {
 JMP ANS_C;
} else {
 JMP ANS_D;
 }
```

ANS_A:

```
if (key % 10 == lastDigit) {
 correctNumber = correctNumber * (key - totalCount);
} else if (key % 10 > lastDigit) {
 correctNumber = (2 * key) - cycleCount;
} else {
 correctNumber = key;
}
JMP LAS;
```

ANS_B:

```
if (key % 10 == lastDigit) {
 correctNumber = (key + 11) - totalCount;
} else if (key % 10 > lastDigit) {
 correctNumber = key + 2;
} else {
 correctNumber = correctNumber + key;
}
JMP LAS;
```

ANS_C:

```
if (key % 10 == lastDigit) {
 correctNumber = key - correctNumber;
} else if (key % 10 > lastDigit) {
 correctNumber = x + lastDigit;
} else {
 correctNumber = correctNumber + totalCount;
}
JMP LAS;
```

ANS_D:

```
if (key % 10 == lastDigit) {
 correctNumber = correctNumber + totalCount;
} else if (key % 10 > lastDigit) {
 correctNumber = key - 21;
} else {
 correctNumber = correctNumber * lastDigit;
}
JMP LAS;
```

LAS:

```
correctNumber = Math.abs(correctNumber) % 100;
return;
```

FAILSAFE:

```
/If you somehow ended up here, you have done something very, very wrong.
If you can't remember where might have you gone wrong, you might as well
restart the entire thing. From the top./
dunceCounter++;
JMP INNI;
```