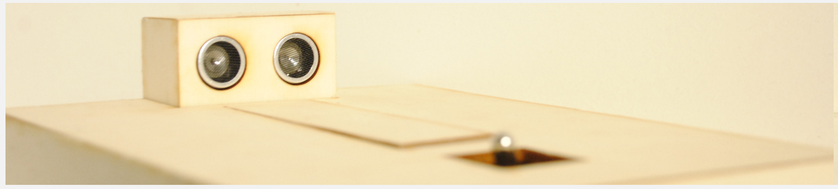


GIZMO

James Howells •

THE CONCEPT

USELESS BOX

A its name suggests, the Useless Box is a box that does very little that is of use to anyone. In an iterative interaction with the user, the box can get progressively more 'annoyed' as the user turns the switch on - at first, it simply turns the switch back off but as it gets more exasperated about this not working, it starts to take more action - hiding the switch from the user, or even running away.

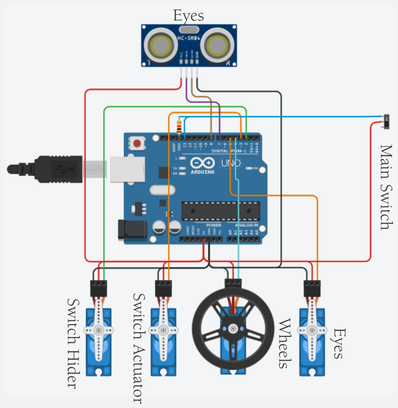
The way the box reacts (how vigorously it turns the switch off) depends on how fast the user switches it on: the longer the user takes to turn the switch on, the 'sneakier' the box is about turning it off.

ELECTRONICS & CODE

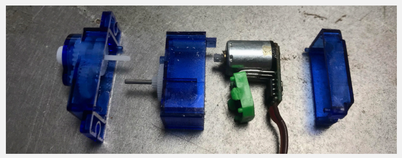
The electronics need to:

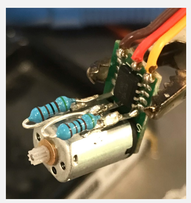
- Show/hide the switch
- Actuate (turn off) the switch
- Turn the wheels
- Show/hide the 'eyes'
- Sense distance using the eyes
- Sense when switch is turned on

The box needs four actuators to power its functions, and an ultrasonic sensor and switch as inputs. For the actuators, I will use servos: one of them modified for continuous rotation (below). These servos will be coupled with some fairly complex mechanical systems to transform the rotational movement into the required movement types (see 'mechanisms').



MODIFIED SERVO





I modified an SG90 servo by removing the potentiometer (green, above) and replacing it with two 5 kΩ resistors (left). This means that the servo doesn't know what angle it is at, and so continues rotating while any angle other than 90° is selected in the code.

CODE

The box behaves differently based on how fast the user reacts. As the CPU is single-core, a built-in timer is hard to execute: instead, I used `millis()` to calculate the dime difference `deltaT` since the previous action (setting `startTime = millis()`, then later `deltaT = millis() - startTime`). This works because `millis()` gives the number of milliseconds since the Arduino booted. A pseudo-code schematic is below.

Pseudo - Code

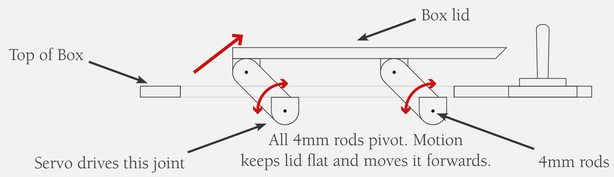
```
timer.start()

if (switch is turned on) { switch.turnOff() } x3

eyes.show()

if (user gets close) { switch.hide() } x3

if (user gets really close) { runAway() } x3

if (user still trying) { give up }
```

MECHANISMS

Despite the outward simplicity of the Pointless Box, there is a lot under the surface. I built the below mechanisms to make it work.

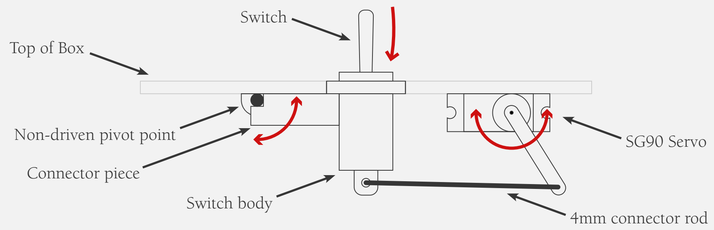
Switch De-activator:

The switch de-activator uses a servo to drive one of four mechanically linked arms that raise the lid of the box, pressing the switch.



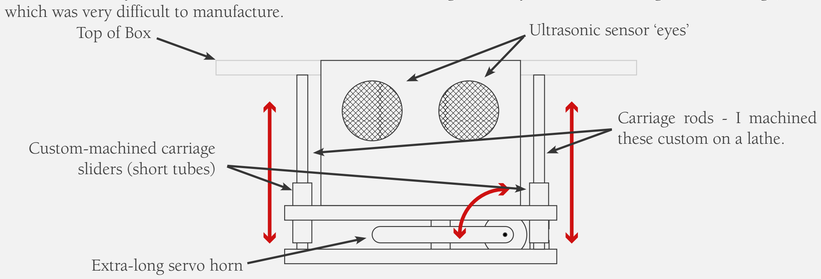
Switch Hider:

The switch hider hides the switch from view using a servo linked to a swivelling carriage by a steel rod. It has to be very sturdy to withstand the pressure of the user flicking the switch. This took several failed iterations of designs that were nowhere near strong enough (see right).



Eye Actuator:

The eye actuator hides/shows the eyes by sliding them up and down. Based on ultrasonic distance values, the box can then run away or hide the switch. This is achieved with a regular rotary servo and a sliding vertical carriage, which was very difficult to manufacture.

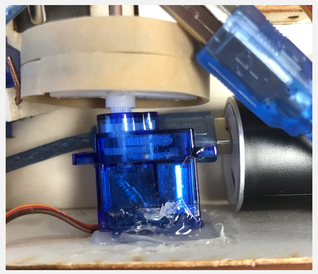


Wheels:

The wheels allow the box to 'run away' and are driven by a regular SG90 that I modified for continuous rotation (see 'electronics' section for details). I could then control the speed of the servo by setting the 'angle' in the software.

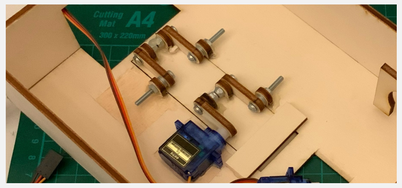
I cut the wheels out of plywood, then, in order to give them more grip, I glued a rubber band around the edge - this meant they wouldn't slip.

Because of this configuration, I could use a simple direct drive with no gearing to drive both wheels, which are attached to the same axle.



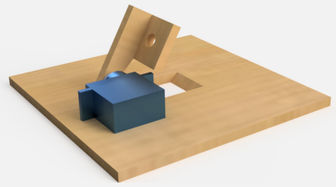
EVOLUTION

The designs that made their way into the final Gizmo, and are presented on this poster, but here are previous iterations:



At first, I was using a very flimsy switch hider using a single servo. This didn't work for 2 reasons: the switch didn't clear the cutout in the box top, and the servo did not hold the switch firmly enough.

The original switch de-activator had threaded bolts and nylock nuts instead of steel bars. This didn't work as the lid continually got stuck and twisted. For my second iteration, I replaced the bolts with 4mm welding rods. This gave it a lot more stiffness and fixed the issue.



I started out trying to make the carriage for the eye actuator out of plywood. This didn't work: there was too much friction and the eyes did not fall back down smoothly. I fixed this with the steel carriage rails explained to the left (see image too).

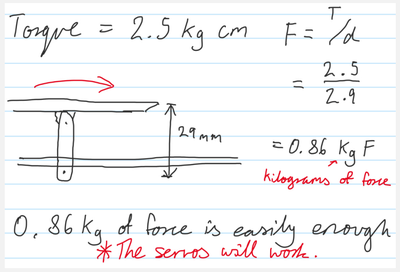
ANALYSIS

Before starting to make the box, I did some analysis of the actuators I was planning to use to make sure they'd be suitable.

SG90 SERVO

Initial Plan: use SG90s to power the switch hider, switch de-activator and eye actuator, with direct drives, un-geared.

Validation: the highest force will be needed by the switch de-activator as the switch is quite stiff. I will calculate the torque the servo can produce and the required distance.



28BYJ-48 STEPPER MOTOR

Initial Plan: use the 28BYJ-48 stepper motor supplied in the starter kit to drive the wheels on the bottom of the box directly: attach the wheels onto the stepper's shaft.

Validation: I will check the torque the stepper can produce and make sure that the RPM is fast enough to drive the box at a reasonable speed.