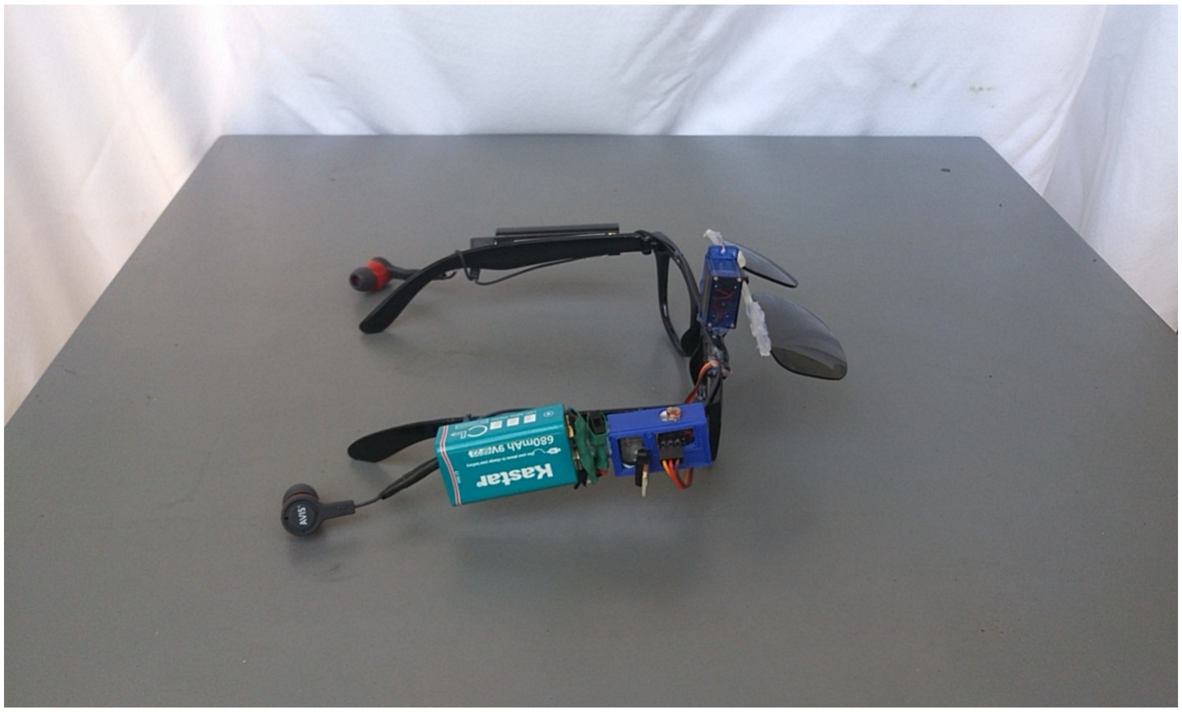
How to Make the Auto-Sunglass

By: Victor Guzman



Required Parts:

1. SG90 9g servo motor **x1** Servo
2. Light sensitive resistor **x1** **LDR**
3. 10k ohm resistor **x1** **10k**
4. Sunglasses **x1** Available at your local dollar store
5. Attiny85 Microcontroller **x1** Attiny85
6. 9v 680 mAh rechargeable battery (Kastar) **x1** Battery
7. MiCar Bluetooth receiver **x1** MiCar
8. Earbuds (generic) **x1**
9. L7805CV linear regulator **x1** L7805CV
10. Small slide switch **x1**
11. Jumper wires

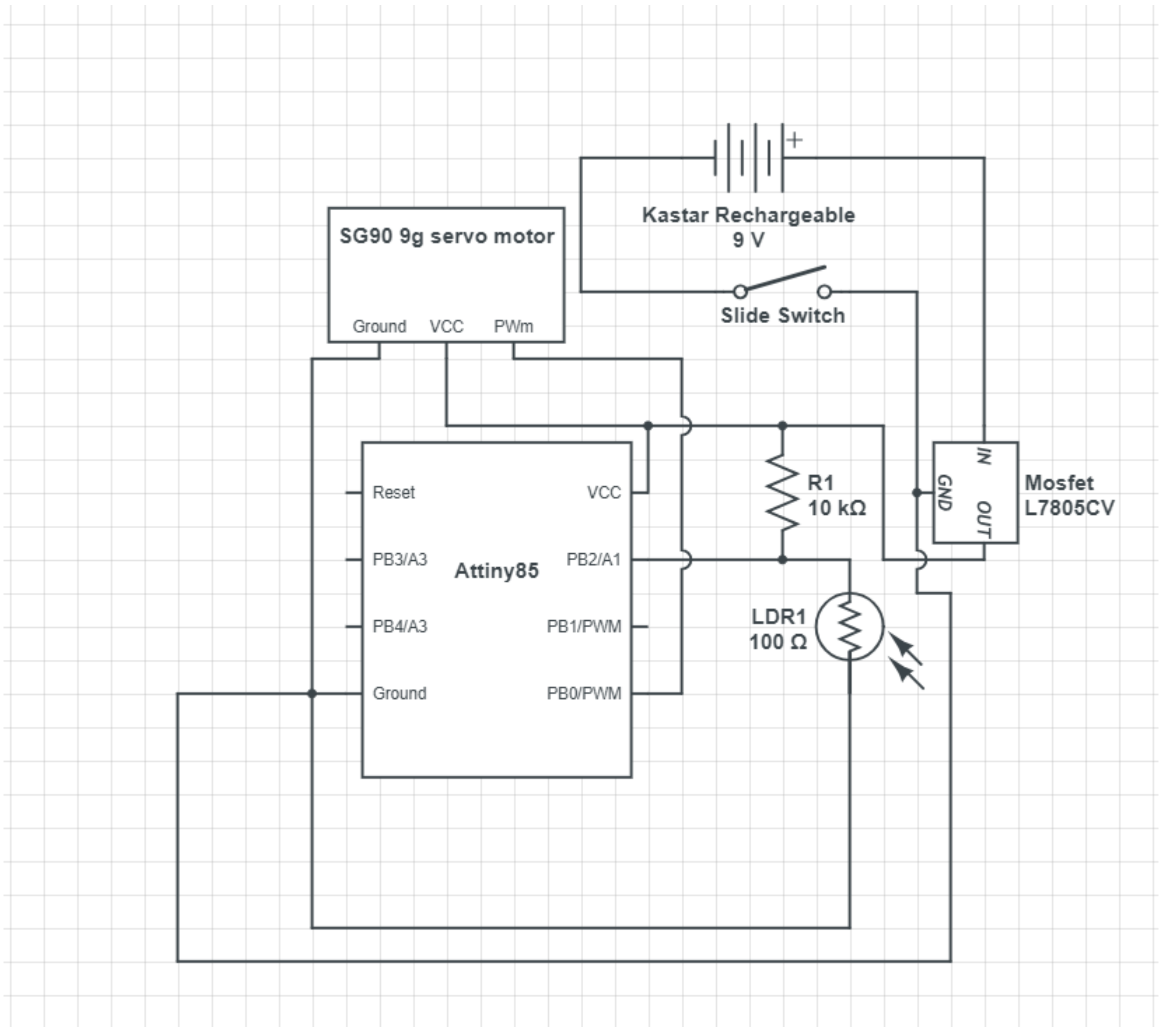
Software and Tools:

- **Arduino IDE**
- **Soldering Iron**
- **Hot glue gun**

Function of Sunglasses:

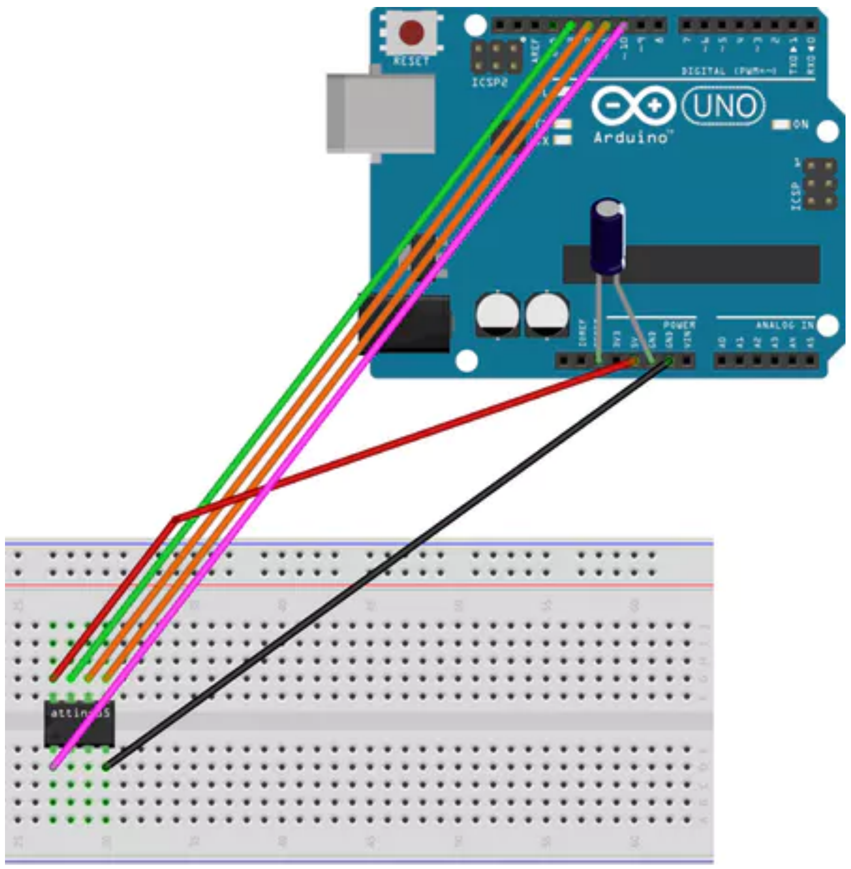
The auto-sunglass is a pair of sunglasses able to detect light intensity to determine whether to close or open a pair of shades to protect your eyes automatically. It is also equipped with a pair of Bluetooth headphones that giving to you easy access to listen to music and answer calls anywhere.

Circuit Diagram:



Programming Attiny85:
(If you already know how to do this skip to the next step)
You will need an Arduino Uno, a 10uF/25V capacitor and a jumper wires.

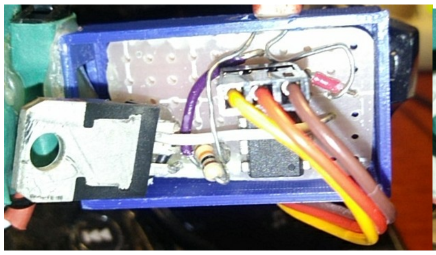
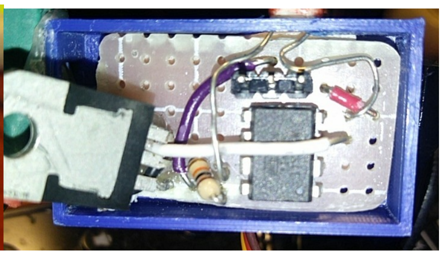
1. Since Arduino doesn't support the attiny85 we need to add the board to the Arduino IDE. In Arduino IDE go to Files->preferences and then in the Additional boards' manager URLs box enter this URL https://raw.githubusercontent.com/damellis/attiny/ide-1.6.x-boards-manager/package_damellis_attiny_index.json
2. Go to Tools->board->board manager, then in boards manager search scroll to the very bottom to "attiny by Davis A. Mellis" then install it. It should now be visible in the list of boards
3. Next, go to File->Examples->arduino isp and then upload it to your Arduino Uno board which now allows your Arduino to program other boards
4. Next connect the Arduino to the attiny85 like so



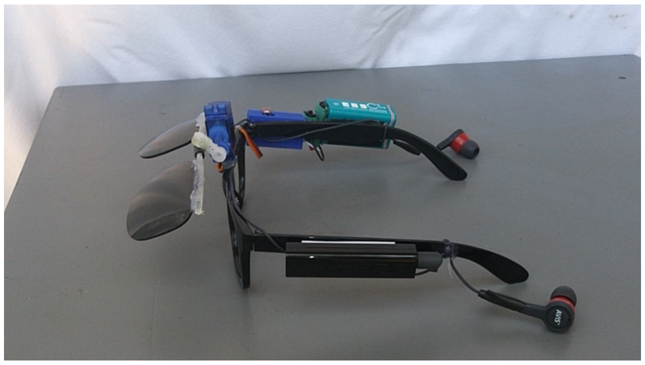
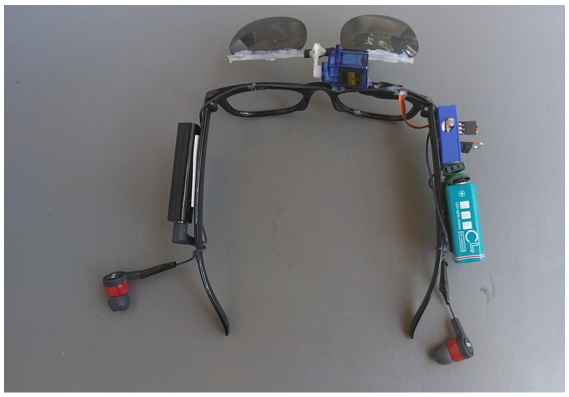
Arduino	Attiny85
5V	VCC
Gnd	Gnd
Pin 13	Pin 2
Pin 12	Pin 1
Pin 11	Pin 0
Pin 10	Reset

Now in tools change to board to attiny25/45/85 and the processor to attiny85. Next, change the programmer to Arduino as ISP. Next set the clock to 8MHz Internal then burn the bootloader.
Finally, upload your code and you should all set. (Code can be found at end of the instruction manual)

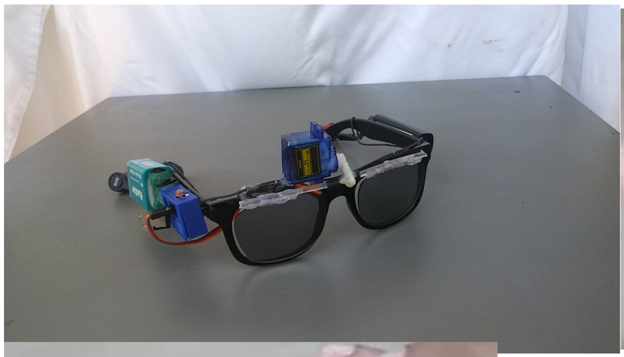
Putting it Together:
First you want to solder the electronic components to a protoboard and cut the protoboard so it fits on the side of the glasses, also attach a 9V battery adapter to the circuit board

Next, cut your earbuds and solder them back together so that they fit across the glasses and so that the earbuds hang down and comfortably fit in your ears. Also, mount the Bluetooth receiver to the opposite side of the glasses where the circuit board was mounted.

Now mount the battery to the same side of the circuit board and attach a stiff piece of plastic to the lenses so that they are covering the open holes and attach the piece of plastic to the servo motor like shown in the pictures



The Code:

```
#include <SoftwareServo1.h>  /* use SoftwareServo Library */

SoftwareServo1 Tyrone;       /* Name the servo "Tyrone" */
int servoPin = 0;           /* Create variable for Servo on Pin 0 */
int ldr = A1;
int value = 0;
int i;/* Create variable for counting refreshes */

void setup()
{
  Tyrone.attach(servoPin);         /* Tyrone is on the servoPin */


  /* The best minimum & maximum pulse widths will depend on the brand & type
     of servo used.  These settings work well with our Towerpro microservos.   */

  Tyrone.setMinimumPulse(496);  /* set minimum pulse width */
  Tyrone.setMaximumPulse(2245); /* set maximum pulse width */
}

void loop()
{
  value = analogRead(ldr);
  delay(100);
  if(value<100)
  {
    Tyrone.write(155);
    for (i = 1; i < 30; i++)
  { SoftwareServo1::refresh();  /* This is the refresh function */
    delay(50);
  }
  }
  else
  {
    Tyrone.write(80);
    for (i = 1; i < 30; i++)
  { SoftwareServo1::refresh();  /* This is the refresh function */
    delay(50);
  }
  }
}
```