Bird Sense

A big thank you to **Maker's Asylum** and its **Innovation School Programme** for helping me incubate this project, this project only exists because of this organisation. A special thank you to my individual supervisor, **Narender Sharma** for constantly being there and directing me around all the road bumps on my way. A special mention to also **Tushar Naik** for his valuable assistance and advise.

Problem

Pigeons are a nuisance for people with balconies and open windows. They dirty the entire place with their feathers and droppings. Often the only solution for them is a pigeon net. Which however are very un-aesthetic. Nobody likes buying a gorgeous flat a putting bird net on the windows in major cities like Mumbai.

Existing solutions

1. Bird nets.
2. Bird repellents using sound frequencies between 15-25 kHz.
3. Roosting spikes.
4. Anti-bird strobe light

Solution Types

1. Detect the pigeon before it enters and prevent it from entering.
2. Detect the pigeon when it enters and then remove it.
3. Hanging dummy scarecrows / predatory birds.

What things are pigeons responsive to?

1. They can hear sound up till 15kHz but can find particular frequencies above 15kHz disturbing.
2. Water thrusts are unpleasant to them.
3. Predatory or abrupt sounds are alarming.
4. Vibrations can be alarming to them.
5. Reflective lights can cause temporary visual disturbance to them.
6. Pigeons can detect the Earth's electromagnetic field.

Before it enters

Detecting:
-Ultrasonic sensor.
-Motion sensor / PIR sensor.
-Artificial Intelligence using a camera.

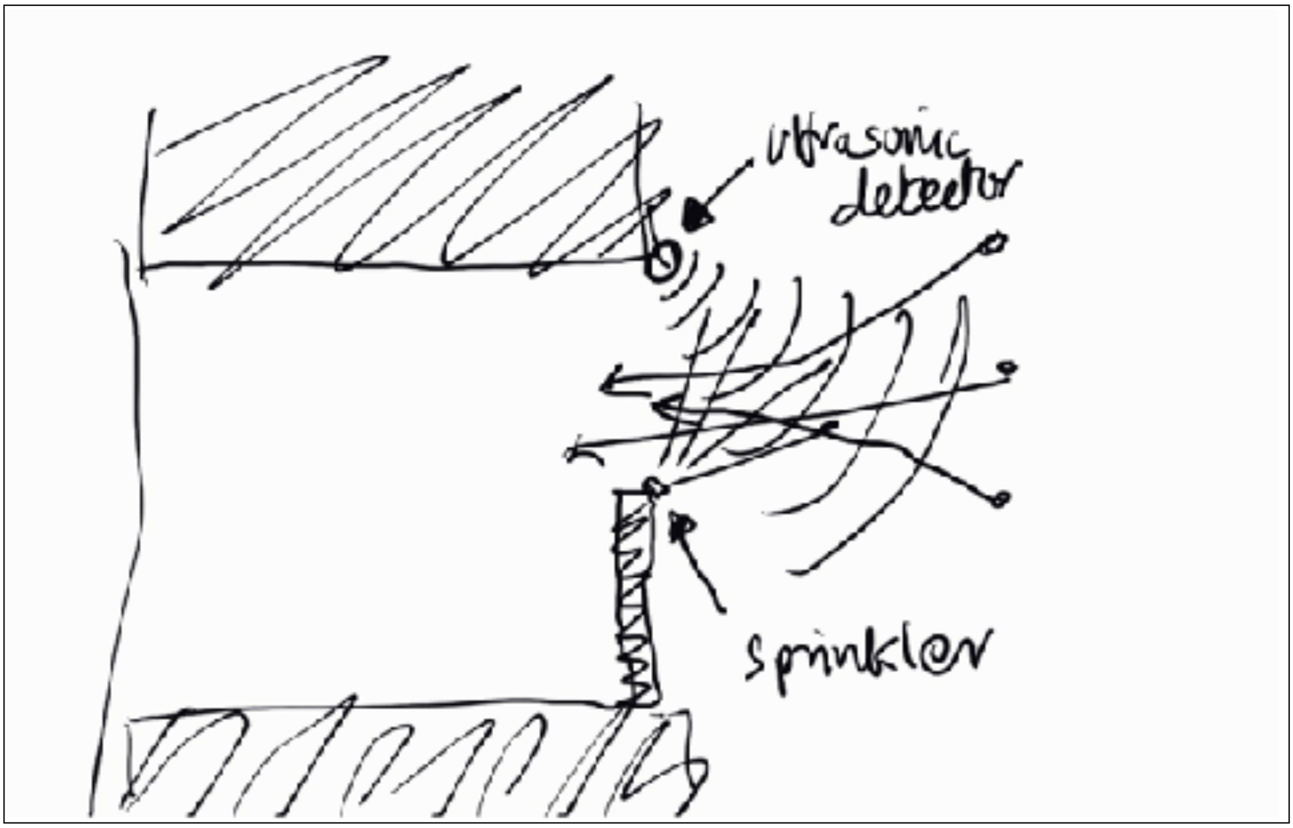
Preventing:
-Temporary Barrier.
-Pop up human / hawk.
-Water thruster.
-Thud Sound.
-Hawk sound.
-Low frequency of sound.
-Light flash (can backfire).

When it enters

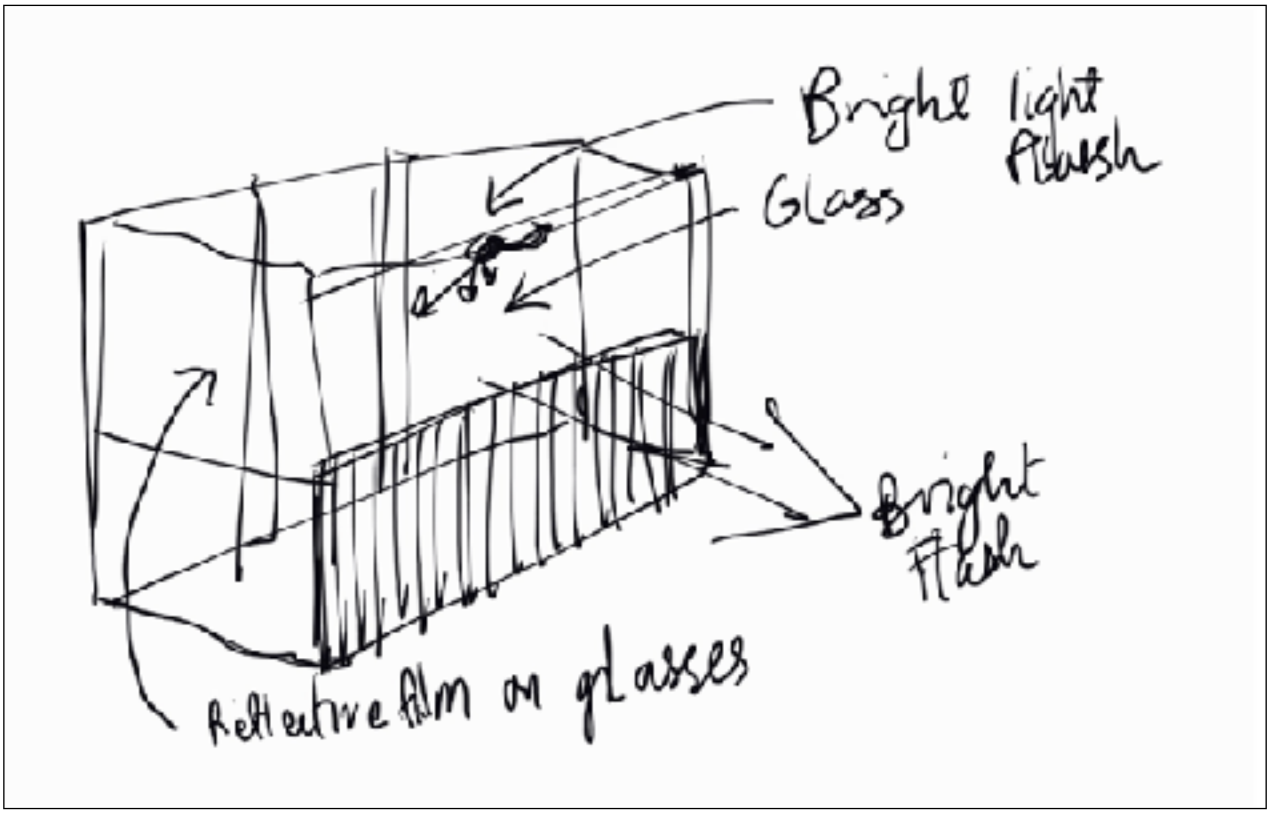
Detecting:
-Ultrasonic sensor.
-Motion detector.
-Artificial Intelligence using a camera.
-Weight sensors.
-Lasers.

Removing:
-Pop up human / hawk.
-Thud Sound.
-Hawk sound.
-Low frequency of sound.
-Vibrating the ground.

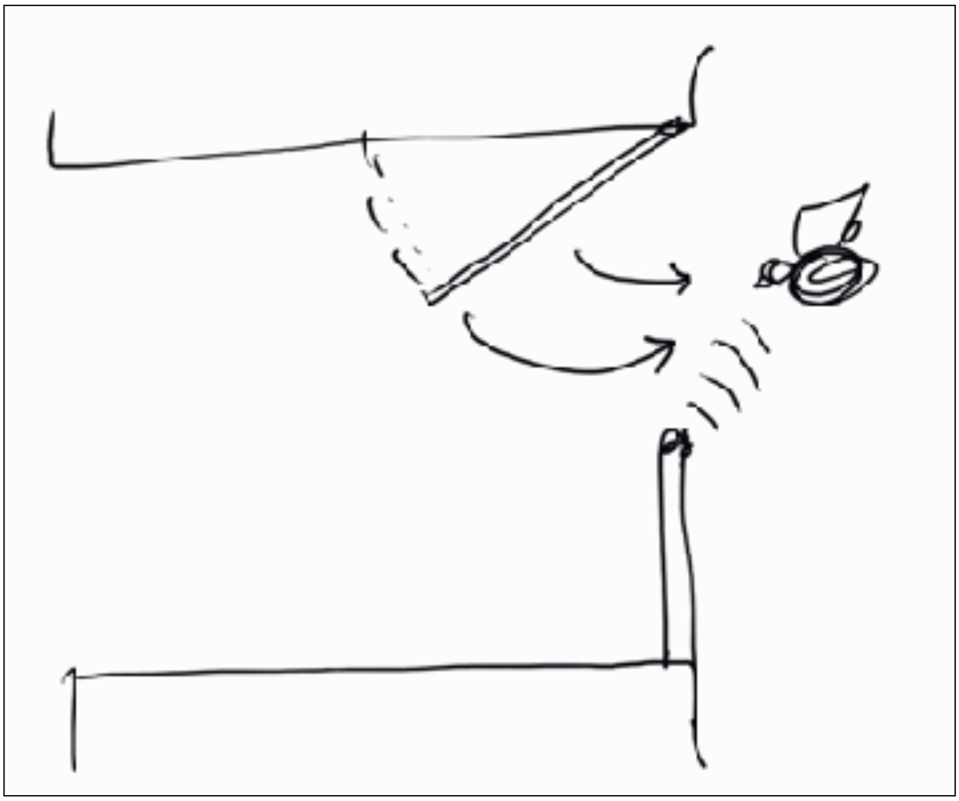
Few Sketches



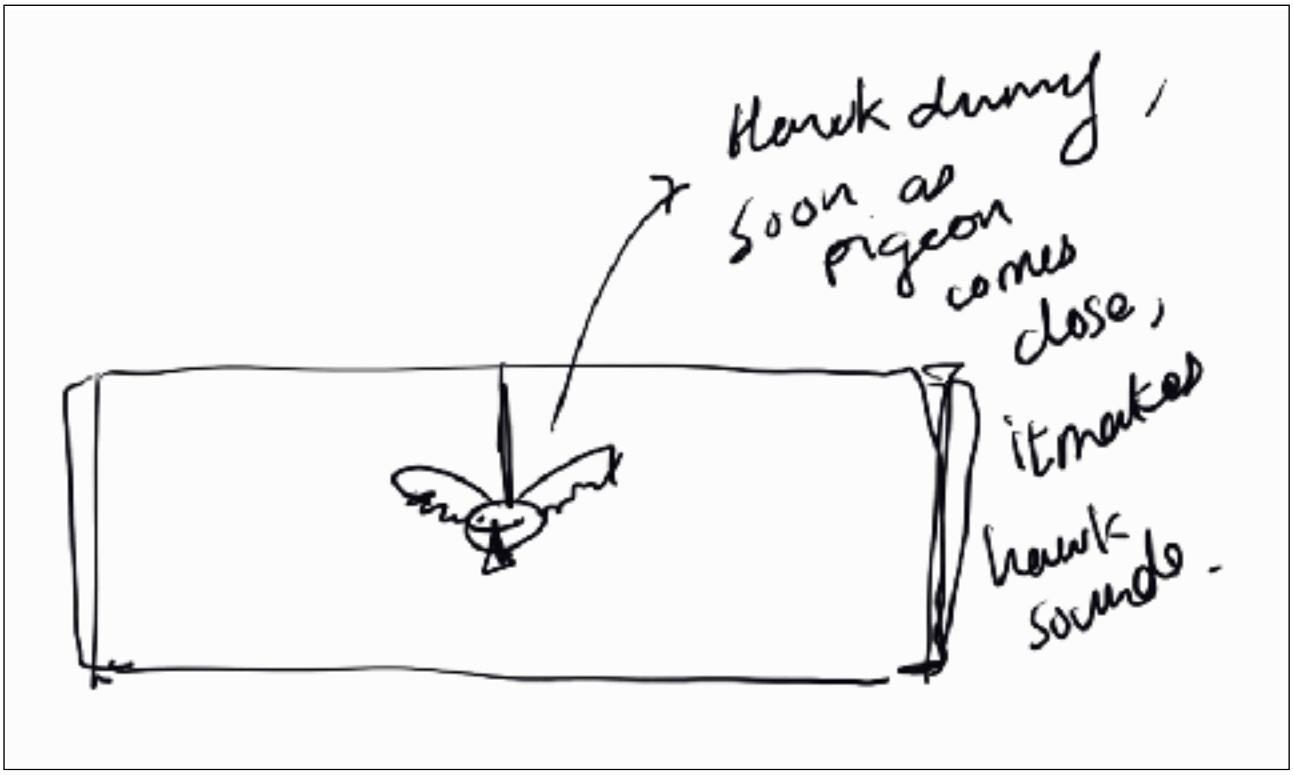
Detecting the pigeon before roosting or entry and preventing using water



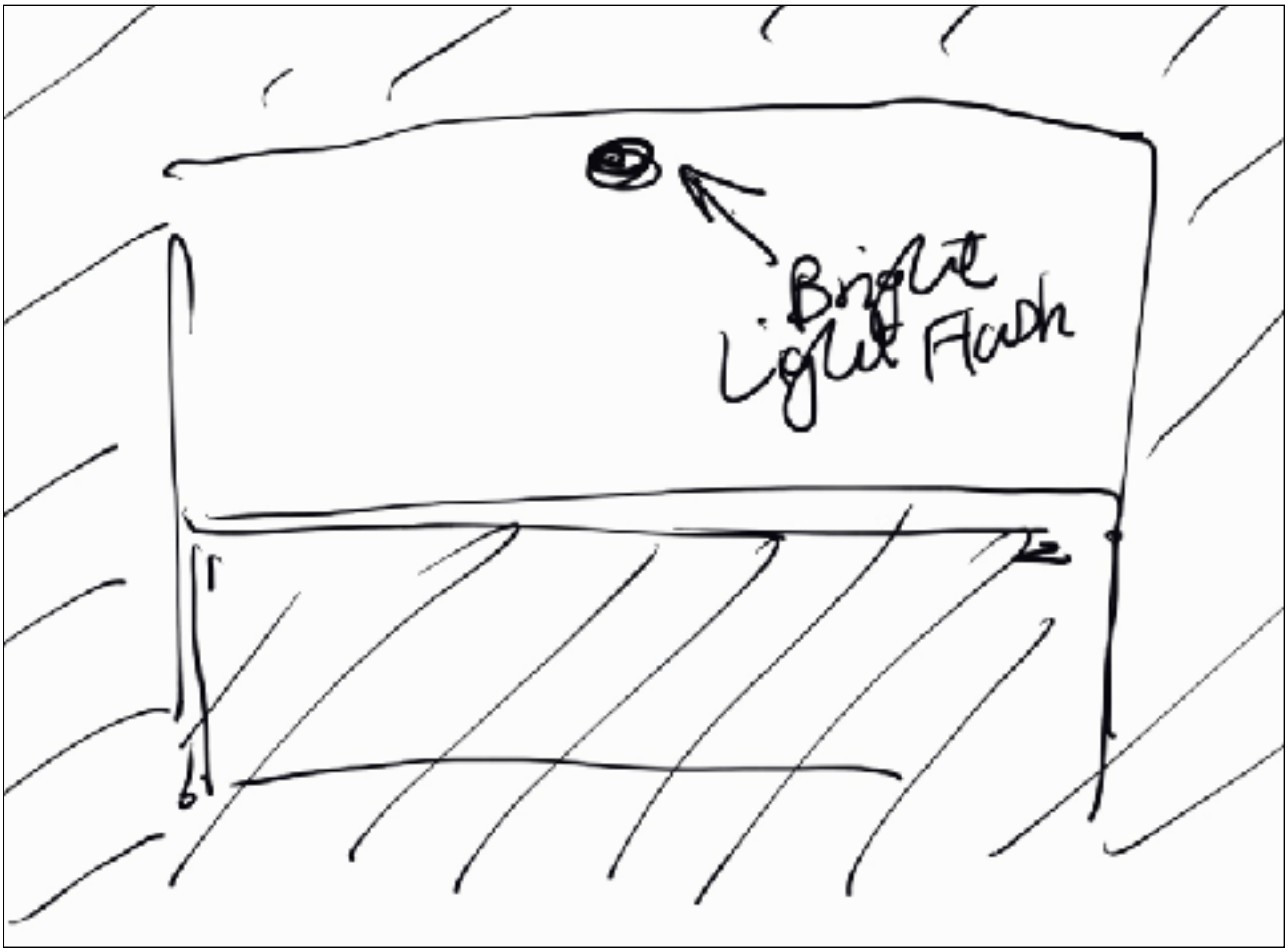
Using the window as a reflective surface to blind the pigeon before it enters



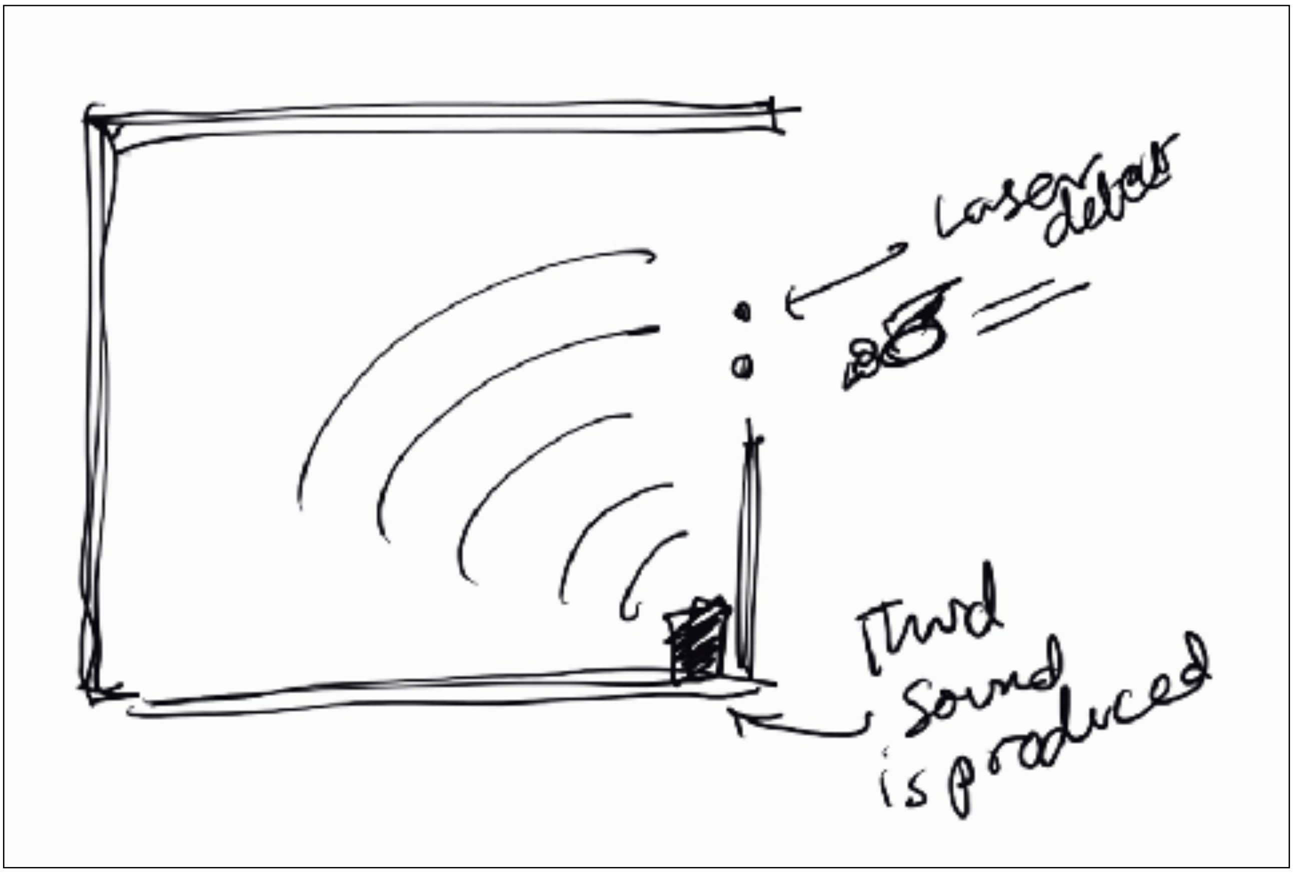
Detecting the pigeon before it enters and then deploying a barrier


Hawk dummy, soon as pigeon comes close, it makes hawk sounds.

Using a hawk dummy that acts as a scarecrow and makes a alarming sound for the pigeon



A temporary bright flash can disturb the pigeon's visual ability



Pigeon is detected and scared away using an abrupt sound

VERSION ONE:

Version one focuses primarily on the concept of detecting pigeons using PIR/Motion sensors before the pigeon enters the balcony and then working on a way to disturb or scare them away.

Experiment 1.0

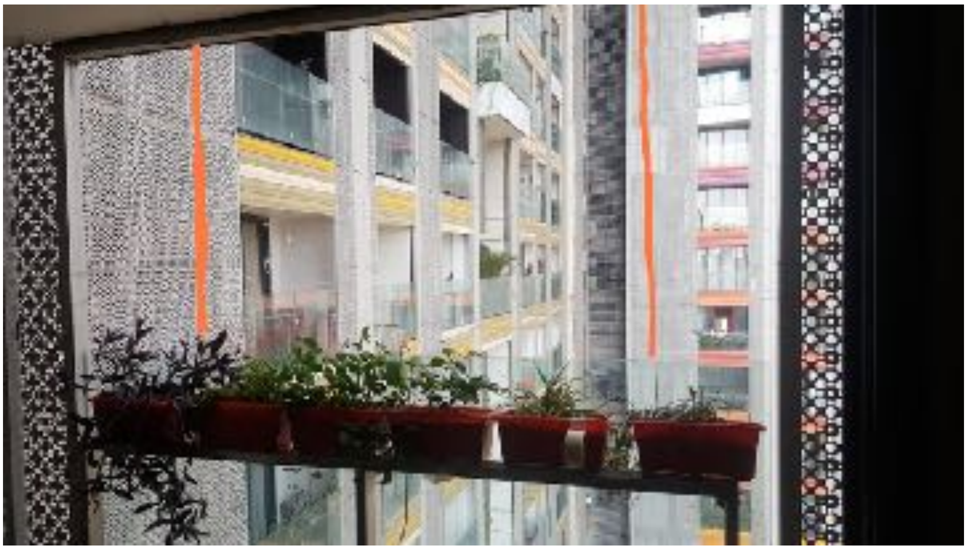
To check whether the pigeon will be detected with the PIR sensor successfully, I setup an ESP8266 in my balcony that logged the time onto an Adafruit feed when the PIR sensor returned a positive feedback.

To verify whether the detection was only for a pigeon and not something else, a camera was put to record the entire balcony. After the experiment I cross checked the time logs and matched it to the video to see whether the detection was a false one or a true one.

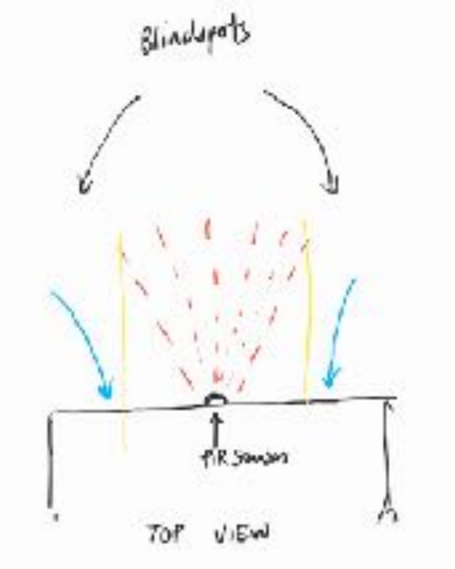
Unfortunately, the experiment was unsuccessful because multiple detections were logged every minute. And almost none of them were pigeon detections. This happened despite the PIR sensor being calibrated to minimum sensitivity and time delay. I suspect the jumper being faulty.

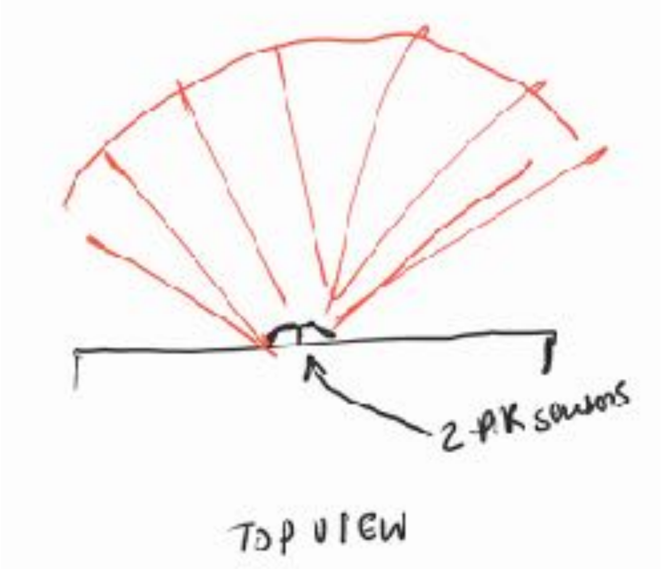
Experiment 2.0

This time I used an Arduino for the sake of simplicity, changed all the wires and also used a buzzer.



This experiment was super successful and had zero false detections, the buzzer only buzzed when I put my hand in front of the sensor or a pigeon came. However the only issue I learnt was that the PIR sensor would be able to detect a bird if it came from the range between the red lines drawn.

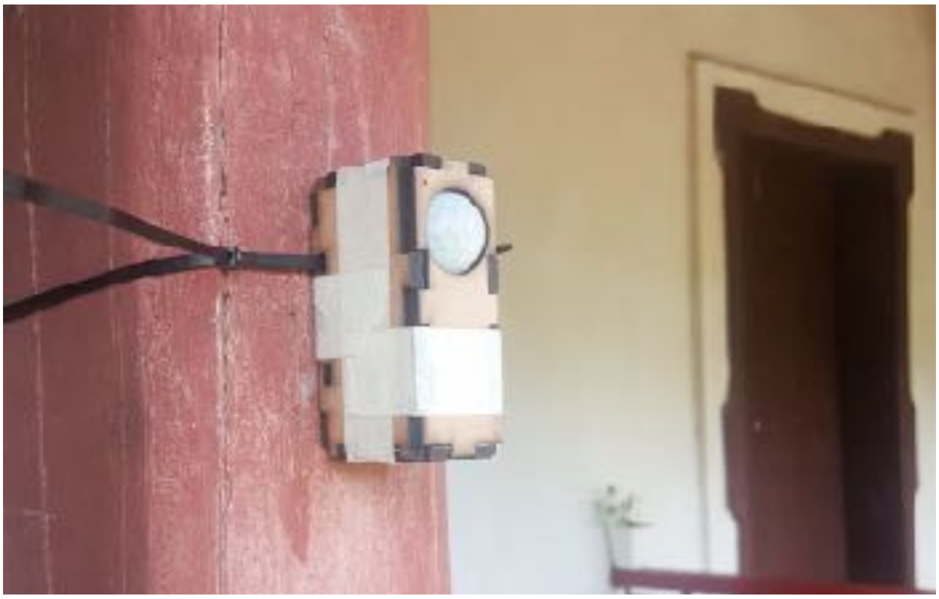




As drawn in the diagram above, if the pigeon enters far enough to the side that it doesn't cut the 3m long virtual cone space created by the PIR sensor, it will not be detected. This blindspot hasn't been a problem yet because pigeons usually enter through the centre. But if they do figure out the blindspot, it will become a problem. To cover the blindspot, two possible solutions are there: use two sensors and get a wider area of detection with more accuracy as shown in the diagram above; or use different frenzel lenses. Option 2 is definitely more cost effective and easier. But it's doubtful how much of the blindspot will it be able cover.

Experiment 3.0 & Prototype 1

Using a laser cutter, a small box was made from 3.5mm MDF wood. Inside the box was a simple circuit with a Arduino Nano, buzzer and PIR sensor. The box was placed on the pillar of a window facing outwards to detect for incoming pigeons. However, this experiment wasn't successful because it detected cars from the nearby street even after being on the minimum sensitivity setting.



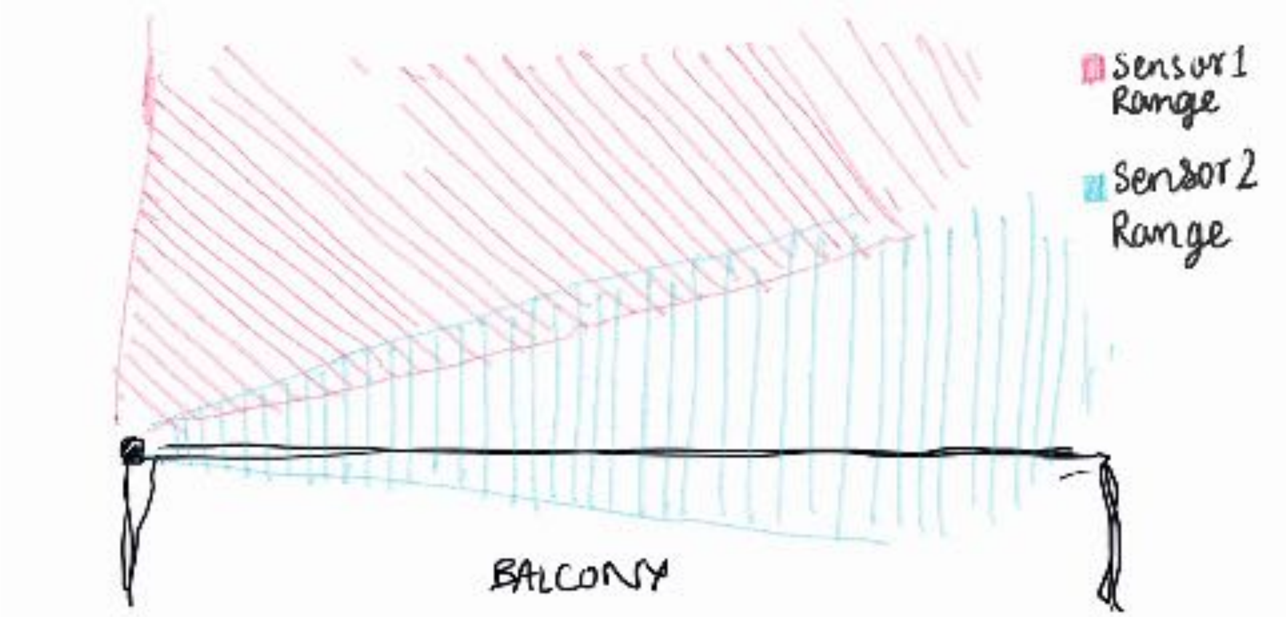
Experiment 3.3

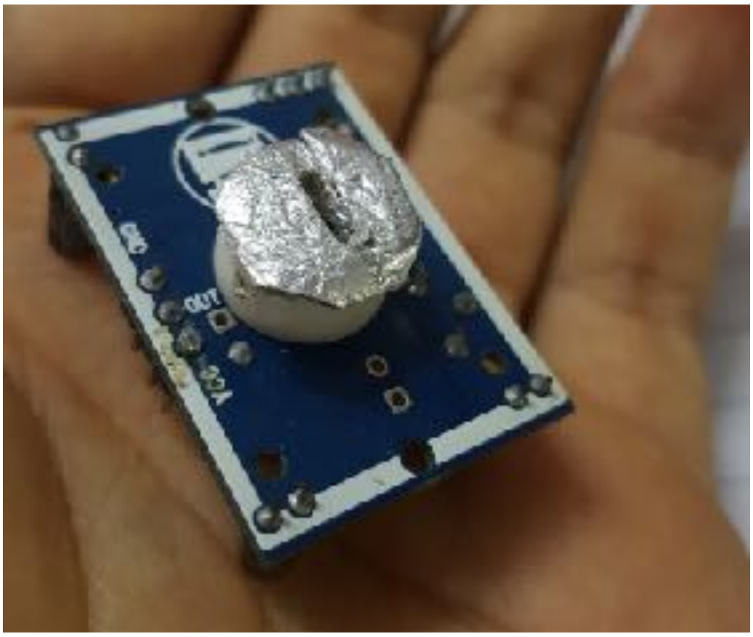
Using the same prototype from the last experiment documented (E.3.0), the device was placed on the balcony railing as shown below.

Since the sensor was still detecting the movement from nearby leaves on branches. I put a small piece of plastic (the double sided tape's removable covering) between the sensor and the frenzel lens. This however reduced the accuracy of detecting an object in its proximity but it also prevented any false positives. Since this particular area didn't have the pigeon problem, we just simply threw a beanie back and forth to simulate the pigeon behaviour (E.3.3_video.mp4).

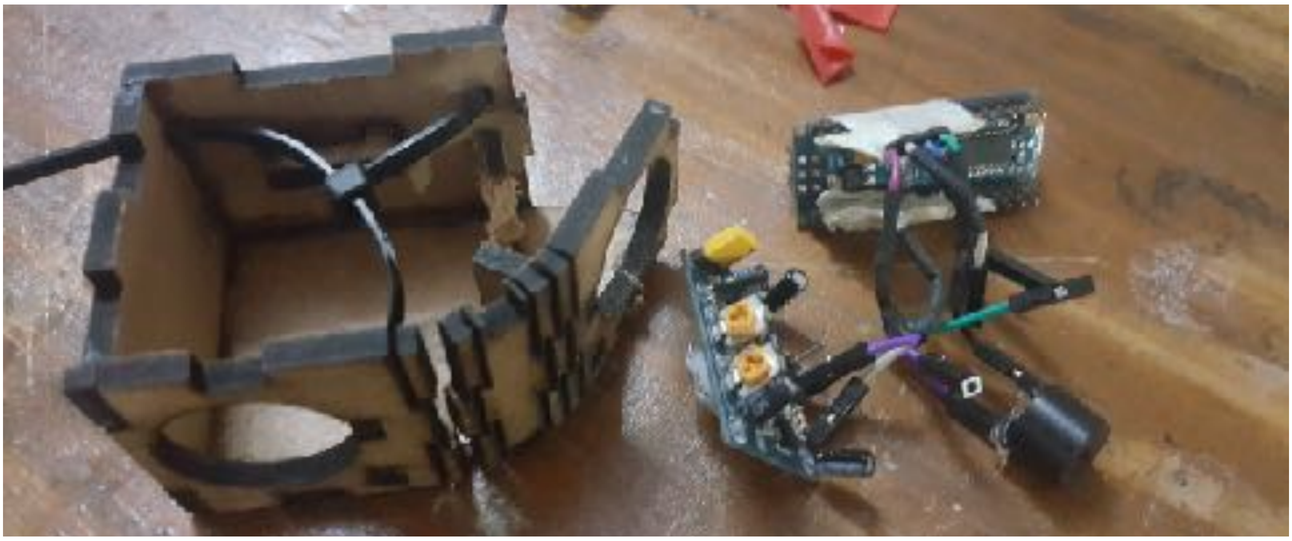
Prototype 2

Now, one major problem still remains, which is human's leaning out of the balcony will trigger a false positive. To solve this problem, we could reposition the device to the balcony/window corner and add another sensor.





Sensor one faces outwards looking for pigeons and sensor two faces right along the railing but with an altered frenzel lens giving it a narrower detection range. Now for a pigeon coming inside, detection pattern should be 1, 2. But for a human being just 2 and if decides to lean out, 2,1. For a false detection like leaves moving nearby, the pattern would be only 1 or 2. Either way, the 1,2 pattern would be primarily unique to incoming objects. For a balcony, a majority of this 1,2 detection pattern would be a bird.



Before making prototype 2 to test this, I had to find a way to narrow down the sensor 2's detection range. We that using multiple techniques, from using aluminium foil to cover the sides of the sensor to 3d printing a cap with a slit. But we ended up using a combination of both these two methods.

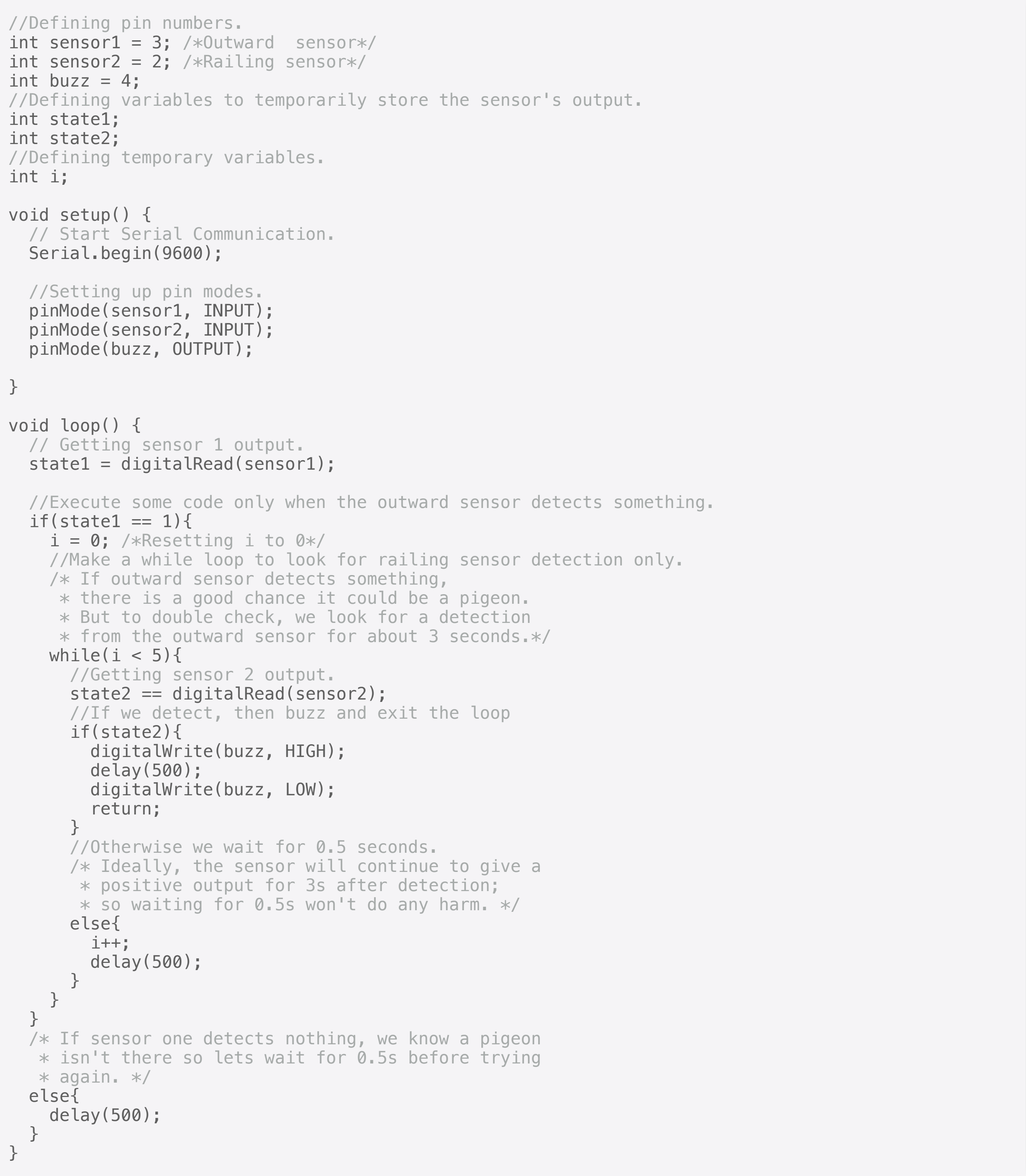
After a lot of failed attempts, we finally figured it out. To incase the entire circuit, a laser cut casing of MDF. However the design wasn't perfect and eventually failed to case the components correctly and it eventually broke because it was too small.

Experiment 4.0 & Prototype 3

Prototype 3 is a remastered version of prototype 2 with most of the errors fixed.

Unfortunately, the location where this was build did not have the pigeon problem, it was difficult to simulate pigeons also because pigeons just like any other living thing emit some infrared light. So we tested this with humans itself! By placing this on a nearby table corner, I observed when anybody would pass by. The experiment was successful (at least with humans 😂) but still it wasn't perfect.

Code:



```
//Defining pin numbers.
int sensor1 = 3; /*Outward  sensor*/
int sensor2 = 2; /*Railing sensor*/
int buzz = 4;
//Defining variables to temporarily store the sensor's output.
int state1;
int state2;
//Defining temporary variables.
int i;

void setup() {
  // Start Serial Communication.
  Serial.begin(9600);

  //Setting up pin modes.
  pinMode(sensor1, INPUT);
  pinMode(sensor2, INPUT);
  pinMode(buzz, OUTPUT);

}

void loop() {
  // Getting sensor 1 output.
  state1 = digitalRead(sensor1);

  //Execute some code only when the outward sensor detects something.
  if(state1 == 1){
    i = 0; /*Resetting i to 0*/
    //Make a while loop to look for railing sensor detection only.
    /* If outward sensor detects something,
     * there is a good chance it could be a pigeon.
     * But to double check, we look for a detection
     * from the outward sensor for about 3 seconds.*/
    while(i < 5){
      //Getting sensor 2 output.
      state2 == digitalRead(sensor2);
      //If we detect, then buzz and exit the loop
      if(state2){
        digitalWrite(buzz, HIGH);
        delay(500);
        digitalWrite(buzz, LOW);
        return;
      }
      //Otherwise we wait for 0.5 seconds.
      /* Ideally, the sensor will continue to give a
       * positive output for 3s after detection;
       * so waiting for 0.5s won't do any harm. */
      else{
        i++;
        delay(500);
      }
    }
  }
  /* If sensor one detects nothing, we know a pigeon
   * isn't there so lets wait for 0.5s before trying
   * again. */
  else{
    delay(500);
  }
}
```

Limitations

- The device had a blind window for about 3s after it had already detected something because of the default delay in the PIR sensor.
- The device was still missing the ultrasonic tweeter (because it was extremely hard to obtain and the shipment had not arrived yet).

- It wasn't water resistant: this was a significant issue because these devices are meant to be placed in outdoor conditions.
- The device would require a super long cable if it had to be set up in a balcony. Two work arounds are there to solve this problem but they are ruled out because either of them are not feasible.
    1. Using solar panels. However, they cannot produce enough energy to balance out the requirement. Bigger solar panels can be used, but the device just becomes bigger. Additionally not all balconies may have direct sunlight.
    2. Use rechargeable batteries. Even if large capacity batteries are used, they will require charging at least after 2 weeks or so, which is impractical for the average customer because it isn't convenient.

Experiment 4.1