# Team Report

## Introduction

**Project Title:** Digital Motion Detecting Alarm System

|  |  |
| --- | --- |
| **Names** | **E-mail** |
| Kaelum Beglin Sweeney | s00267367@atu.ie |
| Kevin Staunton | s00274293@atu.ie |
| Abdullah Ali | s00272245@atu.ie |

**Our GitHub page may be found here:**

[GitHub - KStaunton-ATU/IoT-Project-1: A light based alarm system concept with Pushing Box implementation.](https://github.com/KStaunton-ATU/IoT-Project-1)

**Our Trello board may be found here:**

<https://trello.com/invite/b/67b48fff32ee07260df654a4/ATTId9e707aaea3bd864438927856f95a33735B308F9/internet-of-things-project-1> (Invite Link)

<https://trello.com/b/Ce5Udhan/internet-of-things-project-1> (General Page)

## Outline The Problem

### The Growing Need For Security

The primary problem is the ever-existing need for a security system for home and business owners against trespassers and burglars. The Central Statistics Office has reported a 6% increase in burglaries in Ireland between Q2 2023 to Q1 2024 (McMahon, 2024). This is approximately 9800 burglaries during that time.

One in six homeowners have been victims of a break-in (Quote Devil, 2024). Along with the emotional distress of being a victim of burglary, there is a financial loss of money, jewellery and electronics. Should insurance mitigate this loss, there is still a financial loss incurred as the victim’s insurance costs are negatively affected. In addition, only 60% of homeowners have a burglar alarm system and only 37% have a CCTV camera installed. In contrast, 68% of homeowners do not own a safe and are not considering acquiring one. This highlights a need for informed awareness of the benefits of security systems, including a sense of safety and mitigating factor in the cost of home insurance premiums.

A security system can also address other problems. Such as the need for safety against fire and carbon monoxide. In the context of a business owner, there can be valuable equipment that needs to be monitored at all times to maintain factors such as temperature. A security system can provide a solution that can assist with automating maintenance of valuable equipment or product and prevent disruption of the business activities.

For safety in an industrial setting, it is important to design and implement an effective system. This is both cost effective and is a major step in reaching optimal performance in your business. Notably, failure to implement effective security systems can result in unnecessary downtime and in some cases, injure or kill employees. For instance, during the investigation of the Milford Haven refinery explosion (UK, 1997), the Health and Safety Executive (HSE) found that the major causes of the incident were generation of too many alarms, poor prioritization, poor control room display design and alarm flood (275 alarms in 11 min) prior to the explosion (Nochur et al., 2001).

Based on this information, we can conclude that there is a need for an effective security measure that deters criminal activity and gives home and business owners a sense of peace of mind that their property is monitored and kept safe from trespassers.

### References

* Eoin McMahon (2024) *Home burglary rates increase by 6% in the last 12 months.* Available at: <https://www.phonewatch.ie/blog/posts/posts-home-burglary-statistics/?form=MG0AV3&form=MG0AV3> (Accessed 22/02/2025)
* Quote Devil (2024) *Are Ireland’s Homes Secure Enough?* Available at: <https://www.quotedevil.ie/news/are-irelands-homes-secure-enough> (Accessed 22/02/2025)
* Nochur, A et al. (2001) *Alarm Performance Metrics* Available at: <https://doi.org/10.1016/S1474-6670(17)33592-9> (Accessed 22/02/2025)

## Summary of The Project Solution

The main solutions for all the problems are installing locks, alarm systems and security cameras. This is essential as it could save one from getting robbed, injured and assault. Saving people from living without concerns for their safety. Security cameras, alarm systems and locks save you from thinking about the safety of your house while you are gone out anywhere even abroad. This helps you be aware of your surroundings when you are not around your house or not available.

In our project the Arduino Yun helped us a lot. It was clear that an alarm system was the answer to the problem. The device’s job is to detect and notify the owner of the changes in the surroundings with a sense of urgency.

Since the Infrared Motion Detector wasn’t available, we switched out the motion sensor for a light sensor. The system’s job is to continuously monitor, and if there is a change, the alarm is triggered.

The two main functions of the alarm are:

* A buzzer and LED light turn on for 3 seconds to provide audio and visual signal, so the people nearby know there is an alarm and an alert, which could also inform a thief, to stop a burglary.
* The system logs the event online by using PushingBox. It records the time, device ID, and light sensor values in the spreadsheet.

The solution we identified to address this problem was a device that responded to one or more external stimuli. With our understanding of the Arduino Yun and its peripherals, it was evident that an alarm system was the appropriate answer to the problem. We define an alarm system as a device that constantly “listens” for changes in its surroundings. Once certain thresholds are met, the device acts to alert its owner of the changed surroundings with a sense of urgency.

The hardware setup involves using multiple attachments to the Arduino Yun. Initially, it was our intention to use an Infrared Motion Detector. However, it was unavailable to us and so we had to make a compromise. A light sensor was chosen as substitute. This decision was based on its simple use in triggering within a simple “if” statement in our software, much like the Infrared Motion Sensor. In addition, its function of detecting light still has applications in the context of security and therefore remains thematically appropriate. For example, the lack of light could indicate an unwelcome visitor in a brightly lit area. Alternatively, an excess of light suggests that a door has opened to a dark area. For our project, we use the sensor to listen for a drop below a predefined baseline. When the light value falls below this threshold, the alarm is considered triggered, leading to multiple events.

A buzzer and LED peripheral are used as a local alert, demonstrating both audio and visual indicators of an alert situation. For convenience, these peripherals are turned off after 3 seconds, as a continuous buzzer noise can be irritating since we currently use it for demonstration. These peripherals ensure that people in the surrounding area know that there is an alert situation and also deter would-be burglars from their attempt at breaking into a building.

One major consideration we had when thinking about the software was the use of networking-based services. We quickly determined that while creating a sense of urgency and alertness was our top priority, there was practical use in trying to log when the alarm triggers. From this, we decided to implement two things. The first task was to deploy a Google App script and a corresponding spreadsheet. When the alarm is triggered, the light sensors value is inserted into a string object representing the PushingBox URL. By using the HTTP and Bridge classes, this string is used to make a HTTP request to PushingBox, which then makes use of the Google App script to insert the current timestamp into the spreadsheet along with the Device ID and the light sensors recorded value.

Continuing with the theme of alertness and urgency, we also set up a PushingBox service that sends an email to our student email address. Once again, using the HTTP and Bridge classes, the URL is used as a string to make the HTTP request that will send an email notifying that the alarm has triggered, and the spreadsheet has been updated. We observed that the email was received instantly, therefore demonstrating the effectiveness of the alarm system and its potential application at notifying through other means such as text messages or app notifications.

When considering the types of sensors available with the Grove Shield, we considered the context of an alert system and how it could be used in other situations other than security. Going forward, we have the potential to consider elements of smoke detectors, fire alarms, temperature and humidity sensors to make software that is multi-purposed for different needs in alert systems. We can also repurpose online services such as PushingBox for potential ideas regarding this in a similar manner.

## Project Requirements

1. The alarm must reliably trigger and be trustworthy, not falsely triggering.
2. The alarm must be loud enough to alert someone.
3. The device cannot be too big lest it be easily detectable and avoidable.
4. Minimise the parts required so that it is more easily accessible for anyone to use and not wasteful in line with SDG 12.
5. The device should log when it triggers to make sure the user knows if it triggered while they were away.
6. It should be disarmable without having to remove its own power as that is sometimes impractical.

## Initial Design

#### Proposed Device

The proposed device is a security alarm with enhanced quality of life features that made it easy to use as well as providing useful data about its operations. It will be a small device that detects motion using one of the sensors in the Grove kit. It will be able to be placed anywhere. It should be able to be turned on and off at will and we plan to have it use APIs to send notifications to the user.

#### Proposed Code Design

The proposed code design will be fairly simple. It will build off the labs using void setup to initialize the sensors. Most likely an LED, uh, light sensor or motion sensor. We will use the void loop to continuously check for the right conditions to indicate that someone or something has set off the alarm, using if statements to determine if the situation calls for it. We will also have code in the Arduino file connected to our APIs which we are not fully sure on yet.

#### Proposed Hardware Setup

A diagram of a buzzer

AI-generated content may be incorrect.

We believe all we will need for the hardware portion of the project will be the Arduino as the base of the project offering the programmability aspect allowing us to give function to the sensors we will need to utilise.

Of course, to actually build the alarm with various sensors we will need to attach the grove kit to the Arduino to enhance its functionality.

The planned attachments we will be using are a buzzer, to alert people in the area of the alarm that something is wrong, a motion sensor, to detect the movement in the area which the alarm system is meant to be protecting, an LED which will light up when the alarm is triggered just in case the buzzer doesn’t work then at least there will be something trying to alert someone.

#### Planned APIs

We plan on using two APIs in this project; an API using PushingBox to write a record of every date and time that the alarm device has gone off and an API using PushingBox to alert the user through email of the device triggering so you can always be aware when it triggers.

## Data and APIs used

We are using the Arduino to collect data relating to the alarm. It will send the data of when it was set off to a google spreadsheet using an API very similar to what we learned in the lab exercises in class in the last few weeks, this is to ensure the user is always aware if the alarm is triggered when they were not around. It uses PushingBox to direct the date and time towards our google spreadsheet and creates a new entry on a new line.

We also are using a PushingBox API which sends an email to notify the user whenever the alarm is set off, it can be sent to any email easily by just changing a setting on PushingBox. We are not accessing any external datasets at the moment for this project but plan to when we get the chance to improve on it.

## Implementation of Plan

A close-up of a computer chip

AI-generated content may be incorrect.

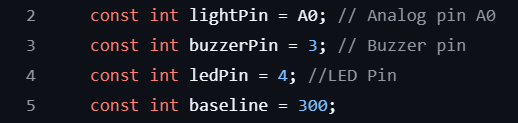
* Equipment needed:
* Light sensor:
* LED:
* Buzzer:
* API to be used:
* PushingBox:
* Google Apps Script:
* Code samples:
* This is a screenshot of the code to be used to make the alarm system to wait for 3 seconds and turn off the alarm.

Parts Required:

* Arduino Yun mini-PC
* Grove Shield
* Buzzer peripheral
* LED peripheral
* Light sensor peripheral
* Arduino IDE
* Bridge and HttpClient libraries

With the Yún set up, the integration with the PushingBox service comes next. A unique device ID is generated within PushingBox, and the Arduino code is updated to include this ID and the endpoint URL. The HTTP request functionality is tested separately using static values to ensure that data is successfully transmitted to PushingBox. Logs and responses from the PushingBox platform are carefully examined for errors and inconsistencies.

To implement this project, the Arduino Yun is needed. For convenience, a Grove Shield will be used to avoid the need for a breadboard and knowledge of electrical pins. A light sensor is connected to an analogue port on the Grove Shield while a buzzer and LED peripheral will be inserted into digital ports, as their values are only on and off. We will test these parts individually to ensure they work correctly and then use them together.

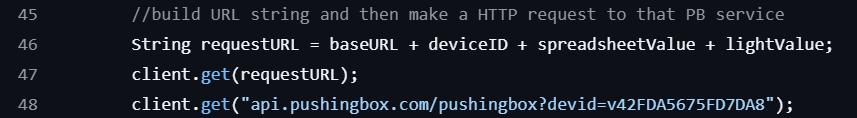


When writing the software, we will be using the Arduino IDE to write and upload our C++ code to the Arduino Yun. The Bridge and HttpClient libraries are required to be installed on the IDE before uploading so that our project will have networking capabilities. Thorough testing will be required with each addition to the codebase.



Regarding networking functions, we will set up a PushingBox service to handle HTTP requests sent from the Arduino Yun. A Google spreadsheet will be created that will have App script deployed. Said script will parse arguments sent in a HTTP call and insert the appropriate values into the next available row of the spreadsheet. The deployed web app URL will be used for setting up the PushingBox service. The PushingBox service will handle the HTTP request sent from the Arduino Yun and send the values to the spreadsheets deployed Web App URL. This is easily the most challenging feature of the project and so in-depth testing is required.

As a secondary networking goal, a call to the PushingBox API will send an email to our student email. Once again, we will rely on our HTTP request from the Arduino Yun to PushingBox to send this email.



With each of these steps realised and adequately tested, the final test will involve using all parts and services together. Successful testing will demonstrate an alarm system that triggers when the amount of light detected falls below a defined threshold. When triggered, info is sent to the Google spreadsheet and an email is sent to an address.

## Testing Approach

Our approach towards testing involved breaking down our objective into the smallest goals possible. The first phase of testing focused on our basic understanding of the Arduino and how it’s peripherals functioned (Unit testing). With this focus, we ran separate tests to trigger the light sensor, buzzer and LED separately and then finally tested with all 3 peripherals connected at the same time. Expanding upon this, we introduced the threshold for the light sensor within an ’if’ statement and made use of the delay function. From this, we were able to make a very basic and easy to understand function.

The second phase of testing targeted our understanding of the PushingBox service and object-oriented development. This time we included the HTTP library and used an object based on this library to make HTTP request to an existing PushingBox service. Rather than observe each minute addition to the code, we decided to implement the changes as best we could and observe what worked and did not work. Based on our results, we iterated and improved upon our code, resulting in our current build.

**Phase one testing:**

|  |  |  |
| --- | --- | --- |
| **Peripheral** | **Expected Result** | **Outcome** |
| Light sensor – Demonstrating its use. | Console should show different light values depending on light shining on sensor. | Sensor successfully gets increased values when lights shone directly on it. Only works in analogue port. |
| Buzzer – Demonstrating its basic function. | Buzzer should create a loud noise when set to HIGH | Buzzer is loud and annoying. Need to turn it off after a certain amount of time passes. |
| LED – Demonstrating its use. | LED should light up immediately. | The LED lights up immediately without issue. |
| All 3 peripherals | LED and Buzzer should trigger for 3 seconds when the observed Light value is below our determined threshold (500) | The ’if’ statement executes successfully under correct conditions. Buzzer and LED trigger for 3 seconds as intended. |

**Phase two testing:**

|  |  |  |
| --- | --- | --- |
| **Test** | **Expected Result** | **Outcome** |
| HTTP string – Before making a request, the string should be accurate. | The printed string should be: api.pushingbox.com/PushingBox  ?devID=v0CC38418000E8B6&LightValue=[XXX] | The outputted string is accurate. |
| Spreadsheet – Make sure the PushingBox service works | When the HTTP request is made, the deviceID and light sensor value should be inserted into a row with the current timestamp. | The spreadsheet is successfully updated with expected values |
| Email – Make sure the PushingBox service works | When the HTTP request is made, an email should be sent to s00267367@atu.ie | An email is successfully sent to the address. |

## Security Measures

Overall, the Arduino does not have many options to secure it when transmitting data through Wi-Fi so we must do our best to make the most out of the basic security measures that would apply to anything that you would normally want safe and secure.

In an actual practical application of this alarm, it would usually be hidden in the first place as to avoid someone seeing it and evading it altogether. Furthermore, we would have it encased in a protective box to avoid destruction of the alarm system or interference with its functionality like unplugging connections or in the worst-case, reprogramming it and accessing the Wi-Fi network it is connected to.

On topic of Wi-Fi security, we would of course give it a very random and long password, replacing doghunter with something safer including special characters and the like. This would avoid disruptions with changing the devices Wi-Fi connection. Similarly, the Wi-Fi password itself would be made just as secure.

## Future Improvements

The alarm system in its current state makes use of three sensors to listen to its surroundings. Our first thought on future improvements is the wide variety of sensors we can potentially add to suit various scenarios. For example, sensors for detecting smoke and temperature would suit a fire alarm. We have also considered the use of an LCD screen as a way of informing a person which sensor has triggered the alarm. By doing so, the device becomes more informative and reduces our reliance on the PushingBox service to send an email which is useful should the Wi-Fi connection become unavailable at any point. The LCD screen could also present real-time information.

A particularly interesting peripheral that interests us is the camera module. Similar to a CCTV camera, this would give us an opportunity to take pictures or video and explore integrating an SD card reader into the design. In addition, we could potentially backup the contents to cloud storage using a networking-based API.

From a networking perspective, it could be useful to record sensor date over a prolonged period of time and store this information for statistical analysis. Doing so could provide a unique insight into how well our device performs. If we are able to observe patterns in the data, we could account for these changes in software to make a more interesting use case. For example, instead of an alert scenario that would demand immediate attention, we could use the data to define a “caution” scenario that would not necessarily need attention but would be more informative to the user.

Expanding on this theme of being informative, the use of Cloud based services could allow us to remotely monitor sensors in real time. This feature would allow a user to view sensor values at any time from any location. This is also an excellent opportunity to include the option to configure alarm triggers, such as increasing/decreasing the light sensitivity threshold to suit a user’s preferences and needs.

One obvious design flaw we noted is the fact that the Arduino Yun is powered by its USB connection to a PC. With this in mind, our main focus will be to research a solution to attach a battery so it may operate in practically any location. Expanding upon this and the limitations of the battery, we have also considered the need for a backup battery that could be used should the primary battery fail. For the sake of sustainability, we will research and consider the viability of a solar panel to maintain the charge of the battery and prolong its overall use. Since the use of a battery is another potential point of failure, we could attempt to monitor its charge using software and send out an email or text message should the battery either fail or be running low on electricity.