Materials

- 8 red laser pointers <3 mW
  The cheap ones are usually easier to dismantle, but you should get a few spare ones, in case.
  We found the less expensive ones were on eBay.
- 2 welding mount
  You should find this around 5 or 6 euros each.
  If you find any other way to mount the lasers please let us know.
- 1 mending plates
  With 4 holes. We found 100mm was a good size.
- 4 corner plates
  30mm
- Bolts
  We use 4×10mm
- 1 Arduino pro-mini
  We just like this one because it is so small. And also the cheapest (less than 15 euro on sparkfun.com)
- FTDI cable or FTDI breakout
  About as expensive as the Arduino also at sparkfun.
- electric cable 0,5mm^2
- 3 Battery holder + 3 AA Batteries.

Tools

- Soldering iron
- Cutting pliers
- Pliers
- Screwdriver
- Wrench
- File

Steps

Step 1

Dismantling the laser pointers

Unscrew the battery compartment, and take them out.

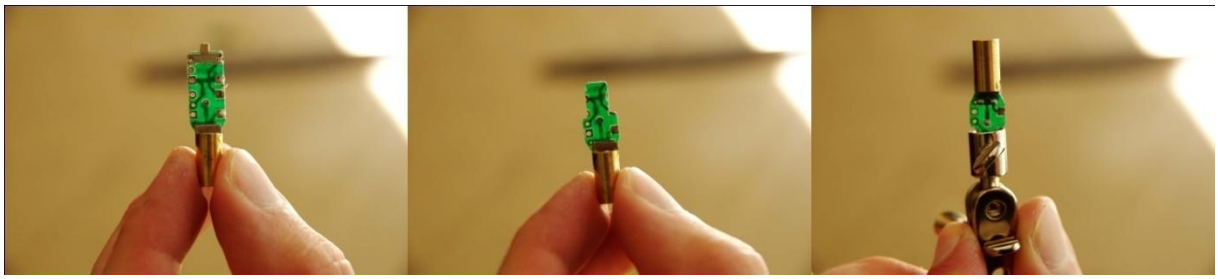
Pull out the top cap which should hold the laser and a small printed circuit board.

Take of the cap, and unsolder the unwanted elements. (on ours we had two LEDs and two push button)

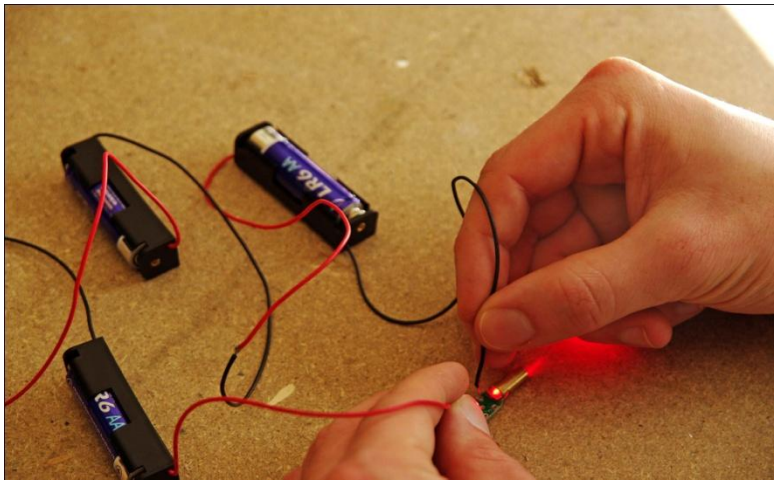
Step 2

Get the lasers working.

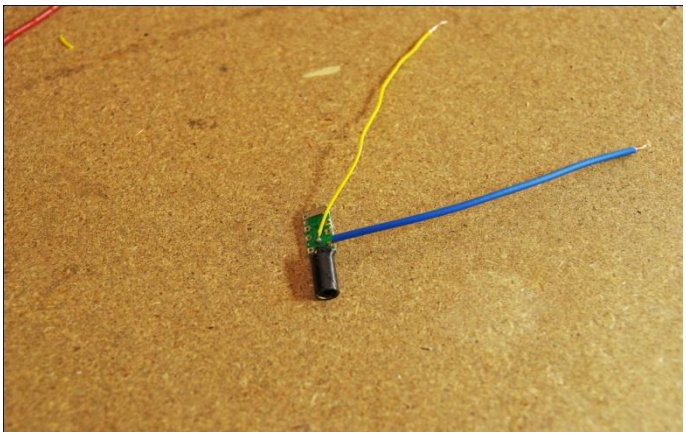
File the circuit of 4 lasers as shown.



Solder in series three 1,5V batteries.
Locate the positive and the negative pole on the circuit. The metallic part around the laser is positive, but it will be much more practical to use the one on the circuit.



Solder two cables of about 10cm long on each pole.
Isolate the metallic piece with scotch-tape.



Step 3

The mount.

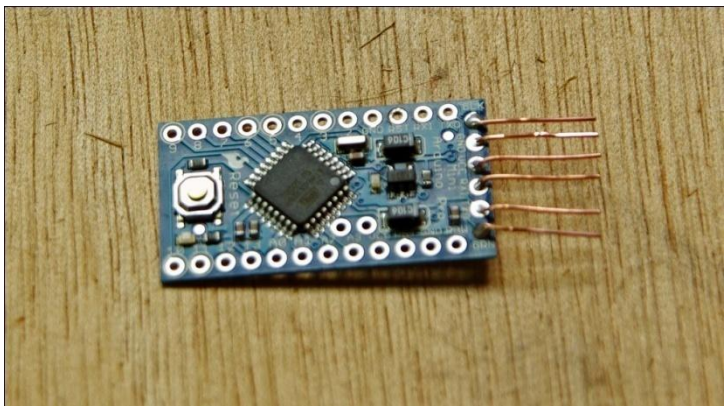
Here we tried deferent ways, but, for now, this one seemed the most practical.
Dismantle the welding mounts and keep the articulations (4 on each).

Screw the corner plates to the mending plates.
Screw the articulations to the corner plates as shown.

Step 4

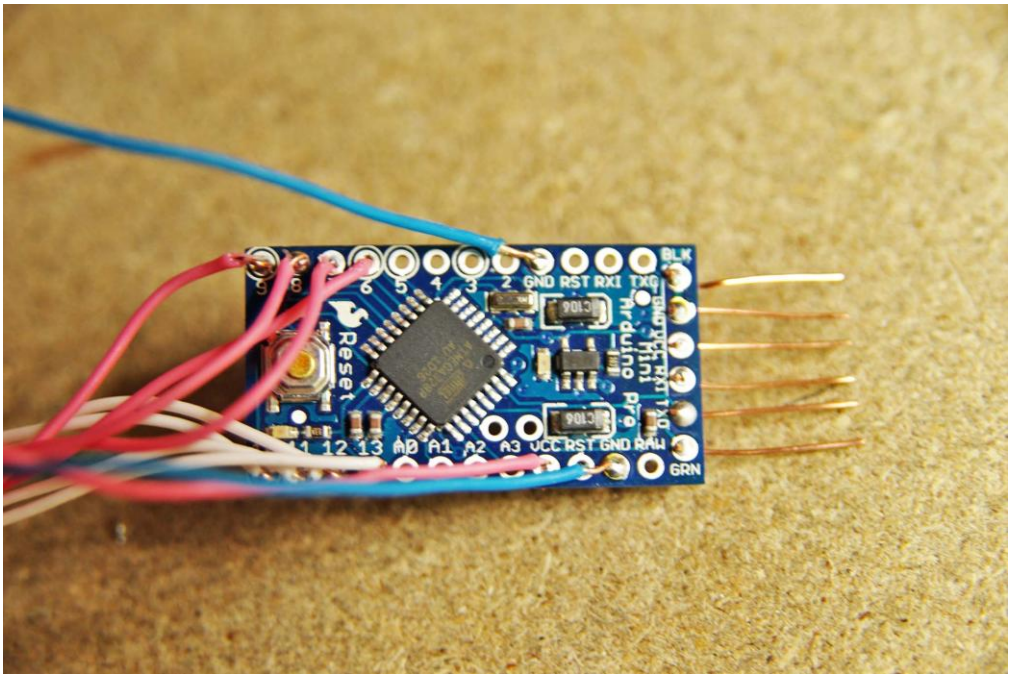
The electronics.

Solder very small wires (1 or 2cm) to pins BLK, GND, VCC, RXI, TXO, GRN of the Arduino.



Solder the negative wire of every laser together and connect it to a GND pin.
Solder the positive wire of each laser to pin 13 to 6 so you have a non-filed laser next to a filed laser.

Connect the positive wire of the batteries to the VCC pin, and the negative to a GND pin.



The Arduino should light up. But you can take the battery out for now.

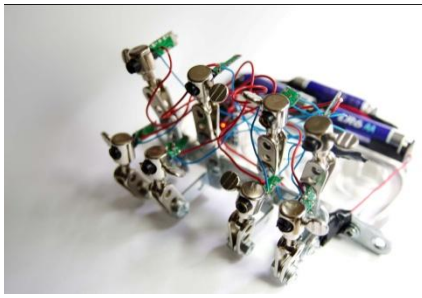
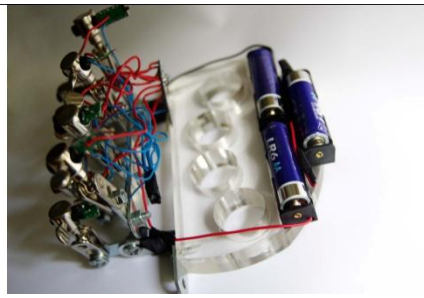
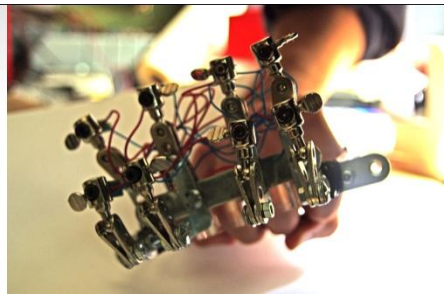
Step 5

Final building step.

Insert the lasers in the articulations in the right order:
For example if you start with the laser plugged in pin 13 and insert it in the top left articulation then the one below will be the laser plugged in pin 12 and the one on his right will be in pin 11.

Now you can fix the mount on anything you want or just hold it like this.

Ex:

Programming

To program the arduino you need to download the software on: http://arduino.cc/.
Install it (you can get some help in *Getting Started*).
Now you can communicate with the Arduino via the FTDI cable.
Make sure you selected the right serial port.

Automated solution:

Download and install Processing :
http://processing.org/download/

On ladyada's website download the Magician Soft:
http://www.ladyada.net/make/minipov3/programming.html
Install it.

Download the POV-Arduino convertor.

Enter your message in the Magician Soft, once you click on *Done*, you get a code that looks like the one below. Just copy the highlighted part.

```
#include <avr/io.h>     // this contains all the IO port definitions
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <avr/pgmspace.h>
#include <util/delay.h>

void delay_ms( uint16_t milliseconds)
{
  for( ; milliseconds > 0; milliseconds--)
  {
    _delay_ms( 1);
  }
}

#define TIMER1_PRESCALE_1 1
#define TIMER1_PRESCALE_8 2
#define TIMER1_PRESCALE_64 3
#define TIMER1_PRESCALE_256 4
#define TIMER1_PRESCALE_1024 5


// We use these macros because binary constants arent always supported. ugh.
#define HEX__(n) 0x##n##UL
#define B8__(x) ((x&0x0000000FLU)?1:0)  \
          +((x&0x000000F0LU)?2:0)  \
          +((x&0x00000F00LU)?4:0)  \
          +((x&0x0000F000LU)?8:0)  \
          +((x&0x000F0000LU)?16:0) \
          +((x&0x00F00000LU)?32:0) \
          +((x&0x0F000000LU)?64:0) \
          +((x&0xF0000000LU)?128:0)
#define B8(d) ((unsigned char)B8__(HEX__(d)))


// store all the image data in program memory (ROM)
// instead of RAM (the default)
const uint8_t large_image[] PROGMEM = {
B8(00000000),
B8(11111110),
B8(00010000),
B8(00010000),
B8(00010000),
B8(11111110),
B8(00000000),
B8(00000000),
B8(00000000),
B8(11111110),
B8(10010010),
B8(10010010),
B8(10000010),
B8(00000000),
B8(00000000),
B8(00000000),
B8(11111110),
B8(10000000),
B8(10000000),
B8(10000000),
B8(00000000),
B8(00000000),
B8(00000000),
B8(11111110),
B8(10000000),
B8(10000000),
B8(10000000),
B8(00000000),
B8(00000000),
B8(01111100),
B8(10000010),
B8(10000010),
B8(10000010),
B8(01111100),
B8(00000000),
B8(00000000),
};

// special pointer for reading from ROM memory
PGM_P largeimage_p PROGMEM = large_image;

#define NUM_ELEM(x) (sizeof (x) / sizeof (*(x)))
int imagesize = NUM_ELEM(large_image);



// this function is called when timer1 compare matches OCR1A
uint8_t j = 0;
SIGNAL( SIG_TIMER1_COMPA ) {
  if (j >= imagesize)
    j = 0;

  // read the image data from ROM
  PORTB = pgm_read_byte(largeimage_p + j);

  j++;
}
```

Put it in the miniPOV-code-Bloc_notes of the POV-Arduino convertor in between <XMLW> and </XML>.
Save it.
Via Processing open the POV_arduino_converter_01.
Run the program and copy the code generated in the consol on the Arduino.
You're done!