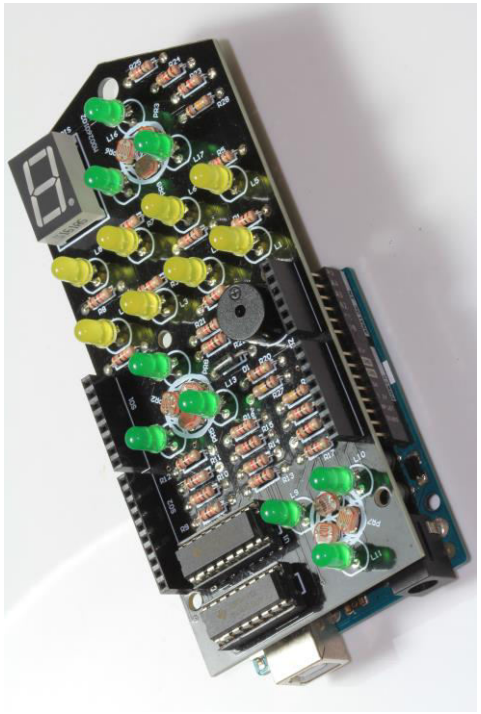
# LASER TARGET SHIELD

(ARDUINO COMPATIBLE)
PRODUCT CODE: M00270063



FEATURE:

- This can be played together with the laser pen buying in stationary store.
- Three sets of target
- One seven segment display and eight individual LED for timing and marking, the actual function is set by player on Sketch.
- Two IC 74HC595 for seven segment display and eight individual LED
- One buzzer
- One sample Sketch is attached.
- Assembly is needed by player.
- Requires 1 Arduino UNO (not included).

**READ BEFORE INSTALLATION:**

- Put the component on the side of screen printing and solder on the back of PCB without printing.
- On component, longer leg is "+".
- Do not put the LED to very bottom, just install as Figure 2.
- Do not connect this shield into the Arduino when downloading the Sketch.

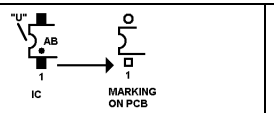


FIGURE 1

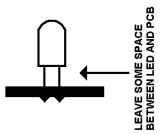


FIGURE 2

FIGURE 3

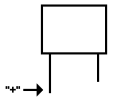


FIGURE 4

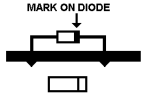


FIGURE 5

**DESCRIPTION:**

The circuit design is based on the Arduino UNO. Of course, this can be used on other broad if the pin location is matched. If this is not matched, just route this yourself.

From the circuit diagram, you can see that the circuit is very simple even there is totally 65 pieces of component. This mean most action and reaction of electronics component are mainly controlled by the Sketch. Maybe only two things are needed to explain;

RESPONSE OF LDR

The resistance of LDR would decrease when laser or more light hit on this. I use PR1, PR4 and PR7 as example. If PR7 is hit, resistance of PR7 get down and voltage level at "a0" also get down. The UNO can measure the voltage at "A0" and decide if there is any laser hit on this.

74HC595

If you want to fully understand this IC, this is better you search the specification on the web. But if you do not want to waste the time on learning, you can see this as black box and just use the code. The function of this IC is just to store and show the value for seven segment display and L1 to L8. By using this method, the UNO can use lesser pin and have more free time to take care other things.

We have done a Sketch for simple laser target, but you can make a more interesting game by writing a more complex Sketch. The below is the suggestion.

1. There are three sets of target. But at one moment, only one target is allowed to be hit. The flashing of LED L9 to L17 would tell the player which one is the right target at that moment. If wrong target is hit, the mark would be deducted.
2. Let one sets of target as starting/stopping button for new game.
3. The buzzer works as timer and the beeping rate become faster and faster when the time is passing.
4. At one moment, only hit one sets of target would get mark but nobody know which one is the right one.
5. Even there is only one sets of seven segment display and can only store the value from 0 to 9, but you can use flashing of seven segment display to indicate 10 to 19, different flashing rate to indicate the range of 20 to 29. Or different beeping rate of buzzer when hit to indicate the range of 30 to 29. Or other different way. The same things can be done on L1 to L8.

There is too much possible rule and way for this game, this is freely designed by the player on writing the Sketch.

**INSTALLATION:**

Just install the component to the PCB M00260102 according to below table.

| ITEM | SYMBOL ON PCB | DESCRIPTION | OUTLOOK | DIRECTION IS IMPORTANT? |
|---|---|---|---|---|
| 1 to 25 | R1 to R25 | RESISTOR, 1K ohms | BROWN, BLACK, RED | NO |
| 26 to 28 | R26 to R28 | RESISTOR, 100K ohms | BROWN, BLACK, YELLOW | NO |
| 29 to 37 | PR1 to PR9 | LDR | FIGURE 3 | NO |
| 38 to 39 | U1 to U2 | DIP 16 SOCKET | 16 LEGS | NO |
| 40 | D1 | DIODE, IN4001 | FIGURE 5 (MOSTLY BLACK) | FIGURE 5 |
| 41 | DIS | COMMON CATHODE SEVEN-SEGMENT DISPLAY | NUMBER "8" | NOTE 1 |
| 42 to 49 | L1 to L8 | LED | YELLOW | YES |
| 50 to 58 | L9 to L17 | LED | GREEN | YES |
| 59 | BZ | BUZZER | FIGURE 4 | YES |
| 60 | SO1 | STACKABLE HEADER – 8PIN | LONG 8 PIN | NO |
| 61 | SO2 | STACKABLE HEADER – 10PIN | LONG 10 PIN | NO |
| 62 | SO3 | STACKABLE HEADER – 6PIN | LONG 6 PIN | NO |
| 63 | SO4 | STACKABLE HEADER – 8PIN | LONG 8 PIN | NO |
| 64 | ON THE TOP OF ITEM 38 | IC, 74HC595 | 16 LEGS | FIGURE 1 |
| 65 | ON THE TOP OF ITEM 39 | IC, 74HC595 | 16 LEGS | FIGURE 1 |

NOTE 1. The direction is right if the dot on the component "SEVEN-SEGMENT DISPLAY" match the dot on the PCB.

**OTHER:**

TOO STRONG THE ENVIRONMENT LIGHT

If you play this shield under a very bright condition, this may reach the maximum detecting ability of the sensor. Any more light such as laser hit on this will have no response.

TOO WEAK THE LASER

After you finish the assembly of the kits but there is no any response (I assume that all your soldering is perfect) when you hit the target. The problem is maybe due to this line (I use PR1PR4PR7 sensors as example).

```
PR1PR4PR7 = analogRead(0); if((va0000-PR1PR4PR7)>70) {shoot0();} // This hit PR1PR4PR7?
```

I have set "70" as triggered level when target is hit by laser. If your laser is too weak (Maybe no battery), the difference between hit or no hit is maybe less than "70". In this condition, There is two solutions.

1. You must set the value less than "70". Setting this value is also very interesting.
2. You can write your own Sketch to find out the right value other than "70" when turning on this game everytime. The concept is that you hit the target, then the Arduino get out the right value but this only do one time. In our Sketch, you can see there is also some automatic calbration code. Calbration() is to detect the lightness of place you play at every round of game.

**CIRCUIT DIAGRAM:**

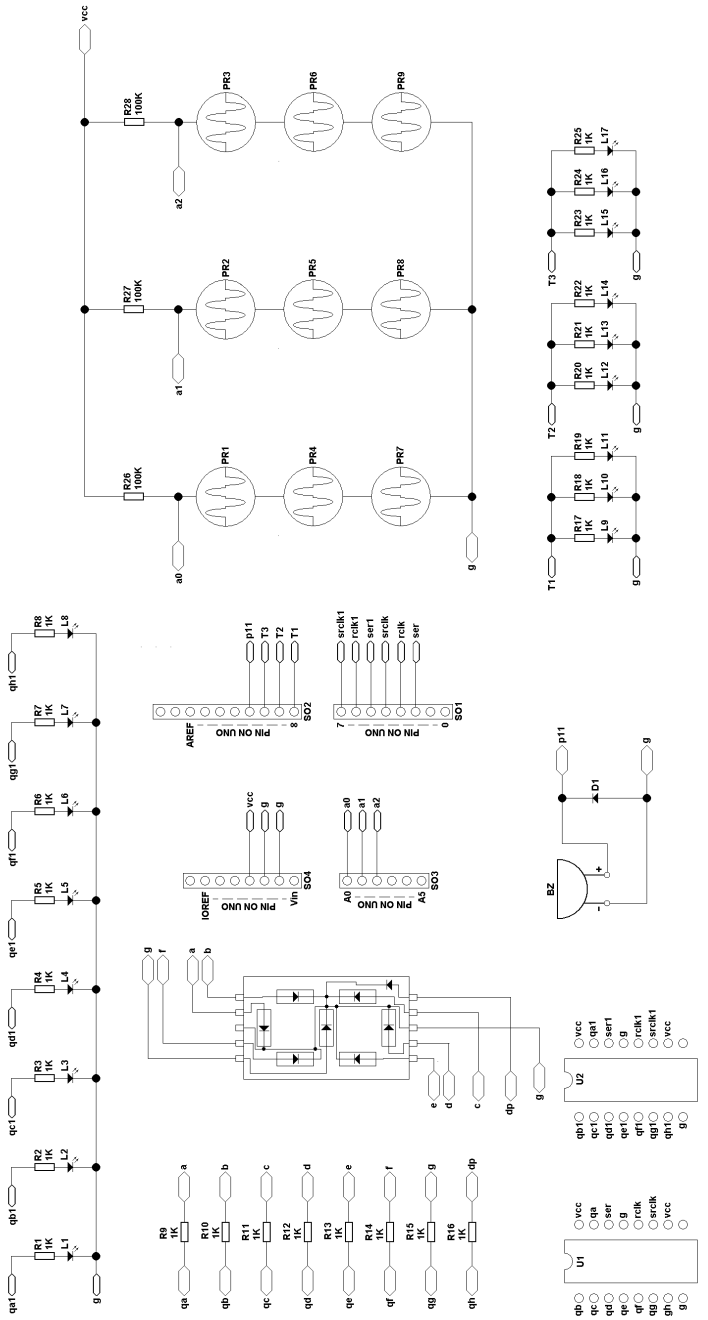



FIGURE 6

**SKETCH:**

```
/*
  * The below code is only one kind of playing mode. When you hit one of three target,
  * the buzzer would beep and certain LED would flash for few time. Also,
  * the seven segment display would add "1". On the other hand, L1 to L8 work as timer.
  * This would light off one by one when this passed certain period of time.
  * When you hit nine time of target or all the L1 to L8 are off, the game is ended.
  * After the end of game, this would do one time of calbration and start again.
  */

/* Pin declaration for L9 to L17 */
int L9L10L11 = 8; int L12L13L14 = 9; int L15L16L17 = 10;

/* Pin declaration for Buzzer */
int BZ = 11;

/* Pin declaration for two IC 74HC595 */
int SRCLK = 4;      int RCLK = 3;    int SER = 2;
int SRCLK1 = 7; int RCLK1 = 6; int SER1 = 5;

/* Variable for nine LDR sensor */
int PR1PR4PR7; int PR2PR5PR8; int PR3PR6PR9;

/* Variable for calbration() */
int va00[100]; int va11[100];
int va22[100];
long va000; long va111;
long va222;
long va0000; long va1111;
long va2222;

/* Variable for two function, sevenSegment(int segNumber) and ledLight(int ledNumber). */
byte segNumber; byte ledNumber;

/* Variable for the timer of this shield */
int timer;

/* Setting up the pin definition and do one time of calbration */
void setup() {

   pinMode(L9L10L11, OUTPUT);    pinMode(L12L13L14, OUTPUT);
   pinMode(L15L16L17, OUTPUT); pinMode(BZ, OUTPUT);
   pinMode(SRCLK, OUTPUT);          pinMode(RCLK, OUTPUT);
   pinMode(SER, OUTPUT);            pinMode(SRCLK1, OUTPUT);
   pinMode(RCLK1, OUTPUT);          pinMode(SER1, OUTPUT);
   calbration();

}

/* The loop() is to detect if the player has hitted the target. */
void loop() {

      for(int i; i<10000; i++) {
        PR1PR4PR7 = analogRead(0); if((va0000-PR1PR4PR7)>70) {shoot0();} // This hit PR1PR4PR7?
        PR2PR5PR8 = analogRead(1); if((va1111-PR2PR5PR8)>70) {shoot1();} // This hit PR2PR5PR8?
        PR3PR6PR9 = analogRead(2); if((va2222-PR3PR6PR9)>70) {shoot2();} // This hit PR3PR6PR9?

/* Timer for the game, this is shown on L1 to L8 */
      timer = timer + 1;
      if(timer == 10000) {
        if(ledNumber != 0) {
          ledNumber = ledNumber-1;
          ledLight(ledNumber);
        }
        else {
          endOfGame(); // Time is up.
        }
        timer = 0;
      }
   }

}
```

**SKETCH:**

```
/* What would happen when the player hit the target PR1PR4PR7. */
void shoot0() {

   segNumber = segNumber + 1;
   sevenSegment(segNumber);
   digitalWrite(L9L10L11, HIGH); digitalWrite(BZ, HIGH); delay(100);
   digitalWrite(L9L10L11, LOW);    digitalWrite(BZ, LOW);    delay(100);
   digitalWrite(L9L10L11, HIGH); digitalWrite(BZ, HIGH); delay(100);
   digitalWrite(L9L10L11, LOW);    digitalWrite(BZ, LOW);    delay(100);
   digitalWrite(L9L10L11, HIGH);
   if(segNumber == 9) { // The target has been hitted 9 times before the time is up.
      endOfGame();
   }
   delay(800);

}

/* What would happen when the player hit the target    PR2PR5PR8. */
void shoot1() {

   segNumber = segNumber + 1;
   sevenSegment(segNumber);
   digitalWrite(L12L13L14, HIGH); digitalWrite(BZ, HIGH); delay(100);
   digitalWrite(L12L13L14, LOW);    digitalWrite(BZ, LOW);    delay(100);
   digitalWrite(L12L13L14, HIGH); digitalWrite(BZ, HIGH); delay(100);
   digitalWrite(L12L13L14, LOW);    digitalWrite(BZ, LOW);    delay(100);
   digitalWrite(L12L13L14, HIGH);
   if(segNumber == 9) { // The target has been hitted 9 times before the time is up.
      endOfGame();
   }
   delay(800);

}

/* What would happen when the player hit the target PR3PR6PR9. */
void shoot2() {

   segNumber = segNumber + 1;
   sevenSegment(segNumber);
   digitalWrite(L15L16L17, HIGH); digitalWrite(BZ, HIGH); delay(100);
   digitalWrite(L15L16L17, LOW);    digitalWrite(BZ, LOW);    delay(100);
   digitalWrite(L15L16L17, HIGH); digitalWrite(BZ, HIGH); delay(100);
   digitalWrite(L15L16L17, LOW);    digitalWrite(BZ, LOW);    delay(100);
   digitalWrite(L15L16L17, HIGH);
   if(segNumber == 9) { // The target has been hitted 9 times before the time is up.
     endOfGame();
   }
   delay(800);

}

/* This function is for displaying the seven segment display. */
void sevenSegment(int segNumber) {

   byte zero    = B11111100; byte one      = B01100000; byte two      = B11011010;
   byte three = B11110010; byte four    = B01100110; byte five    = B10110110;
   byte six      = B10111110; byte seven = B11100000; byte eight = B11111110;
   byte nine    = B11110110; byte full    = B00000000;
   byte temp;
   if(segNumber == 0)    {temp = zero;}    if(segNumber == 1) {temp = one;}
   if(segNumber == 2)    {temp = two;}      if(segNumber == 3) {temp = three;}
   if(segNumber == 4)    {temp = four;}    if(segNumber == 5) {temp = five;}
   if(segNumber == 6)    {temp = six;}      if(segNumber == 7) {temp = seven;}
   if(segNumber == 8)    {temp = eight;} if(segNumber == 9) {temp = nine;}
   if(segNumber == 10) {temp = full;}
   for(int l = 0; l<8; l++) {
      if(bitRead(temp, l)) {
         digitalWrite(SER, HIGH);
      }
      else {
         digitalWrite(SER, LOW);
      }
// When this go from LOW to HIGH, this let one bit of data to be moved forward to next stage of shift register.
      digitalWrite(SRCLK, HIGH);
      digitalWrite(SRCLK, LOW); // This let "go from LOW to HIGH" of SRCLK can run again at next round.
   }
   digitalWrite(RCLK, HIGH); // When this go from LOW to HIGH, shift-register data is stored in the storage register.
   digitalWrite(RCLK, LOW); // This let "go from LOW to HIGH" of RCLK can run again at next round.

}
```

```
/* This function is for displaying L1 to L8. */
void ledLight(int ledNumber) {

   byte zero    = B00000000; byte one    = B10000000; byte two = B11000000; byte three = B11100000;
   byte four    = B11110000; byte five = B11111000; byte six = B11111100; byte seven = B11111110;
   byte eight = B11111111;
   byte temp;
   if(ledNumber == 0) {temp = zero;}    if(ledNumber == 1) {temp = one;}
   if(ledNumber == 2) {temp = two;}      if(ledNumber == 3) {temp = three;}
   if(ledNumber == 4) {temp = four;}    if(ledNumber == 5) {temp = five;}
   if(ledNumber == 6) {temp = six;}      if(ledNumber == 7) {temp = seven;}
   if(ledNumber == 8) {temp = eight;}
   for(int l = 0; l<8; l++) {
      if(bitRead(temp, l)) {
         digitalWrite(SER1, HIGH);
      }
      else {
         digitalWrite(SER1, LOW);
      }
// When this go from LOW to HIGH, this let one bit of data to be moved forward to next stage of shift register.
      digitalWrite(SRCLK1, HIGH);
      digitalWrite(SRCLK1, LOW); // This let "go from LOW to HIGH" of SRCLK1 can run again at next round.
   }
   digitalWrite(RCLK1, HIGH); // When this go from LOW to HIGH, shift-register data is stored in the storage register.
   digitalWrite(RCLK1, LOW); // This let "go from LOW to HIGH" of RCLK1 can run again at next round.

}

/* What would be done when the game has finished one round due to two condition. */
void endOfGame() {

   digitalWrite(L9L10L11, LOW);    digitalWrite(L12L13L14, LOW);
   digitalWrite(L15L16L17, LOW); digitalWrite(BZ, LOW);
   delay(4000);
   calbration();

}

/*
  * The propose of calbration() is to adjust the sensitivity
  * of nine LDR according to lightness of place you place this shield.
  */
void calbration() {

   digitalWrite(L9L10L11, HIGH);    digitalWrite(L12L13L14, HIGH);
   digitalWrite(L15L16L17, HIGH);
   segNumber = 10;
   ledNumber = 0;
   sevenSegment(segNumber);
   ledLight(ledNumber);
   delay(500);

/* Take 100 sets of data for calbration. */
   for (int i = 0; i<100; i++) {
      PR1PR4PR7 = analogRead(0);    PR2PR5PR8 = analogRead(1);
      PR3PR6PR9 = analogRead(2);
      va00[i] = PR1PR4PR7;    va11[i] = PR2PR5PR8;
      va22[i] = PR3PR6PR9;
      va000 = va00[i] + va000;
      va111 = va11[i] + va111;
      va222 = va22[i] + va222;
   }

/* The average of 100sets of data on above */
   va0000 = va000/100;    va1111 = va111/100;
   va2222 = va222/100;

   va000 = 0;    va111 = 0;
   va222 = 0;
   segNumber = 0;
   ledNumber = 8;
   sevenSegment(segNumber);
   ledLight(ledNumber);

}
```