

ES96 Section B Final Report - Spring 2019

Section Leader: Kelly Miller

Teaching Fellows: Joel Dapello, Anahide Nahhal

Students: Billy Koech, Aidan Crawford, Charlie Colt-Simmonds, Andrew Yang, Amanda Gomez, Theresa Manivanh, Terrance Williams, Gunnar Allison, Anthony Kenny, Anthony Aybar, Augusta Uwamanzu-Nna

I. Executive Summary

Our engineering team was tasked with improving the effectiveness of pest control on and around Harvard's campus. This was to be accomplished either via direct reduction of local pest populations or via improvement of Harvard's existing pest control methodologies. To begin to conceptualize this complex problem, the team produced both stakeholder and systems maps of the problem space in order to visualize both potential solution pathways and potential roadblocks to implementation of those solution pathways. After fully analyzing the problem space, the team was able to generate the following problem statement:

How might we improve the effectiveness of pest control methods and decrease pest access to food sources at Harvard?

From this problem statement, the team began a comprehensive ideation process. This process involved brainstorming for all possible solutions to the defined problem. Once created, this list of solutions was evaluated using a three-tiered linear evaluation system devised by the team. This linear evaluation process narrowed down the solution space to two possible solution pathways, both of which were implemented by the team.

A. Re-Engineered Pest Trap Integrated Trash Can

During the team's initial research phase it was determined that trash was a major component of pest food access around Harvard. To mitigate this access, the team designed a trap that would be integrated into the lid of a standard trash can. The pest would be attracted to the trap by the smell of the trash and would then be trapped within the lid-trap where upon it would be killed using a spring-loaded "snapping" mechanism. The pest carcass could then be disposed of in the attached trash can by a technician.

B. Improved Data Collection and Visualization

In initial interviews with Harvard Environmental Health Services (EHS), the team determined that current methods for the collection and storage of pest data were inefficient. Therefore, the group designed a beta-version of a data collection and visualization server. This server was implemented using Amazon Web Services (AWS) under the domain “www.crimsonpest.com” The website allows users from the Harvard Community to input pest sighting locations and other information. Once submitted, this information is collected into a cloud-storage database. The group also designed an infrared motion sensor, which communicates the GPS location associated with motion to the cloud database via WIFI. The server then allows an EHS administrator to access the data superimposed onto a map of Harvard campus for better data visualization and trend identification.

The working goal of this project is to provide Harvard EHS with a framework via which they may implement changes to current operating practices that will address the concerns named in the problem statement and thereby contribute to the mitigation of pest activity on and around Harvard campus.

II. Table of Contents

I. Executive Summary

II. Table of Contents

III. Introduction

IV. Researching the Problem Space

V. Investigating the Solutions Space

Ideation Phase and Divergent Solutions

Criteria and Process for Convergence

VI.A. Solution 1 - Trash Can Trap

The Idea

The Design Process

Killing Mechanism

Field Testing

Prototype 2

VI.B. Solution 2 - Data Tracking and Visualization

Data Storage and Visualization

Motion Sensing Device

VIII. Appendices Product specifications Detailed cost calculations

III. Introduction

At Harvard, as with any other university campus or public space, the existence of rats and other pests is not appreciated in the slightest. Current pest control methods have made tremendous progress at mitigating the prevalence of rat burrows on campus and effectively driving down the rat problem. However, more can be done to suppress the pest population. For the spring 2019 semester, our subsection of the ES96 Engineering Problem Solving and Design course was tasked with finding a better solution for suppressing the pest population at Harvard. Our client, Richard Pollack Ph.D., is a Senior Environmental Public Health Officer at Harvard who specializes in pest management strategies throughout the campus. Through our semester-long course, we partnered with Dr. Pollack to determine ways in which current pest control methods could be improved.

Apart from being a visual displeasure, rats and mice also pose major threats to public health and infrastructure. They are serious vectors for disease and are capable of chewing through electrical wires which can become a fire hazard and technology nightmare. Currently, whenever pest issues are found on campus, they get reported to Yard Operations. Yard Operations then keeps various parties informed, including our client, through disorganized email chains. Then, one of five pest control companies gets hired to handle the actual pest issue. Rats and mice come to Harvard for the same reason they would be attracted to any other public space: an endless supply of food waste. This means that overflowing trash cans, exposed dumpsters, and uncovered waste bins indoors and outdoors are the primary sustenance for pest populations that live near them. To drive out existing rats, relevant parties on campus use artificial rock traps, burrow gassing, poison, motion detectors, and pest-proof trash cans. While the more aggressive pest control methods are effective at neutralizing pests, having a pest-proof trash can is only effective if all other trash cans in the nearby vicinity are pest-proof. Pests get into dorms and university facilities through door gaps as small as a quarter of an inch, and when there is no gap to let them through, they chew one that will. Once the pests are inside, their presence is unhappily felt by students and faculty. Our client's goal is a future at Harvard where pest sightings are a rarity, and our mission this semester was to find additional ways to make that goal a reality.

IV. Researching the Problem Space

The first step we took in this semester long project was to collect data and other useful information about the existence of mice and rats at Harvard. This section contains a step-by-step breakdown of everything we did to get all the information that we needed.

Site visit, interviewing client

Our client was kind enough to give us a slideshow presentation at our first meeting outlining the nature of the mouse and rat problem and what was currently being done to fight against it. The presentation contained valuable information about ways that mice and rats get into buildings, ways to detect their presence, and ways to contain them. Following the presentation and a question and answer session, our client showed us around the Harvard Yard, showing us various rat burrows in the yard, and examples of traps that are currently deployed. Specifically, Dr. Pollack showed us a trap that was designed to look like a fake rock, opening it up to show the killing mechanism inside. He also showed us burrows that were marked with red flags, signalling them for treatment by pest control.

From all of the information given to us from both the presentation and the walk around the yard, we were able to form an idea in our minds of the severity of the pest issue at Harvard as well as various factors that we should consider when thinking of ways to approach the problem. There were many more people involved with the pest issue than just the pest control people, so we knew after this initial discussion with Dr. Pollack to be aware of the fact that students, faculty, administration, parents, and the general public would all have an interest and stake in this issue. From this point forward, we set out to map out the various stakeholders and aspects of the issue to consider. Dr. Pollack was around both in person and via Slack to answer any questions we had during this process

Systems map

A systems map is a diagram that breaks down a larger problem into smaller subcomponents, so that it's easier to visualize what has an effect on what, and what aspects of the larger problem are the most important to address. In the making of our systems maps, we thought of any and all aspects of the problem that had been brought up by our client Dr. Pollack, as well as any others ideas we had.

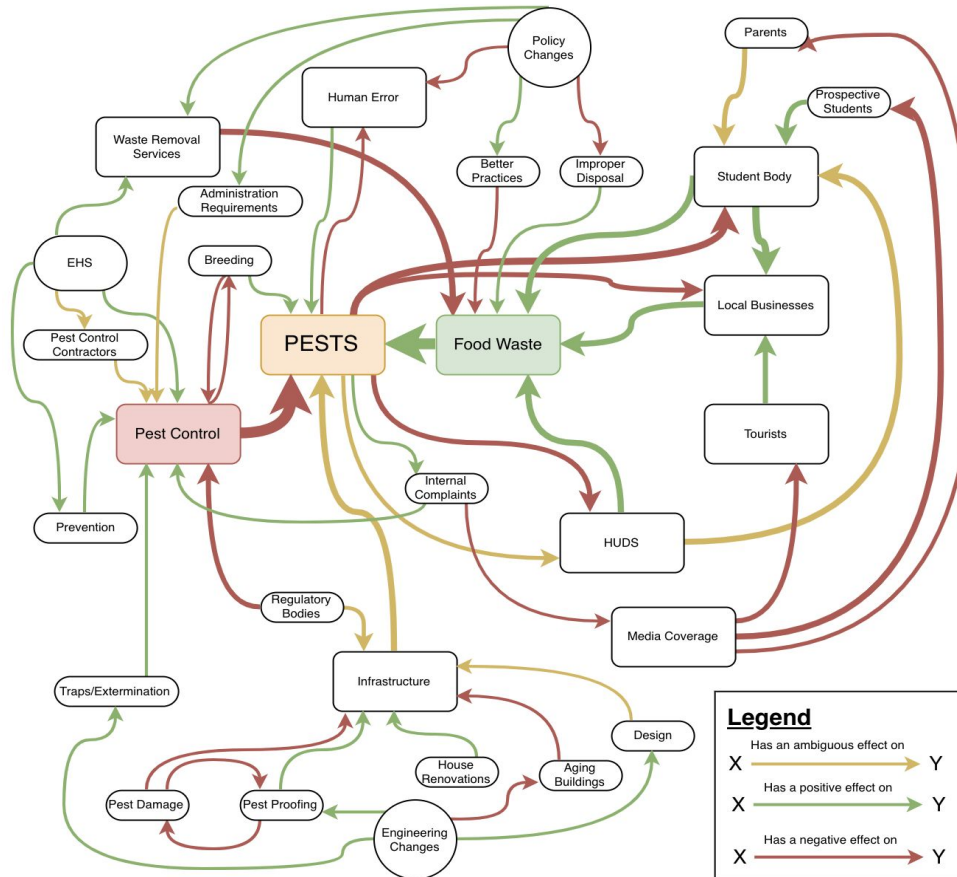


Figure 1: Pest Problem Systems Map

While complex, the systems map makes clear the nodes that are most central to the pest control problem: food waste, the infrastructure, and pest control, among others. The color coding, explained in the legend, shows the kind of relationship that two nodes have with each other. To arrive at the systems map, we started off with a list of the various nodes that we wanted to include. From there, we each took three to four nodes, and did research about the effect that it might have on other nodes. We then split into groups of four and each group made its own systems map. The one shown above has elements of all three groups, and reflects the best version of the systems maps that we made. For the rest of the project, the systems map served as a good reference for all of the solutions and designs we considered.

Stakeholders and influence map

The next step was to make a stakeholders and influence map, which is similar to the system map in that it maps out different sections of the larger issue. The only difference here is that the stakeholders and influence map has entities of people as nodes, rather than processes or actions.

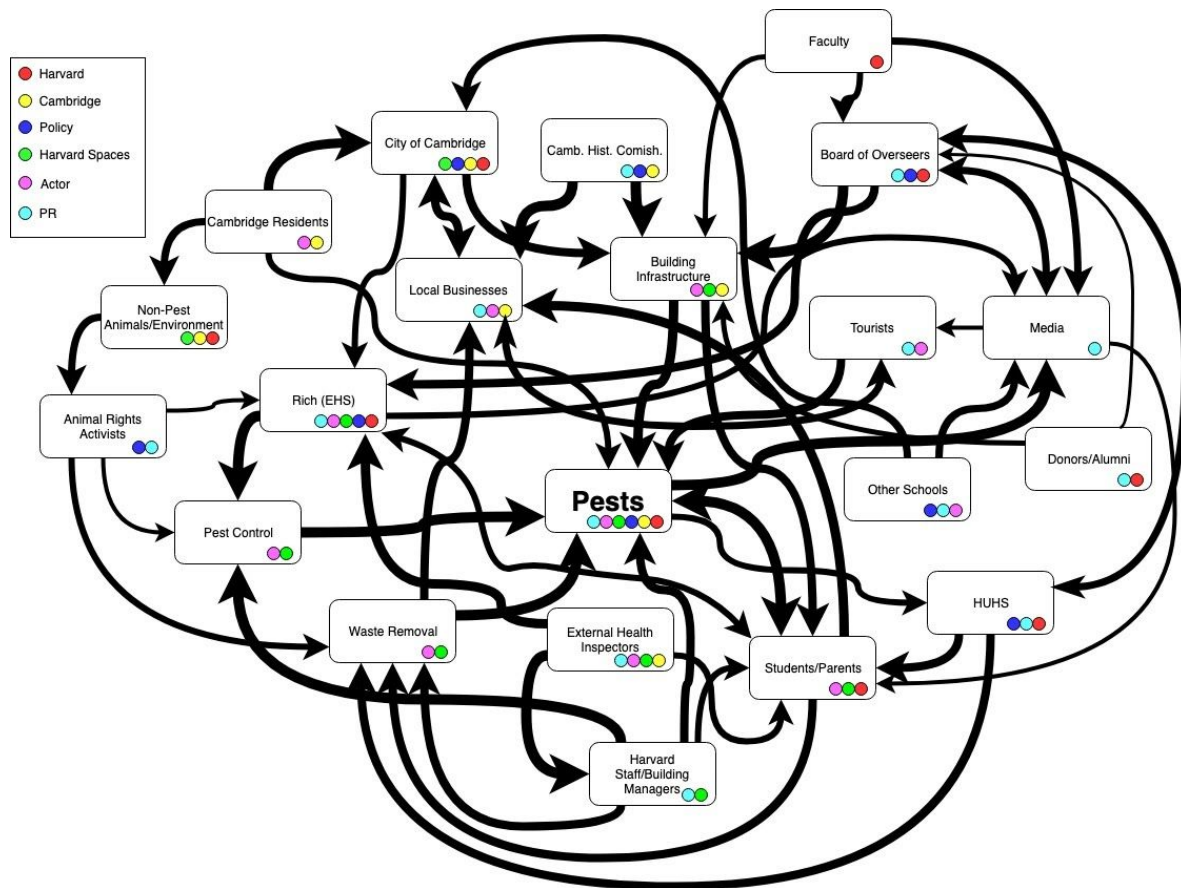


Figure 2: Pest Problem Stakeholders/Influence Map

Equally as complex as the systems map, the stakeholders and influence map shows the plethora of individuals involved in ameliorating the pest problem at Harvard. There are a couple of things we did to help the map display more information than just a interconnected jumble of lines. First, the thickness of the lines reflects the relative magnitude of the influence that the different stakeholders have on each other. This is the influence part of the map. Second, we came up with broad categories for different solutions, and placed each of the stakeholders in as many of these broad categories that they fit. The color coded dots reflect this.

In order to generate this map, we were each assigned two nodes, and were tasked with researching the influence that each of these nodes would have on every other node. We did research on the internet in order to determine the influence that one stakeholder would have on another, and a few of us made site visits (restaurants, for example) or phone calls to the stakeholders. Here is an example of that research, looking at the impact that the City of Cambridge has on other stakeholders:

- City of Cambridge (Cambridge, Policy, Harvard, Harvard Spaces)
 - 5 -> Other schools (In Cambridge)
 - Other schools in Cambridge aren't controlled in every way by the municipal government, but they have to follow the general city guidelines (health, safety, accessibility, etc.)
 - 2 -> External Health Inspectors
 - They're both from the government, just very disparate branches of it. Same team; don't really control each other.
 - 8 -> Buildings and Infrastructure
 - The City has strict guidelines on building new infrastructure, and requires permits to be obtained: <https://www.cambridgema.gov/inspection/buildingelectricplumbingpermits/howtoapply/buildingpermits>
 - 1 -> Donors/Alumni
 - The fact that Harvard is in Cambridge hasn't changed ever, and likely won't affect donations or alumna attitude
 - 8 -> Businesses in the Square
 - Similar to the Buildings and Infrastructure category, businesses in the square must adhere to Cambridge's guidelines in areas such as food safety, building structure, sanitation, etc.
 - 6 -> Rich/EHS
 - The City dictates certain pest control practices that can't be used in certain situations (e.g. no carbon monoxide close to a place of residence)

Figure 3: Influence Research

Other research

Beyond the research that went into making the systems map and the stakeholders and influence map, we also did our own research in trying to gain a better understanding of the problem. Examples of additional research that we did include sending a survey to one of the houses to get a sense of eating habits, including whether students like to eat in the dining hall or bring food up to their rooms. We also look at some redacted records that our client Dr. Pollack was able to provide us that gave us a sense of current pest reporting methods.

Problem statement

Having put together all of this research, we as a group now had a strong understanding of the key issues in the larger mouse and rat pest problem at Harvard. As a final ending to our work in the problem space, we set out to write a problem statement, which would be a concise outline of the pest issue, and the specific things we would target in addressing that larger pest issue. To have multiple options for a problem statement, we split into three groups, with each group drafting their own problem statement. Similar to the stakeholder map, the final problem statement was a combination of the three different problem statements that were drawn up. Here is our problem statement:

“Pest populations on Harvard’s campus negatively impact public perception, infrastructure, and human health. These populations are **highly dependent on access to food for survival**. Current pest control efforts are targeted at reducing the number of burrows and pest-proofing campus

infrastructure, **but as long as food sources exist, pests will continue to infiltrate our campus. In the short term, more effective pest control techniques** could be implemented to suppress the problem, but pests cannot be adequately suppressed until their food sources are removed.

How might we improve the effectiveness of pest control methods and decrease pest access to food sources at Harvard?"

This problem statement, which emphasizes access to food, would be a guide for us as we moved forward into the solution space.

V. Investigating the Solutions Space

Ideation Phase and Divergent Solutions

Once we settled upon our problem statement, we entered into our ideation phase. The purpose of this phase was to generate potential solutions that addressed our problem statement as effectively as possible. To facilitate this, we sat as a large group and brainstormed, recording every suggestion and idea no matter the practicality or effectiveness. We were taught that this was the divergent solution space. The reasoning behind having this space is that we take every idea and evaluate it later in our convergent solution selection because all ideas could potentially lead to something worthwhile. Our list of divergent solutions was as follows:

- Trash Can Trap
- Rat Sterilization
- Waste Disposal Fine
- Introduce more cats to campus
- Data/motion tracking + Database (Filtering)
- Pest I.D. Course
- More robust cans
- Policy to only eat in Dining Halls
- More trans cans with lids
- Trace amounts of rat poison left in human food
- Daily waste removal
- Hawk wildlife reserve
- Mobile rat trap (more active + surveillance)
- Food Scent Trap

- Mitigate rats’ sense of smell
- Store food in a deoxygenated environment
- Artificial rat sebum trap
- Ways to ease job of pest control (ex. Rat burrow mapping)
- Waste disposal incentive program
- News stories (positive) about Harvard’s efforts (last step in a multi-step approach)
- Insulation with poison for holes in buildings

As one can see, we ended up with a lot of ideas, and we needed to whittle our list down considerably. We did this by entering our convergent solution space.

Criteria and Process for Convergence

Our group’s objective when beginning the process of convergence was to identify which ideas could effectively be evolved into solutions that both address our problem statement and advance the pest control needs of Harvard Environmental Health and Safety. To meet this goal, we first analyzed our list of divergent solutions at a high-level to identify the overarching commonalities between certain ideas. This allowed us to categorize our ideas based on general intervention strategies and see where many of them fell (Table 1).

Policy	Data
Students Only Eating in Dining Halls	Statistics on Rat Sightings Motion Tracking of Rats Pest Identification Course Database Improvements Burrow Mapping
Food/Waste	Traps
Improving Trash Cans Improving Food Storage Methods More Effective Trash Removal	Mobile/Active Rat Traps Trash Can Traps Food Scent Traps

Table 1: Grouping of divergent ideas to identify the predominant intervention strategies

We then worked to establish criteria that would allow for us to objectively assess the viability of each divergent solution. We did this by first identifying broad categories that should be optimally met for a useful solution; these categories were informed by our exploration of the problem space, our client’s specific needs, and achievability given our time frame. This led us to identify seven relevant categories and their guiding questions:

1. Effectiveness

Does this idea address our problem statement? Can we measure the success of this idea?

2. Practicality

Can we achieve this in our time frame? Is this idea acceptable, realistic and scalable?

3. Safety

Is this idea safe? Are there any relevant environmental and/or health concerns?

4. Total Cost

How much will it cost to design and implement this idea on a broad scale?

5. Engineering Interest

Does this idea harness the engineering abilities and skill set of our team?

6. Sustainability

Can this solution be sustained and relied upon in the future?

7. Perception

How will the media and relevant stakeholders view this idea and its implementation?

After establishing the assessment criteria, our group worked to identify a grading method that would allow us to rank each idea against the others. This was an iterative process, as we first used a metric scale with scores ranging between 1 (worst case) and 5 (best case) and tailored score descriptions to each category. We however did not find that this strategy correctly identified the most optimal solutions for convergence. Furthermore, while all seven of these categories highlight things that should be considered when narrowing our solutions, they are not of equal significance and thus should not be weighted equally.

We therefore opted for a linear evaluation system that organized our criteria into different tiers according to their level of importance (Figure 5). This allowed for us to delineate which criteria should fundamentally be satisfied for the solutions we converged upon. We further agreed that each finalized solution should pass tier one by being both effective and practical as a prerequisite for continued evaluation at tiers two and three. This helped validate our assessment process by giving more importance to these very key aspects.

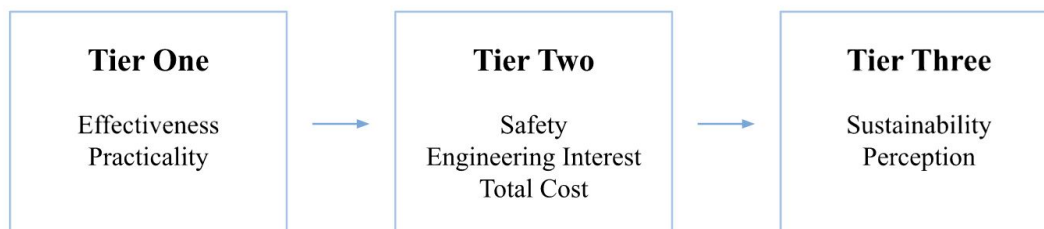


Figure 5: Tiered linear evaluation system for convergence where all convergent solutions must pass tier one to be evaluated at the following tiers; each solution passes a tier after successfully satisfying each category.

Each potential solution was qualitatively evaluated at each category (via yes or no), instead of being assigned a numerical score to prevent the original imbalances we noticed between different categories. Executing this evaluation system resulted in ideas like introducing more predatory animals onto Harvard's campus to be filtered out due to impracticality, and finally allowed for our group to converge on two solutions that address our two-pronged problem statement:

(1) Engineering a More Robust Rat Trap for Trash Can Retrofitting

(2) Improving Harvard's Pest Control Database via Data Tracking and Visualization

Our first solution is something that successfully passed all tiers of our evaluation system and could be realized in the short-term and serve as a more active pest control approach. We were initially hesitant about how our second solution would be perceived by relevant stakeholders due to its emphasis on collecting pest data. However, we realized that these concerns could be addressed if this data was collected securely and only completely shared with relevant parties, like Harvard EHS. This solution is something we believe could be realized in the long-term and serve as a proactive pest management strategy.

VI. Solutions

VI.A. Solution 1 - Trash Can Trap

The Idea

The first idea that we felt fulfilled our convergent ideation process benchmarks was to build a better version of a rat trap that would more effectively reduce rat activity around Harvard's trash. This would fulfill the second part of our problem statement - "How might we ... decrease pest access to food sources at Harvard?"

Our motivation came from many videos that our client showed us of rats avoiding traditional rat traps. We wanted to build something that would more effectively trap rats, and could be placed in areas that provide rats on campus with one of their main sources of food - trash bins, such as the one pictured below.



Figure 6: Standard rolling trash bin

As we refined our idea, we came up with the following features that we wanted our design to have:

- Would be easily incorporated in some way to the existing trash bins around campus.
- Would effectively trap and kill rats and mice attempting to use trash bins as a food source.
- Would be easily reset and emptied of dead rats.
- Would not inadvertently grant rats access to the food within the trash can.

From here, we began our design process.

The Design Process

Our first design was very basic. It would be a box that attached to the underside of the trash bin. In the top would be a one way entry that the rats would be able to enter through, but not exit. The box would have a hinged bottom through which dead rats could be disposed of. There would be air holes in the box that would allow the smell of trash/food to waft through and attract rats, removing the need for bait that would need to be replaced. The part that we still needed to figure out was the killing mechanism.

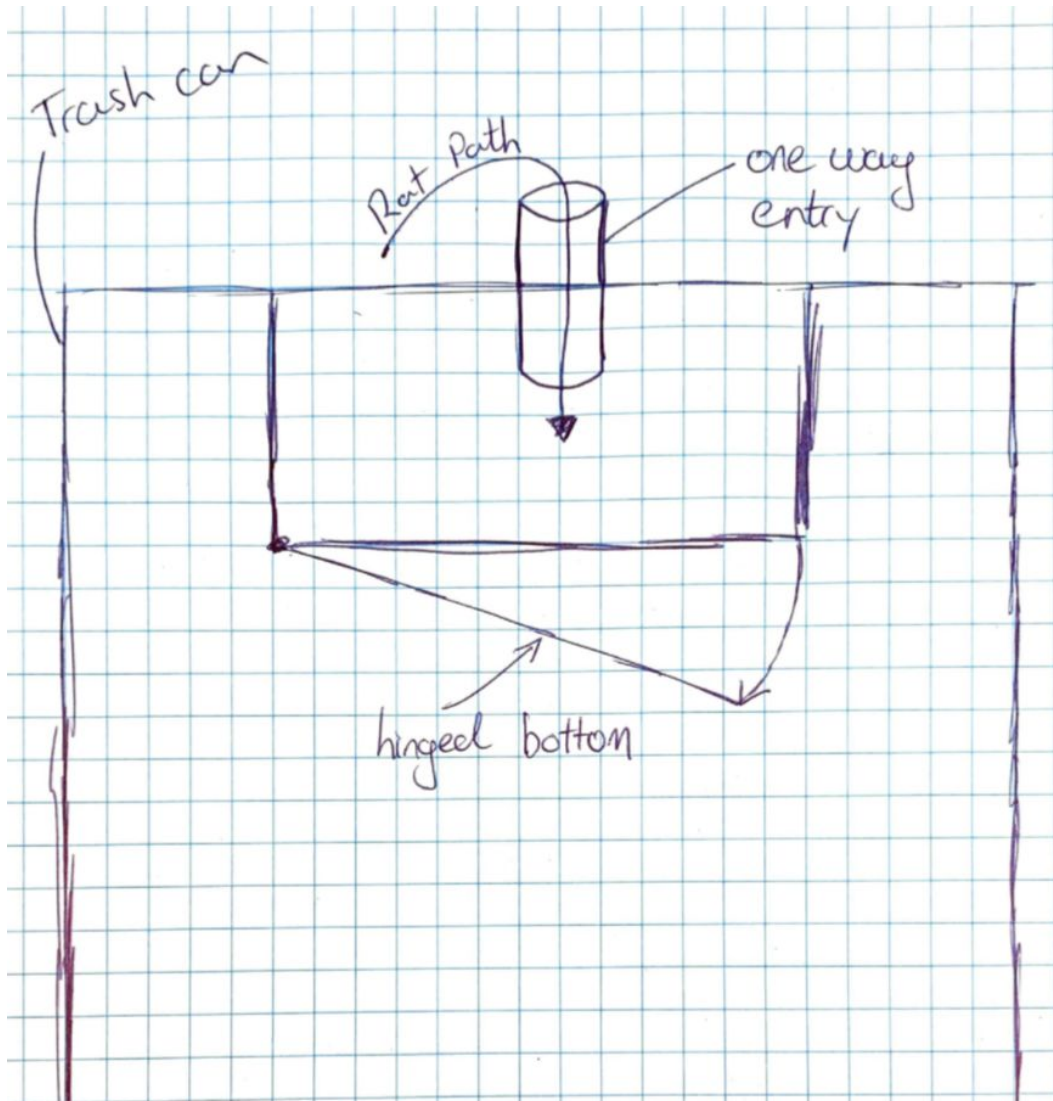


Figure 7: Retrofit trash can rat trap sketch

Killing Mechanism

With the key design features figured out, we needed to decide how the rats would actually be killed. The following options were discussed:

- **Poison:** While poison is a proven effective method of killing rats, it had some downsides for us. First, it would eventually need to be replaced, which is something that would negate our easy reset feature. Secondly, the University administration was very strict in limiting the use of poison on campus to very specific applications, and thus would be unlikely to approve its use in these traps.
- **Electricity:** This was discussed briefly, but it became obvious very quickly that there would be too many legal and safety limitations for this to be feasible.

- **Traditional Rat Trap:** This was considered to be used either as originally designed or augmented in some way. The advantages of using this would be the traps inexpensiveness, proven track record and ability to be slightly modified. In the end it was decided against due to too much complexity in the reset mechanism.
- **CO₂ Powered Piston Trap:** We found a trap that is designed by a New Zealand company (<https://goodnature.co.nz/>). The biggest advantage here was the easy reset, but in the end it was deemed far too expensive to be feasible for our use.
- **T-Rex Rat Trap** (pictured below). We eventually settled on this trap as it was relatively inexpensive, proven to be effective in trapping and killing rats, had a hole in the bottom near the trigger mechanism that would fit with our air hole design, and also had a one-motion, easy reset mechanism.



Figure 8: T-Rex Rat Trap (engaged position)

Prototype

For our first prototype, we created a one-way entrance using an inch of 2'' diameter PVC and stiff wires. We folded one end of the stiff wire around the edge of the PVC and glued the end in place while leaving the other end hanging about 3 inches off the back of the piece of PVC, by doing this we allowed for an easy and smooth entrance for the rat to get into the trap but if the rat were trying to escape they would be confronted by the stiff ends of the wire and would be unable to leave. We then created a box 30 x 24 x 13 centimeter box out of ¼ inch acrylic. These dimensions were chosen to give rats space to move around and into the box and to give enough room for the two T-Rex traps added at the back of the box. The height of the box was chosen due to the height of the trap when it was primed and set to be used being 12 centimeters. The base of

the prototype would be hinged in place at the back of the trap(1) and would be held closed by a magnetic door catch at the front of the trap that had a 60 pound pull strength(2). Slots were cut in the sides of the sides pieces of the wall so that we could jigsaw them together using acrylic glue(3). Holes were added to the base plate to screw the traps in place using 4-40 screws and hex nuts(4). A larger hole was placed underneath each trap such that the smell from the trash in the trash can could waft through and entice the rats(5). At the back side of the box, there are two rectangular cutouts that allowed for the T-Rex traps to sit partially out of the back of the box (6) such that a metal rod can be attached to the lid directly behind the box and over the latch of the T-Rex trap. This metal rod would press against the T-Rex trap latch as the box is opened to reset the T-Rex trap and simultaneously empty the dead rats out of the box(7, not included in picture but shown for effect of where rod would go, see prototype 2 for a better picture).

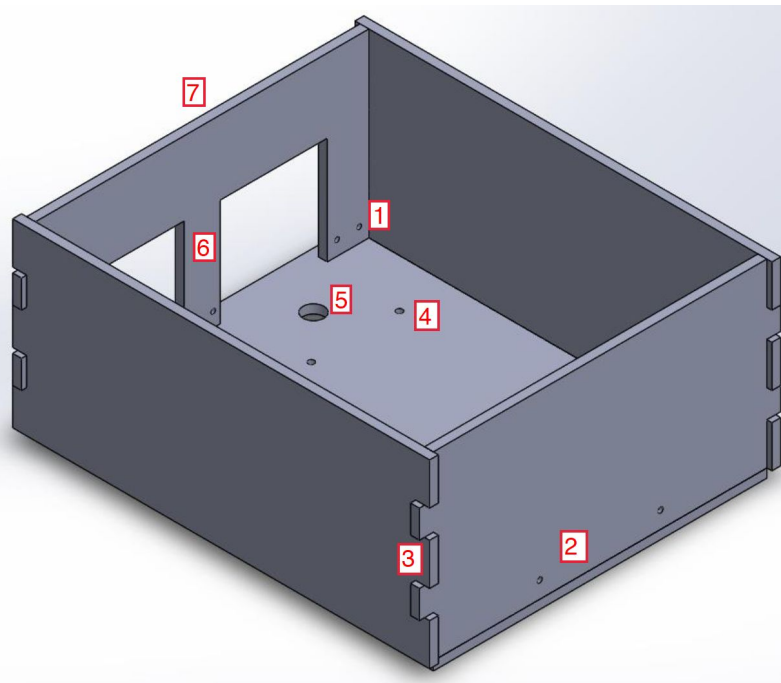


Figure 9: CAD rendering of prototype trap

Field Testing

Returning to the attachment of the trap to the trash bin, for our initial testing phase we drilled a hole on the lid of the trash bin, allowing us to attach the one-way valve which was constructed using PVC pipe and stiff wiring. We then drilled smaller holes around the lid in order to attach the trap using zip-ties. Using these zip-ties provided us enough strength to hold our trap in place but in our second iteration we looked for a more effective design. For this prototype we did not test our reset mechanism, as our main concern in our field test, was testing our concept's ability to bait rats into coming into our trap and its ability to kill these rodents. With the attachment complete, we set out to test our mechanism. We used a trash bin provided by Harvard sanitation department, deposited odorous food waste inside to provide the built-in

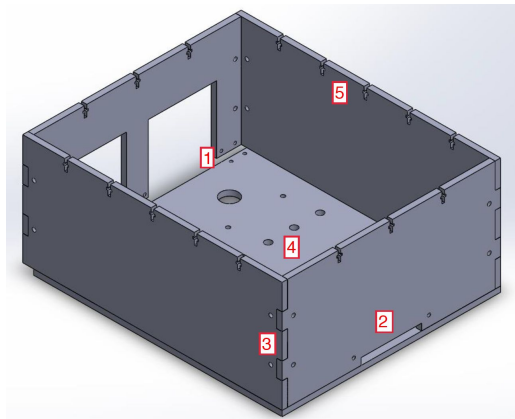
bait, and placed the trash bin in a location across the river with the help of our client. We collected data by placing motion-activated cameras around the trash can which was able to capture a number of rats, as well as squirrels and birds, intrigued by the trash bin and searching around the one way valve. Sadly, we have not seen any rats killed by our trap, but our client, Dr. Pollack, explained that rats are neophobic and as such they do not like trying new things such as going down a one-way valve in a new trash bin. But Dr. Pollack also believes that if the trash bin were to remain on site longer we would see rats begin to dive in. Another complication with this test was that the bin was placed next to larger dumpsters that afforded rodents easier access to waste than with our trash can.



Figure 10: Images of testing trash can and video monitor

Prototype 2

While putting together the prototype with the trash can for our first field test we ran into some initial problems with the design. First was that the back of the box, which was hinged closed, interfered too much with the base and so for the next iteration the base was cut short so that the base could hinge open more easily(1). An additional cut out was placed at the front so the magnetic catch could be released more easily by the user(2). Additional holes were added at the corners so that metal L-brackets could be used to hold the box together, which was stronger and more reliable than the acrylic glue we initially used(3). Additional holes were added to the bottom of the base to allow for more trash smells to waft through the base(4). We added 16 T-slots to the top of the trap so that we can use 4-40 screws and hex nuts to attach the trap to the bottom of the trash can lid(5). We decided to do this since we realized that the trash can we obtained was made of a plastic that is very difficult to adhere any glue to, which is why we had to zip tie the trap in place for the trial. (right: Figure 11: CAD rendering of second prototype)



Last, in a live demo of the trap mechanism, we can see how the metal rod would be situated so that while the base is being opened, the trap would reset and simultaneously allow for any dead rats to drop out of the trap and into the trash can below(6).

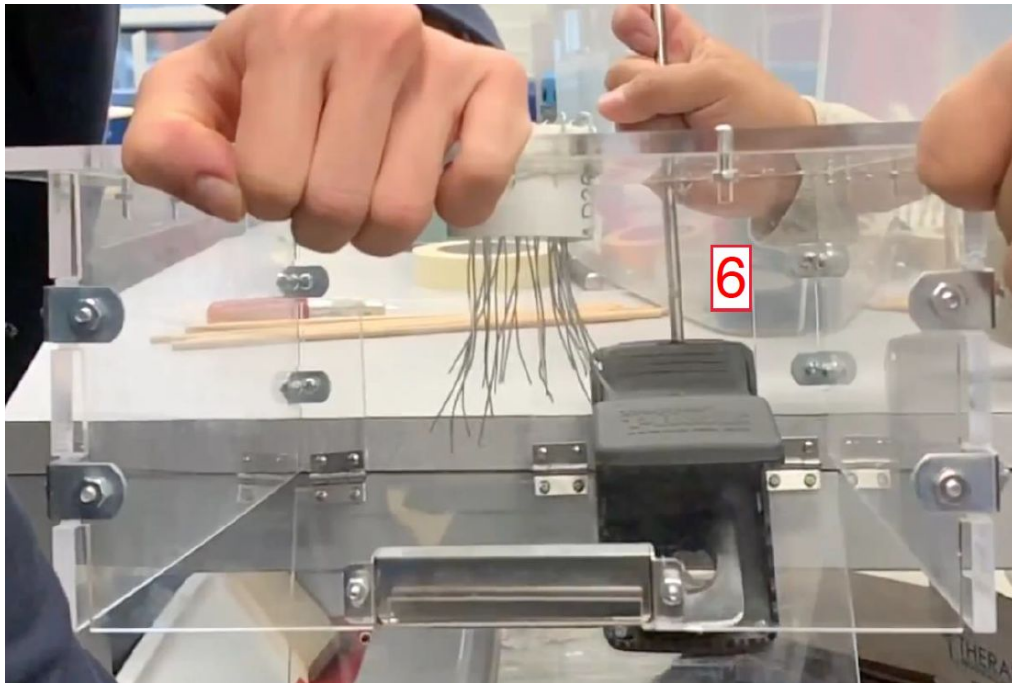


Figure 12: Side view of prototype

In total, our trap, outside of the cost of the trash can, would cost approximately \$48.60. This price would significantly decrease if the parts were bought at bulk prices but due to the small order of parts in total, the price is relatively steep at this point in time. For a detailed breakdown on price, see appendices at the end of this report.

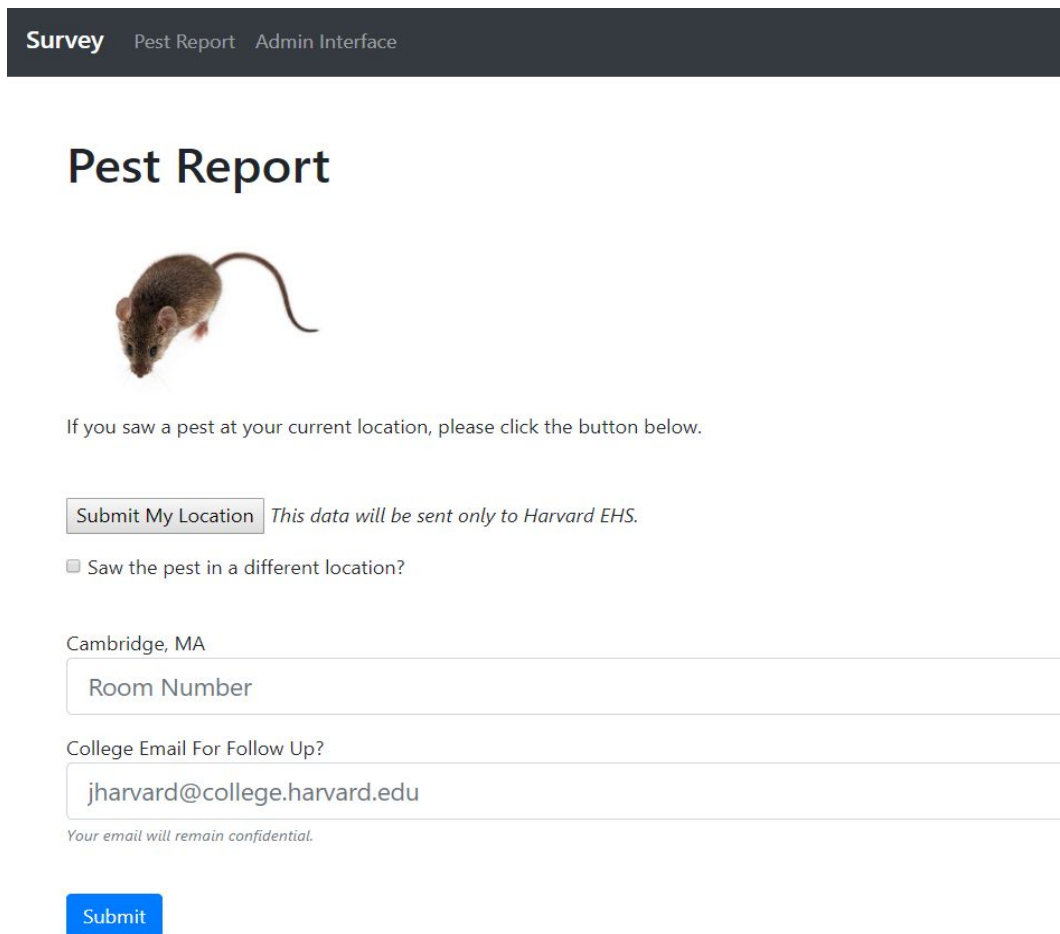
VI.B. Solution 2 - Data Tracking and Visualization

The Idea

The second solution that we wanted to implement was a way for our client to better collect and visualize data regarding pest tracking and human sightings of pests on our campus. Currently, when our client handles reports about pest issues, the pest notifications come in the way of several hundred emails on a weekly and sometimes daily basis. That is, rather than being in a database or spreadsheet that can be easily queried or kept for historical records, the pest data he gets is in a log jam of email chains. We wanted to fix this, but we also wanted to make a ways for pest data to be better obtained. Currently, if one sees a pest in their dorm or somewhere else on campus, they have to contact our client via email or report this issue to Yard Operations in a long web form. What we decided to create was a shortened web form that allows the user to click one button which automatically submits their geographic location. If the user wishes, he or she can add in their email and/or room number, and if they saw that pest at a different


location than where they are when filling out the pest form, they have the option of filling out the location manually. From another angle, we decided to make a improved motion detection sensor. This sensor, equipped with PIR detection and GPS, was designed to be placed in areas that human eyes can't see to test for the existence of pest activity. Together with the expedited web form, these two methods would help to crack down on what remains of the pest population at Harvard.

The Web Form:



Survey Pest Report Admin Interface

Pest Report



If you saw a pest at your current location, please click the button below.

This data will be sent only to Harvard EHS.

Saw the pest in a different location?

Cambridge, MA

Room Number

College Email For Follow Up?

jharvard@college.harvard.edu

Your email will remain confidential.

Figure 13: Screenshot of beta pest report web form

General Functionality:

The app that we created runs using a python-written flask framework that is served by an Amazon web server called Elastic Beanstalk. Whenever a user goes to crimsonpest.com and submits their location via the blue 'submit' button, the user data is redirected by the server to the database where the entry and all its information is logged for safe-keeping. The information

from the database can be queried from the web app on the administrator's interface or it can be used to automatically populate a map with pins showing where the pest issues were reported. This map can be chronologically tailored to show reports that occurred in the last 24 hours, last 7 days, last month, etc. Likewise, when the motion detector picks up a reading that something has moved in front of it, it's always connected to local internet, so it directly logs data into the same database as the user sightings. The following section lists more information on the details that allow our app to run properly.

Amazon Web Service (AWS) Library:

Elastic Beanstalk

Cloud operator that serves the code that runs the website. Controls how the website behaves and looks but does not store any of the data collected from users.

Dynamo DB

Cloud database that stores all of the data collected from the survey website and the motion detector. Sends this data to the Elastic Beanstalk website for mapping and data visualization purposes.

Route 53

Controls the handling of the 'crimsonpest.com' domain name. Re-routes traffic from 'crimsonpest.com' to the Elastic Beanstalk operator.

Amazon EC2

Controls the region and load balancing aspects of the Elastic Beanstalk operator. Allows for additional customization and security within the Elastic Beanstalk operator.

Certificate Manager

Used to verify authenticity of the website. Provides an authentic security certificate so that the website can serve requests on secure (HTTPS) ports.

Cloud9

Virtual terminal used for beta testing of backend code for eventual FLASK application that controls the survey / visualization website.

AWS EB CLI (Command Line Interface)

Local command package that was used to communicate directly from technician's computer to Elastic Beanstalk operator, used for uploading new file packages to cloud servers.

AWS Route Account Login Credentials:

Username: Harvardpest@gmail.com

Password: Richrats2019

AWSAccessKeyId=AKIAJQNXMLM3SKUWAVGAQ

AWSSecretKey=oTAtUT7W7MWxexqJIBE7u57eS0ihD/aTH1Ns/PK3

Building the Web App:



Configuration

Python 3.6 running on 64bit

Amazon Linux/2.8.2

Newer version available

Change

The web app was designed on a Cloud9 cloud terminal server in order to streamline collaboration and debugging. After experimenting with several methods of website implementation. The web-app team decided it would be best to hardcode the proposed website using a FLASK (v.1.0.2) package running in Python 3.6. A full list of current

system requirements is produced below and was passed to the Elastic Beanstalk server via the EB CLI in a requirements.txt formatted file:

Package	Version
boto3	1.9.137
botocore	1.12.137
Click	7.0
cs50	3.1.0
DateTime	4.3
docutils	0.14
Flask	1.0.2
itsdangerous	1.1.0
Jinja2	2.10.1

jmespath	0.9.4
MarkupSafe	1.1.1
python-dateutil	2.8.0
pytz	2019.1
s3transfer	0.2.0
six	1.12.0
SQLAlchemy	1.3.3
sqlparse	0.3.0
termcolor	1.1.0
urllib3	1.24.2
Werkzeug	0.15.2
zope.interface	4.6.0

Table 2: system requirements

Once the full backend code package was completed in the Cloud9 server it was downloaded as a compressed ZIP format file for implementation. Using the EB CLI, a virtual environment was created in the Elastic Beanstalk operator. Within this virtual environment, a specific server was created for the website to which the code package was deployed.

```
Aidans-MacBook-Pro-6:~ aidancrawford$ export PATH=/Users/aidancrawford/.local/bin:$PATH &&
cd v2/eb-flask && source virt/bin/activate && eb
usage: eb (sub-commands ...) [options ...] {arguments ...}

Welcome to the Elastic Beanstalk Command Line Interface (EB CLI).
For more information on a specific command, type "eb {cmd} --help".

commands:
  abort          Cancels an environment update or deployment.
  appversion     Listing and managing application versions
  clone         Clones an environment.
  code          Configures the code source for the EB CLI to use by default.
  config       Modify an environment's configuration. Use subcommands to manage saved confi
gurations.
  console      Opens the environment in the AWS Elastic Beanstalk Management Console.
  create       Creates a new environment.
  deploy      Deploys your source code to the environment.
  events      Gets recent events.
  health     Shows detailed environment health.
  init       Initializes your directory with the EB CLI. Creates the application.
  labs      Extra experimental commands.
  list      Lists all environments.
  local     Runs commands on your local machine.
  logs     Gets recent logs.
  open     Opens the application URL in a browser.
  platform Commands for managing platforms.
  printenv Shows the environment variables.
  restore  Restores a terminated environment.
  scale   Changes the number of running instances.
  setenv  Sets environment variables.
  ssh     Opens the SSH client to connect to an instance.
  status  Gets environment information and status.
  swap   Swaps two environment CNAMES with each other.
  tags   Allows adding, deleting, updating, and listing of environment tags.
  terminate Terminates the environment.
  upgrade Updates the environment to the most recent platform version.
  use     Sets default environment.

optional arguments:
  -h, --help            show this help message and exit
  --debug              toggle debug output
  --quiet              suppress all output
  -v, --verbose        toggle verbose output
  --profile PROFILE    use a specific profile from your credential file
  -r REGION, --region REGION
                        use a specific region
  --no-verify-ssl     don't verify AWS SSL certificates
  --version            show application/version info

To get started type "eb init". Then type "eb create" and "eb open"
(virt) Aidans-MacBook-Pro-6:eb-flask aidancrawford$ eb
```

The command sequence for initialization of this virtual environment FLASK server from within the EB CLI can be detailed as followed:

```
~$ mkdir eb-flask
```

```
~$ cd eb-flask
```

```
~/eb-flask$ virtualenv virt
```

```
~$ source virt/bin/activate
```

```
(virt)~/eb-flask$ pip install flask==1.0.2
```

```
(virt)~/eb-flask$ pip freeze > requirements.txt
```

```
~/eb-flask$ eb init -p python-3.6 flask-app --region us-east-2
eb create web-env
~/eb-flask$ eb open
```

More detailed instructions on Elastic Beanstalk FLASK service by the EB CLI can be found here:

<https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create-deploy-python-flask.html>

Additional Necessary Packages:

Python v.3.6 - base coding language

Pip - easy package installer

Virtualenv - virtual environment generator package

Route 53 Domain Hosting:

Amazon's Domain service, Route 53 was used to purchase the domain name "crimsonpest.com" This domain is intended to be the outward-facing domain for the Elastic Beanstalk operator. Using a simple canonical name (CNAME) routing policy, the Route 53 service was instructed to reroute all incoming traffic requests to the application load balancer for the Elastic Beanstalk operator.

Domain Name: crimsonpest.com.

Type: Public Hosted Zone

Hosted Zone ID: ZA1XV2K0KT2N

Record Set Count: 3


Comment: HostedZone created by Route53 Registrar

Name Servers *: ns-1864.awsdns-41.co.uk
 ns-493.awsdns-61.com
 ns-659.awsdns-18.net
 ns-1265.awsdns-30.org

** Before the Domain Name System will start to route queries for this domain to Route 53 name servers, you must update the name server records either with the current DNS service or with the registrar for the domain, as applicable. For more information, click the ? icon above.*

Tags: View and manage tags for your hosted zones using [Tag Editor](#)

Query Logging: [Configure query logging](#)

Name: www.crimsonpest.com. 

Type: CNAME – Canonical name

Alias: Yes No

TTL (Seconds): 300 1m 5m 1h 1d

Value: awseb-e-f-AWSEBLoa-MY2BW57LLB7S-849790538.us-east-2.elb.amazonaws.com

The domain name that you want to resolve to instead of the value in the Name field.
 Example:
 www.example.com

Routing Policy: Simple

Route 53 responds to queries based only on the values in this record. [Learn More](#)

Via Amazon’s certificate manager, www.crimsonpest.com, and its associated elastic beanstalk operator, was issued a DNS-verified security certificate under the following identifier: a0485a21-e13e-45a3-96ec-38ea3995d9ce
 This certificate allows the website to be served securely under HTTPS protocol as well as standard HTTP protocols.

Load Balancing:

The group decided to implement the Elastic Beanstalk-served FLASK app using an application load balancer rather than a classic load balancer. Such a change will allow for easier implementation of backend-forced HTTPS routing by the user and will therefore make the site more secure. This will also provide the requisite amount of security required by the browser to allow for the geolocation program to run.

Data Storage and Visualization

In order to make our data solution useful for our client, we needed to be able to store pest sighting data in a way that is intuitive and easily retrievable. The current system, a large collection of emails, can make organization difficult in terms of track of report locations, status, time, etc. We wanted to make this process easier, so we began thinking of ways to tabulate data. After experimenting with many methods such as Microsoft Excel spreadsheets and SQL tables to

be written in Python, we settled upon a data management service named DynamoDB that was suggested by one of our team mentors, Joel.

DynamoDB is a program included in Amazon's Web Service repertoire that allows a user to create data tables in which he or she can store data. The benefits of using the web service included the fact that it was in Amazon's Web Service, making it easily interactable and compatible with using other Amazon Services. Additionally, one can interact with the service using a variety of languages including Python, Javascript, Java, Ruby, etc. making it relatively easier to work with than an obscure, specialized program. Finally, with our data project being a much smaller scaled project than those DynamoDB is equipped to handle, the service was free for our use, greatly contributing to keeping our project within the team's budget.

After deciding to use DynamoDB for our project, we began experimenting with the service, seeing how it worked and learning how to interact with it programmatically. We then began to create our table. Our table, named Pest_Sightings, is one with three columns: time, location, and info. The 'location' column is the primary key or hash attribute, the primary means in which data is sorted. It's inserted into the table using a [latitude,longitude] format. The 'time' column is the sort key or range attribute, the secondary sorting criteria. It allows us to organize our data further by sorting any values that may have the same primary key. Our time data is formatted using the ISO 8601 format (Ex. 2019-04-22 20:33:26.198920). Finally, our 'info' column consists of data that wouldn't be a good means to sort by, but is great for our client to be privy to. We included information such as student email address, room number (if submitted), and the source of the data entry (either the web form or the motion detector). What makes the 'info' column so useful is that additional information sections can be appended depending upon what information our client wishes to collect from users that report pest sightings. For example, if Dr. Pollack wanted the user to input whether they thought the pest was a mouse or a rat, the web form can be changed to reflect this new input and the database can accept the new information without making any changes to the other data in the table. Once we completed our table setup, we began to populate our table with test data. This step proved to be a great challenge as we needed to figure out how to communicate with our DynamoDB table from outside of Amazon's framework. After much research and testing, we were able to write to our data through non-Amazon interfaces such as our webform and the motion sensors. Below is an image of some of our table entries as well as an expanded data point.

<input type="checkbox"/>	Location i	Time
<input type="checkbox"/>	36.2048° N, 138.2529° E	2018-04-15T17:03:07+00:00
<input type="checkbox"/>	51.5074° N, 0.1278° W	2019-04-15T17:03:07+00:00

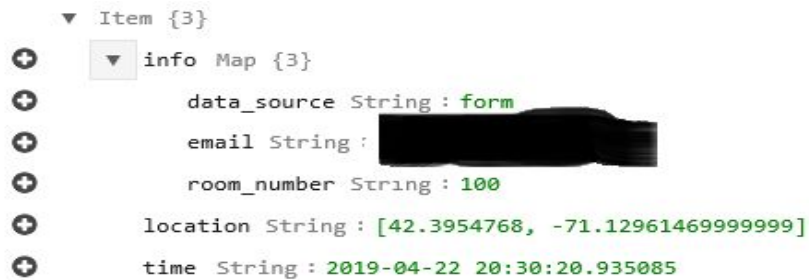


Figure 14: Database entry format

After populating our table with data, we then worked to be able to scan our table for all data entries. However, when doing so, we discovered that our data was not in the correct format needed to plot our data in Google Maps. We then wrote a function in Python to take our data and convert it into the GeoJSON format Google Maps needed.

```

def get_scan():
    response = table.scan()
    items = response['Items']
    # print(items)
    array = {"type": "FeatureCollection", "features" : []}
    #iterate through the data, appending each entry into an individual feature
    for i in items:
        array['features'].append( {
            "type": "Feature",
            "geometry": {
                "type": "Point",
                "coordinates": list(map(float, i['location'][1:][:-1].split(",")))
            },
            "properties": {
                "time": i['time'],
                'data_source': i['info']['data_source']
            }
        })

    #convert array into a JSON-formatted string
    geojsoned = json.dumps(array)

    return geojsoned

```

Figure 15: Python code for converting items to GeoJSON format

From here, we used sample code from Google Maps Javascript API in addition to some code we wrote in order to create a map marker and infowindow for each data entry point. Pictured below is a snapshot of our map depicting all data points in our table.

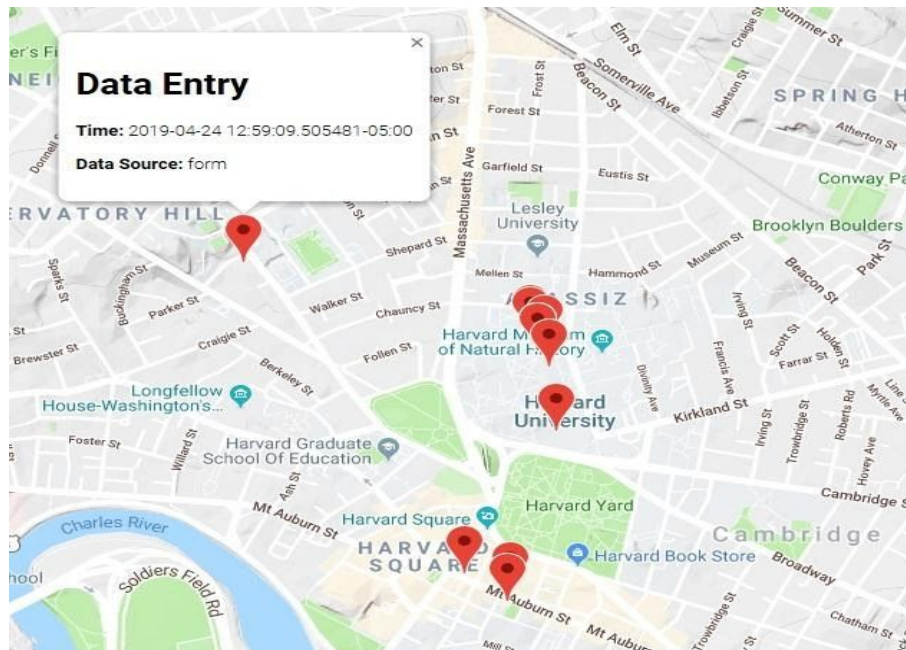


Figure 16: Map of Cambridge populated with test-points for pest data

While we were successful in implementing a working data storage and plotting system, there are some improvements we would like to make in the future. We would like to include more features such as the ability to filter markers based on time frame. This could allow our client to see points that were only generated 24 hours previously. We would also like a means to differentiate the motion detector marker from the web form markers in order to more easily visualize the sources of data detection. Finally, we would like to implement the ability to delete marker points from within the map. This would allow our client to delete points that have been successfully dealt with by EHS and Pest Control with the click of a button, removing the entry from both the map and the DynamoDB database. With these improvements, in addition to improved security measures to conform to standard web development practices, we are confident that our data visualization will be very useful to our client.

Motion Sensing Device

The other technique used to collect pest sighting data was a motion sensing device, designed to detect passing rodents and record each motion event to the DynamoDB database

described above. The device is capable of determining its location within a few meters using a GPS module, and can update the database with rodent sightings in real time.

This was done using a Raspberry Pi 0 W in connection with a battery, a PIR motion sensor, and a GPS module. The total costs of these components can be seen in the Appendix section. The Raspberry Pi 0 W was chosen for its compact form factor, low power consumption, wifi capability, and ease of programming. After initially using a monitor and keyboard to set up SSH, we were able to program it almost entirely from our own machines, accessing the Raspberry Pi's terminal through SSH. The PIR motion sensor used is very similar to the ones used in most automatic light switches. It senses a differential between two infrared heat sensors, and is thus able to detect motion - specifically from heat sources like humans or rodents. Though the sensor is designed for detecting humans, we were able to increase its sensitivity in order to pick up the smaller heat signatures associated with rodents. The output of the PIR sensor consists of a single data line that is driven high when motion has been detected, and then back low a few seconds later. This data line was connected to an interrupt pin on the Raspberry Pi, so that it can spend most of its time in a low-power sleep state, only waking up when a rodent is detected. The GPS module has a similarly simple connection scheme, requiring only power, ground, and a single data line from its data-out pin to the RX serial pin on the Raspberry Pi. All the Raspberry Pi had to do was monitor this line and parse the incoming GPS data in order to determine its position. Once it has its location, it can power down the GPS module on the assumption that the device will remain stationary until picked up and moved by an operator.

Any time the PIR sensor is triggered, the Raspberry Pi exits sleep mode, and sends a timestamp and location to the Amazon DynamoDB database, then powers down to wait for the next PIR triggering event. A diagram of the finite state machine implementation of this device along with sample code is included in the appendices. The whole device is housed in a laser-cut housing with a hole in one side for the PIR sensor, and enough room inside to fit all the rest of the components mentioned above.

The figures below shows the motion sensing device prototype encased in acrylic and pictured from different views.

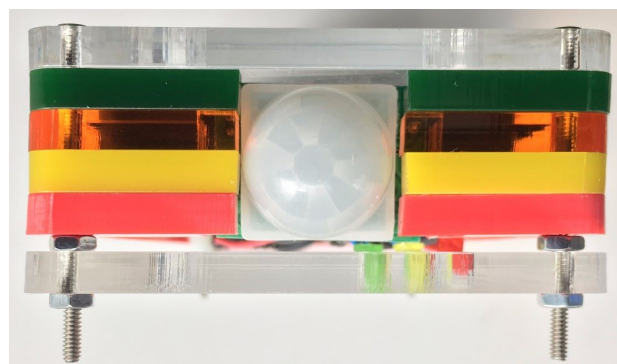


Figure 17: Front view of the motion sensing device prototype pictured from the end with the PIR sensor

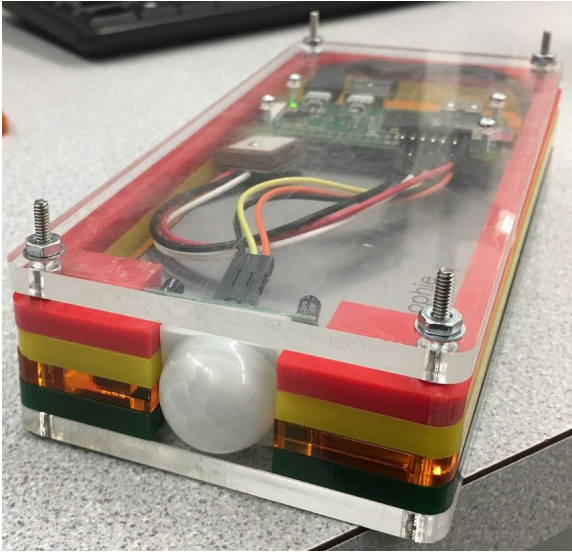


Figure 18: Isometric view of the motion sensing device prototype

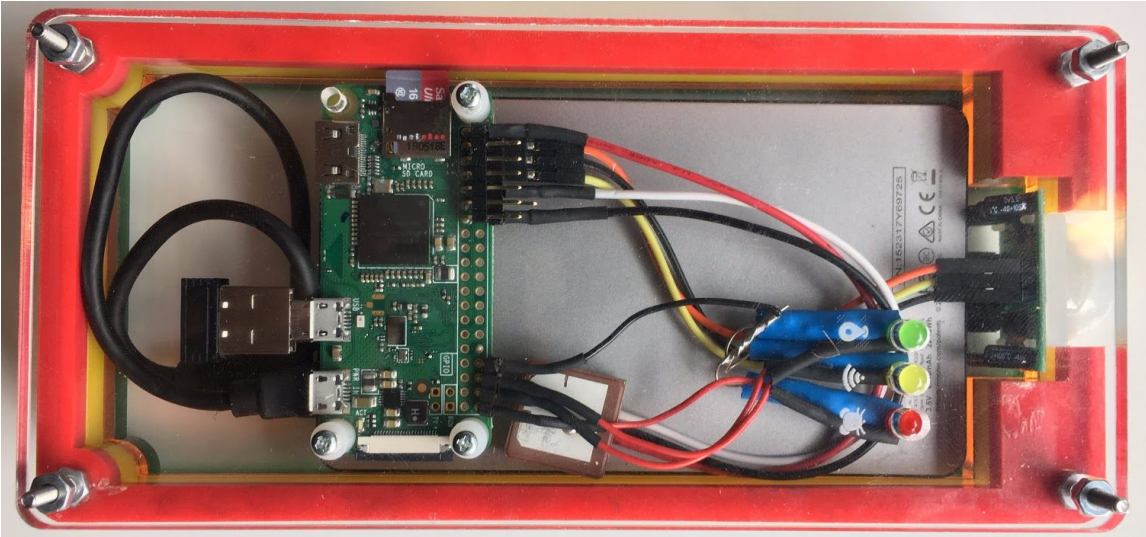


Figure 19: Bird's eye view of the motion sensing device prototype

VII. Conclusions -

During the early stages of exploring the problem space, the team developed the following problem statement: **How might we improve the effectiveness of pest control methods and decrease pest access to food sources at Harvard?** This statement helped us moving forward to view many options for our solution seeing as the question contains two parts; improving effectiveness of pest control methods, and decreasing pest access to food sources. After all the ideation the two solutions were chosen, a new design for a rat trap that is integrated into a trash can, as well as a database to better store data and give the client an easy way to view problem areas and hotspots around campus.

The first solution, the trash can trap, was developed to solve the part of the problem statement that dealt with pest access to food as well as more effective pest control. The innovative trap uses the trash and waste that already attracts pests as bait for the trap that kills them. This helps prevent pest access to food because instead of the rats eating trash and going back to their burrows they get trapped and ideally die. The trash can trap in theory is effective in solving a part of the problem we were presented with. In practice, there would need to be more data to show that it actually was effective in trapping and killing more rats.

The second solution, the data solution, was created to make gathering and viewing data and information easier for the client. Although, it does not directly solve the problem articulated, the team thought it was an important and essential step for creating effective pest control for Harvard's campus. Part of the problem statements says that we want to improve the effectiveness of pest control, and it is difficult to accomplish this without an easy and reliable method of analyzing the data of where the rats are located and how many there are. The data solution effectively allows all pest sightings to be organized into one place where it can be viewed by our client on a map of the campus. Although this solution does not directly eliminate any rats or pests from the area, it is essential to improving the effectiveness of pest control; knowing the right locations for the pest control companies to target greatly increases the effectiveness. With both of the solutions, we effectively targeted both sides of the problem that is articulated in the problem statement. With more time, and further tests we would be able to know more about how effective the designs are in the field.

VIII. Appendices Product specifications Detailed cost calculations

Trash Can Trap Detailed Cost

Item	Use in Trap	Price/Volume	Vendor	Amount used in trap	Cost in Trap
12" x 24" - Clear Acrylic Plexiglass Sheet - 1/4" Thick Cast	Used for sides and base of trap	\$10.49 per 12'' x 24'' sheet	https://www.estrplastics.com/1-4-x-12-x-24-Clear-Acrylic-Plexiglass-Sheet-p/1002501224.htm	2 sheets	\$20.98
WOOCH Door Magnetic Catch - 60lb High Magnetic Stainless Steel Heavy Duty Catch for Kitchen Bathroom Cupboard Wardrobe Closet Closures Cabinet Door Drawer Latch (3.74 in Silver, 2-Pack)	Magnetic catch to close base of box	\$7.99 for 2-Pack	https://www.amazon.com/WOOCH-Door-Magnetic-Catch-Stainless/dp/B07K6CT795/ref=sr_1_6?keywords=magnetic+door+catch&qid=1557433366&s=gateway&sr=8-6	1 catch	\$4
PVC-DWV Pipe, Schedule 40, Cellular Core, 2 in. x 10 ft.	1-way entrance	\$7.13 for 10 feet of PVC	https://www.homedepot.com/p/Genova-Products-PVC-DWV-Pipe-Schedule-40-Cellular-Core-2-in-x-10-ft-70021F/300282331	1 inch	\$0.06
Bend-and-Stay 304 Stainless Steel Wire Matte Finish, 5-lb. Spool, 0.02" Diameter	1-way entrance	\$64.55 per spool of 4,655 feet	https://www.mcmaster.com/8860k81	1.5 feet	\$0.02
Zinc-Plated Steel Corner Bracket	Corner brackets to hold sides together	\$0.43 each	https://www.mcmaster.com/catalog/125/2632	8 (2 per corner of box)	\$3.44

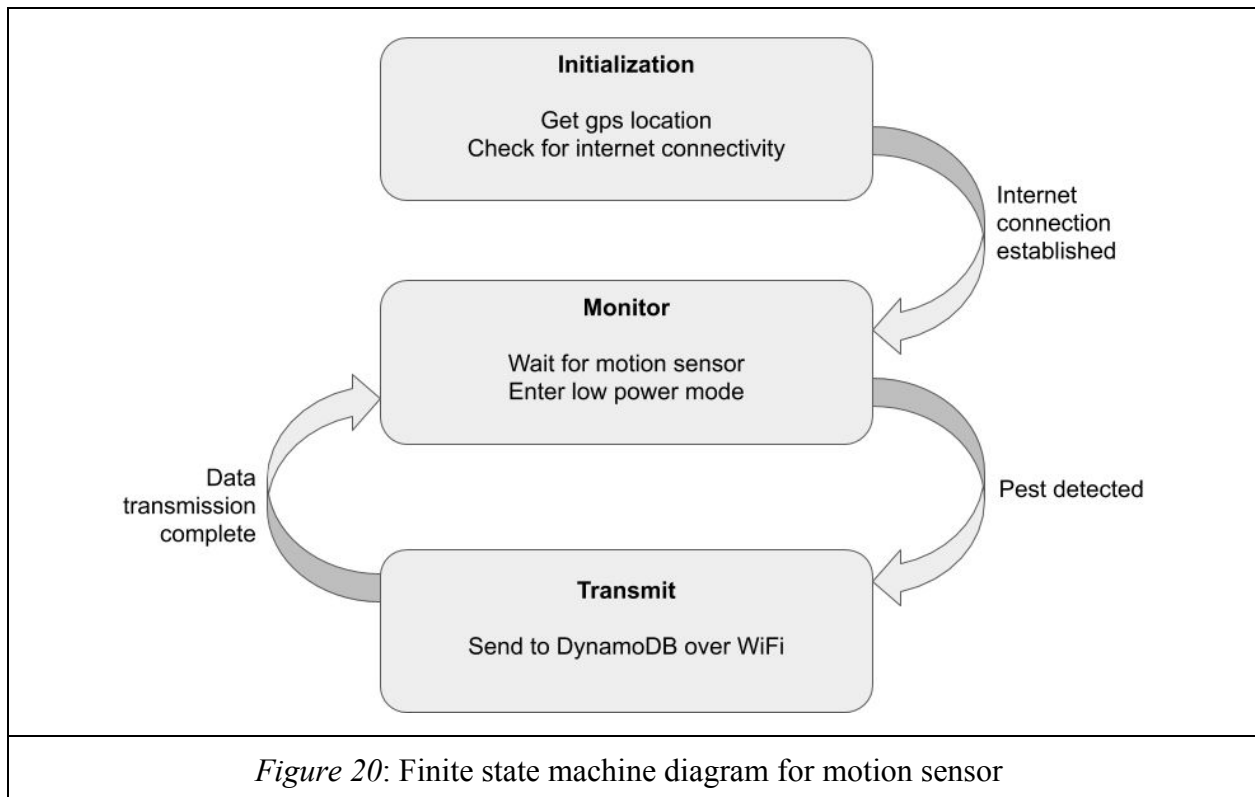
18-8 Stainless Steel Phillips Flat Head Screw Passivated, 6-32 Thread Size, 3/8" Long	Screws used with corner brackets to hold sides together	\$5.13 per pack of 100	https://www.mcmaster.com/91771a146	16 (2 in each corner bracket)	\$0.82
Low-Strength Steel Hex Nut Zinc-Plated, 6-32 Thread Size	Hex nuts used to hold bracket screws in place	\$1.28 per pack of 100	https://www.mcmaster.com/90480a007	16 (2 in each corner bracket)	\$0.20
Surface-Mount Hinge with Holes Brass, Nonremovable Pin, 1" x 1/2" Door Leaf	Hinges used at back of box	\$1.98 each	https://www.mcmaster.com/1603a23	3	\$5.94
18-8 Stainless Steel Phillips Flat Head Screw Passivated, 4-40 Thread Size, 1/2" Long	Screws used to attach hinges, T-slots, and T-Rex trap to base)	\$4.61 per pack of 100	https://www.mcmaster.com/91771a110	32 (12 in hinges, 16 in T-slots, 4 in T-Rex trap)	\$1.48
Low-Strength Steel Hex Nut Zinc-Plated, 4-40 Thread Size	Hex nuts used with hinges, in T-slots, and in attachment of T-Rex trap	\$0.89 per pack of 100	https://www.mcmaster.com/90480a005	32 (12 in hinges, 16 in T-slots, 4 in T-Rex trap)	\$0.29
Bell Labs Trapper T-Rex Rat Trap (4 Traps)	Traps at the back of the box, used as the killing mechanism for trap	\$22.74 per 4-pack	https://www.amazon.com/Bell-Labs-Trapper-T-Rex-Traps/dp/B0084X5ONI	2	\$11.37
Total					\$48.60

Motion Sensor Parts List

Item	Cost	Source
Raspberry Pi 0 W	\$10	https://www.raspberrypi.org/products/raspberry-pi-zero-w/

PIR Sensor	~\$10	Harvard EE Stockroom
GPS Module	\$15.95	https://www.sparkfun.com/products/13740
Battery	\$8.99	https://www.amazon.com/mophie-powerstation-External-Universal-Smartphones/dp/B07894KBQV/ref=sr_1_20?keywords=mophie+battery&qid=1557885626&s=gateway&sr=8-20
TOTAL	\$44.96	

Motion Sensing Finite State Machine and Sample Code



```

# try to acquire data from gps module
latitude, longitude, gps_time = GPS.get_gps_data()

```



```

location = "[" + latitude + ", " + longitude + "]"
print("GPS location: " + location)
# write location to local file
FM.write_data(gps_data_file, location)

# turn on gps led
GPIO.output(LED_gps, 1)

# encode data for database
_time = str(datetime.now())
email = "NA"
rnum = "NA"
src = "motion_sensor_#1"
data = {
    'location': location,
    'time': _time,
    'info': {
        'email': email,
        'room_number': rnum,
        'data_source': src
    }
}

# go to next state
NextState = 'Monitor'

```

Figure 21: Sample code for Initialization state

```

# constantly update the time
data['time'] = str(datetime.now())

# enter low power mode by turning off all LEDs
GPIO.output(LED_gps, 0)
GPIO.output(LED_motion, 0)
GPIO.output(LED_wifi, 0)

if ratDetected:

```

```
    # go to transmit state
    NextState = 'Transmit'
    # reset Flag
    ratDetected = False

else:
    # stay in monitor
    NextState = 'Monitor'
```

Figure 22: Sample code for Monitor state

```
# send data to database
dB.write_data(dB.PEST_TABLE, data)

# also write to local file
FM.append_data(pests_local_record, data)

# go back to monitoring
NextState = 'Monitor'
```

Figure 23: Sample code for Transmit state