FINAL PROJECT REPORT

ANALYSIS AND DESIGN

Car damage monitoring system

National College of Ireland

BSc in Computing

2016/2017

Graham Kinsella

X13558787

Graham_kinsella3@hotmail.com

National College of Ireland

Table of Contents

Executive Summary

The purpose of this document is to set out the requirements for the development of a car damage monitoring system.

The intended customers are owners of motor vehicles and car manufacturers.

Cars have many various sensors that detect impacts to trigger airbags, seatbelts, detect engine knocks for fuel mixture but they do not contain a dedicated monitoring system for impacts.

This application will allow a user to monitor impacts to their vehicles and alert them if they may have potentially caused damage. When a vehicle is contacted by another, the sudden shift in acceleration is monitored by an accelerometer and if above a set threshold, warn a user that severe damage could have occurred.

The second usage is in relation to "Dent and run" cases. In a survey conducted by Motors.co.uk, One in five UK drivers admit that they have damaged another car and left the scene without making the owner aware of the incident. If this happens without the presence of onsite CCTV or a witness it can be near impossible to trace the perpetrator.

This system will combat that through the capturing of images upon impact and the capturing of geolocation data. This means if a car is hit and the perpetrator leaves the scene, a record of the impact will be captured and recorded to aid the vehicle owner in tracing the parties responsible. If the owner of the car does not notice the damage at first and makes other stops, they can use the location data to determine where the event occurred.

This could be expanded on by using the data collected from various impacts from everyday road use (potholes, paths, objects on road) to determine overall wear to the cars components such as suspension.

1 Introduction

1.1 Background

Many road users are involved in minor collisions daily. Most of these may not seem to have caused much damage as motorists cannot see the immediate damage to a car without being able to access areas often hidden behind panelling or body trim such as bumpers.

Unbeknownst to most people, bumpers and other car parts are plastic and reform to their original shape after an impact even though damage to body components behind them may have occurred. This leads to many motorists leaving the scene of an accident thinking no damage bar superficial surface scratches have occurred only to find out later that their car is badly damaged/wrote off.

Another reason for the development of this application is in regards to another common issue faced by motorists. A car could be parked at one place for several hours and hit by another vehicle. When the owner returns, they may not immediately notice and travel to many other destinations before finally realising their car has been damaged at some point in the journey.

With this application, the location where the damage occurred and images from cameras mounted on key points in the car to allow for a 360-degree view will capture details of the user can then view it and return to the location and possibly access witnesses or CCTV footage.

1.2 Aims

The intent of this project is to build an application that takes input from a vibration sensor and monitors for impacts to the vehicle. The system will record vibration strength and execute different functionality depending on the severity of the impact detected. In the case of a severe impact, images from the event and geolocation data from location the impact occurred will be captured and an alert will be sent to the user that some serious damage may have occurred. The recorded impacts can then be viewed by the user with any device that connects to the internet by navigating the URL of a hosted site

The broken-down objectives can be quantified as:

- Software must be written to evaluate the variance in impacts
- In the event of an impact, the strength should be assessed by the software to determine how severe it is
- If the impact is classed as severe, Capture images from camera and capture geolocation data and store this in a database
- Send an alert to user if severe impact has occurred using on-board LEDs
- Allow a user to view impact history through a web app

2 Architecture Design

2.1 Hardware Architecture

As this system is unique, a fair amount of research was required to determine the best hardware to use for the purpose outlined. A **Raspberry Pi** was chosen as the microcontroller for many reasons. It is small enough to fit in any vehicle, even a motorcycle, without causing a hindrance to the owner. It is widely supported and has many different add on boards that would increase the projects chance of success and further development by broadening the range of sensors that can be used.

Many sensors were researched such as Impact, vibration and Knock sensors but they proved to be unsuitable for the project.

Impact requires that the sensor be impacted for current to pass through. This is not appropriate as the sensor will be used to detect impacts to the environment the device is in not physical impacts to the sensor itself.

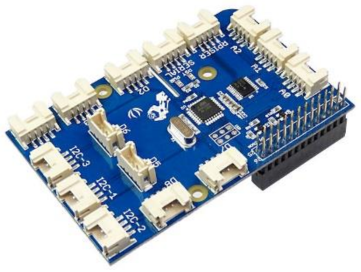
A knock sensor detects if a vibration occurs but has no ability to determine strength of vibration so this was ruled out.

The next sensor that was researched was a vibration sensor. This one seemed to tick all the right boxes and was used for some of the development time but ended up proving to not be right for this project. The vibration sensor works by detecting vibrations to flexible MEAS film attached to the sensor by wire. This film needs to be flexed for the sensor to detect vibration and as above the sensor needs to monitor for impacts in its environment not directly interacting with the sensor itself so this was ruled out

The Sensor that has been chosen is a **Grove Pi 16g 3D accelerometer** created by Seedstudio. When a vehicle is impacted the sudden change in acceleration that will occur can be determined by measuring the state of the device along 3 axes (X, Y, Z) and if above a set threshold then the system can be activated. Car impacts are generally under 9g so this sensor should be able to handle any impact which the application is designed to monitor for.

To capture geolocation data a **Grove GPS Sensor** will be used this will be more useful in a mobile environment as it will remove the need for an internet connection to determine current location.

In conjunction with the Grove Pi sensors an add on board for the Raspberry Pi is required to allow access to libraries to run the sensor. The **GrovePi+** provides access to multiple libraries for various sensors and should allow for sensor integration more easily than using the General Input Output (GPIO) Pins on the Raspberry Pi



Images will be captured via a **Raspberry Pi Camera v2.** This camera is developed by Sony and provides high quality 8mp images. It is mounted to the Raspberry Pi board itself and has its own dedicated socket.

An **LED** is activated when system detects an impact to tell the user that severe damage has occurred this LED will stay on until the user presses the reset button attached to the device. This is in line with typical vehicle standards. Generally, if a fault occurs in a system within a motor vehicle an LED is lit up to make the owner aware of a fault. This LED could be replaced with one included in the vehicle dash and can be turned off via an OBD (On-board Diagnostics) reader after inspection by a qualified professional like other vehicle faults.

The system is reset by using a **Grove button**. This shuts off the LED. This logic can be replaced with the use of an OBD reader to turn off the LED.

List of Hardware outlined:

Raspberry Pi 3 Model B

Grove Pi+

Raspberry Pi Camera v2

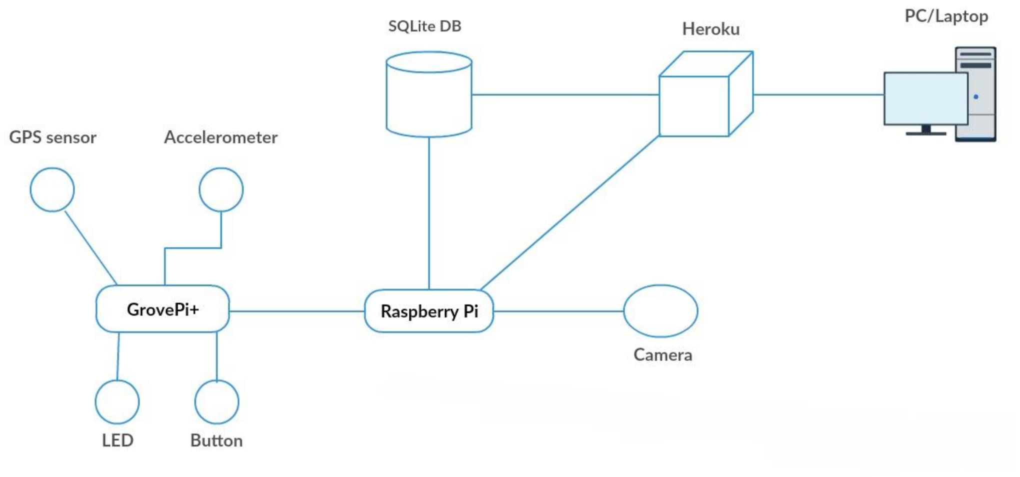
Grove 16g 3 axis accelerometer

Grove GPS Sensor

Grove LED

Grove Button

2.2 System architecture diagram



A GrovePi+ Board which will handle sensor input and relay the inputs to Raspberry Pi. A python program on the Raspberry Pi will receive data from the GrovePi+ and images from the camera unit. The program will write information to a Database which is uploaded to a cloud service, Heroku, and can be viewed by the user via a web app.

2.3 Software Architecture

The software components of this project are written in **Python**

The raspberry Pi runs a modified Debian based linux system called Raspian. The software components for the monitoring system have been written in Python and the view presented to the user is written in **HTML**. Python is one of the most common languages used with Raspberry Pi development, Python was originally designed as a teaching language so allows for complex scripts to be written without the added hindrance to development of learning a complicated language.

Python is well supported in relation to the Raspberry Pi and will open access to a lot of support communities. This will be advantageous as this project is not something that has been done many times before.

Most of the libraries included with the Grove Pi also use Python in examples so it made sense to use this language. The GrovePi+ however does support C, C#, Java, PHP and Python.

SQLite was used in implementing the database. This was chosen as it is a server-less implementation of SQL meaning the database itself is relatively small. SQLite also requires no administration like would be needed with Client/Server Databases (postgres, SQL Server).

This excerpt from the SQLite website www.sqlite.org best describes it "*Because an SQLite database requires no administration, it works well in devices that must operate without expert human support. SQLite is a good fit for use in cellphones, set-top boxes, televisions, game consoles, cameras, watches, kitchen appliances, thermostats, automobiles, machine tools, airplanes, remote sensors, drones, medical devices, and robots: the "internet of things".*"

There are 6 classes within the application that provide the main functionality

Impact.py

This class is responsible for the main functionality of the whole application and contains the impact monitoring code

When the class is ran the `upload()` function is ran which initially checks if the microcontroller is connected to the internet by pinging www.google.ie. If it gets a successful response it uploads the database to Heroku (Cloud hosting service) and then sleeps the function for 30 minutes before checking again. If it is not successful in connecting to the internet is checks every 15 minutes until it is. This keeps the web app up to date with the latest information.

Within this class exists the function `def crash()` When the application is started this function is ran in a loop, it continually monitors GPS data by calling the GPS() function and the input from the Accelerometer `adxl345.getAxes()`. If the Accelerometer reading is above a set threshold the system is activated.

An image is captures and stored to an image called out.jpg this is then opened and using Base64 encoding it is encoded into a string. The GPS data is taken from the sensor and stored as a string.

These two (GPS, Image) the sensor data that kicked it all off and the current data are all added to a list and committed to the DB as a record. The system then continues to monitor for impacts

The `warn` () function is then activated which turns on an LED until the user clicks the button turning it off.

GPS.py

This class is responsible for reading GPS data from the Grove GPS sensor. This code came from SeedStudios and was modified for purpose. It allows all the GPS data taken from the GPS sensor to be split into individual values. These values are then used in Impact.py to tell the user where the impact occurred. The returned values can be taken from the web app and input to google earth to show the location the impact occurred.

LED.py

This class is responsible for the logic that turns on the LED when an impact occurs and turns off the LED when the button is pressed. This code came from SeedStudios and was modified for purpose. When the LED has been activated by an impact it will stay on until the lightoff() function located within this class is called with a button push.

Upload.py

This class contains the method `def upload()` which is called in Impact.py to keep the web app updated with the most recent impacts. On execution, a http request is sent to www.google.com, if it receives a response then the latest database is pushed to Heroku by executing a shell script Upload.sh, which is also located within the solution. After execution, the script waits for 30 minutes before trying to upload again.

If the request to google is unsuccessful then the system goes into a wait state for 15 minutes before trying again.

Requirements.txt is associated with this upload process. It contains the packages that Heroku needs to install to run the app successfully. It is a list of the needed imports. These are installed during the migration process on the cloud environment.

Web.py

This class is responsible for displaying the records held within the sqlite db within a browser. This is achieved by using Flask, flask is a microframework for python that allows web apps to be scripted in python. Within this class, each record within the database is assigned to a list this is then passed to a html file records.html where a loop is executed that displays each of the records within the DB. In this html file the Base64 encoded string that was captured in Impact.py when the impact occurred is serialized and displayed back to the user.

Search functionality works by taking a date input from the user and checking if a record containing these characters exists in the DB. Delete takes the ID of the record on which the delete button is pressed and executes a query on the DB to find it and then removes it from the records.

3 entry points/methods exist in this Web.py,

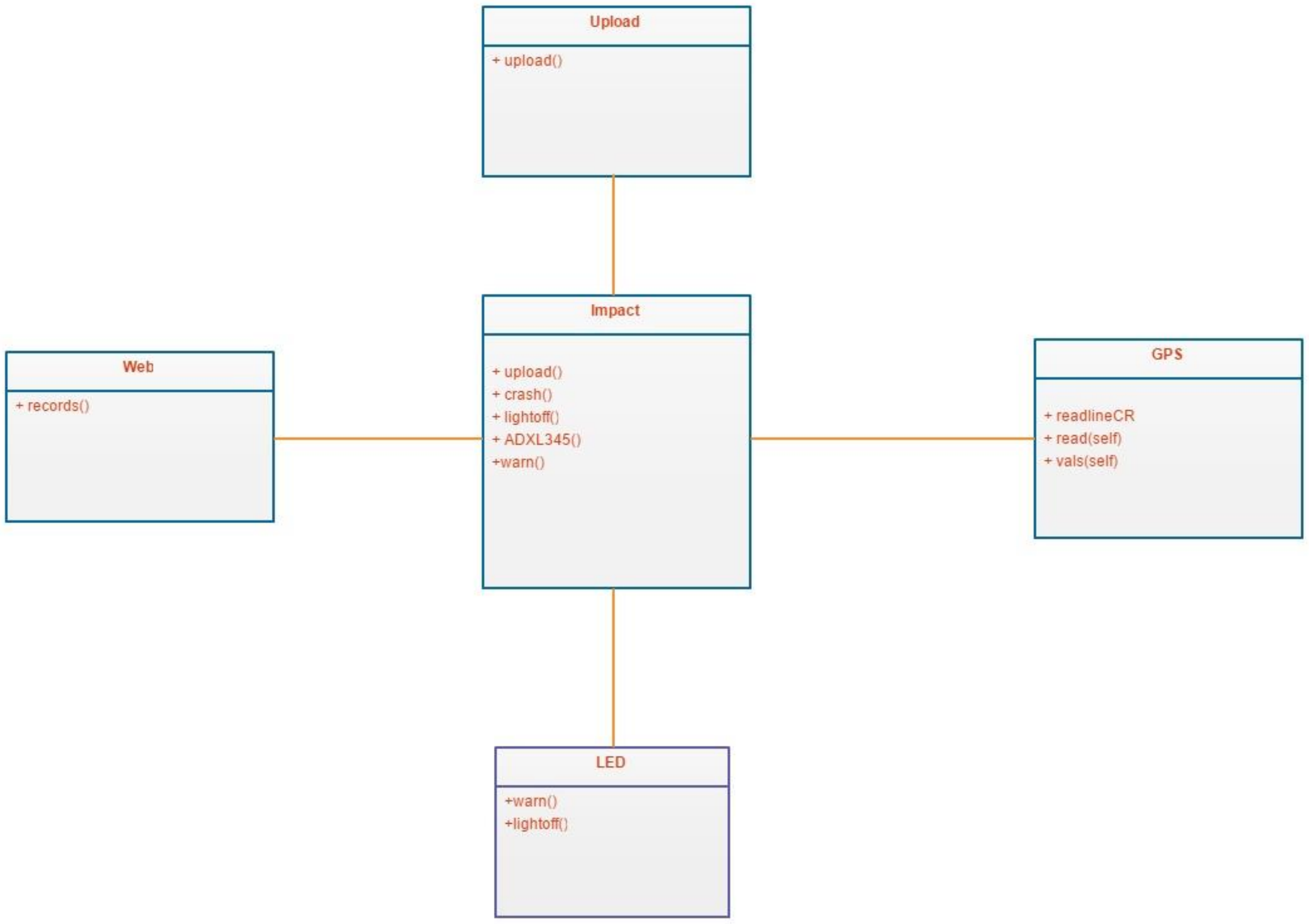
/ser which is associated with the search functionality displayed on screen

/delete which is associated with the delete functionality displayed for each record.

/record which displays all records in DB

Procfile.py tells Web.py which port to run on when ran on Heroku. It takes the port that Heroku has chosen and uses that to run the web app.

Class diagram



3 System Design

Requirements:

Requirements will outline the functional and non-functional requirements necessary for this project.

Testing:

This section outlines test cases that will be ran on the system to ensure it works as it should

Conclusions:

This section sums up key points and gives the advantages, disadvantages, limitations and opportunities of the project

Research:

The research section contains research into existing systems similar to this project and explains the differences between these and the system outlined in this document.

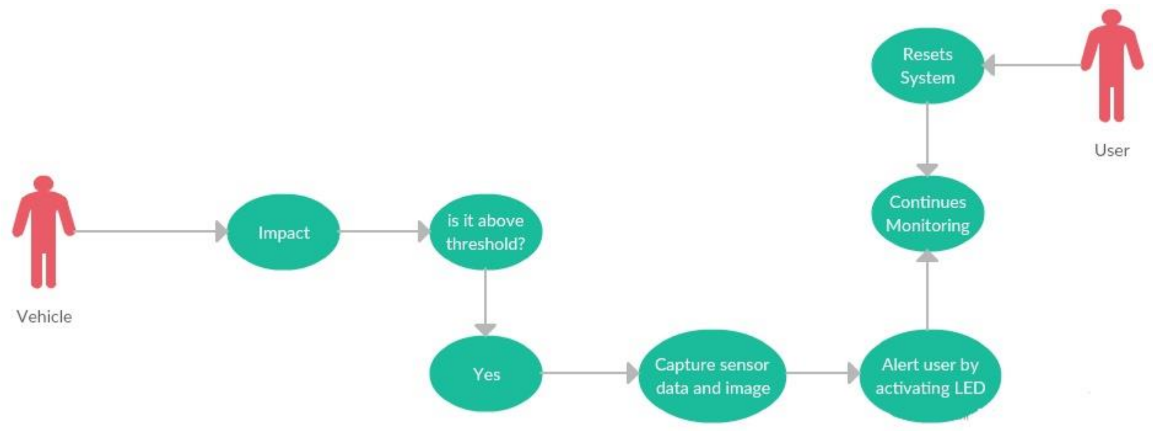
3.1.1 Requirement 1 <Alert user>

3.1.1.1 Description & Priority

If sensor detects impact over Severe threshold, system is activated and user is alerted that Damage could have occurred.

Priority 1

3.1.1.2 Use Case



Scope

Impact occurs to vehicle, sensor detects Impact above Severe threshold, System goes through End to End flow and alerts user.

Description

Use case describes the activation of system and stepping through end to end

User is alerted that severe damage may have occurred by LED being activated. Stays on until the user presses the reset button.

Flow Description

Precondition

Sensor is actively waiting for impact to occur

Activation

Use case activated when impact occurs from external force e.g another vehicle

Main flow

1. The system detects impact
2. The python script determines if impact is strong enough for activation
3. The system gathers geolocation data
4. System takes image from camera input
5. System alerts user

Termination

LED stays on until user resets

Post condition

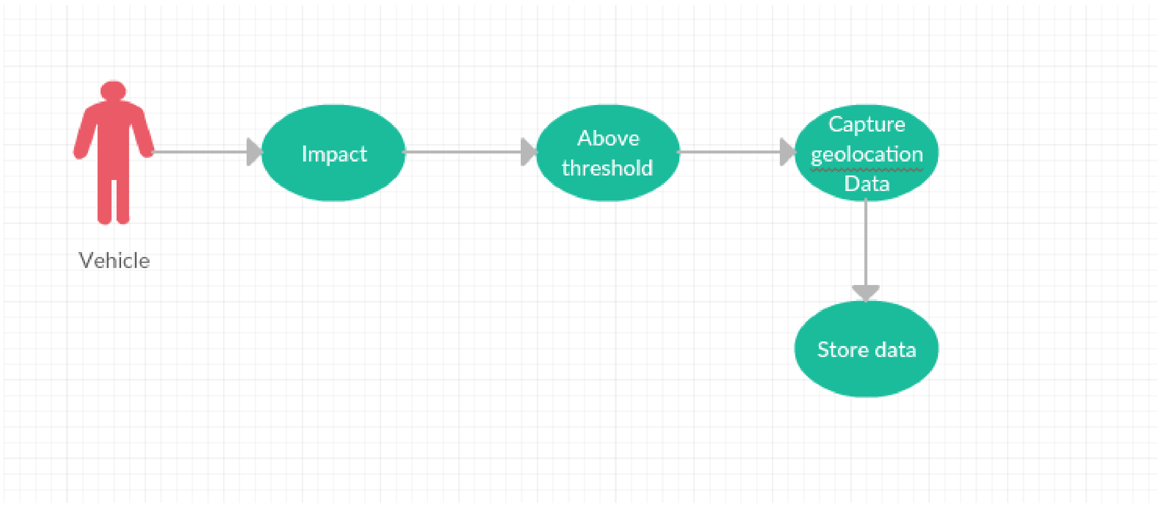
System resumes monitoring for impacts

3.1.2 Requirement 2 <Capture User's Geolocation Data >

3.1.2.1 Description & Priority

After activation, users Geolocation data should be captured

3.1.2.2 Use Case



Scope

Geolocation data capture after activation of system

Description

This use case describes the capture of Location data after triggering of system

Flow Description

Precondition

System detected impact above set threshold

Activation

Gathers user's geolocation data

Main flow

1. Sensor activation
2. Systems determines sensor is above threshold
3. System captures users Geolocation data

Termination

Data captured and stored

Post condition

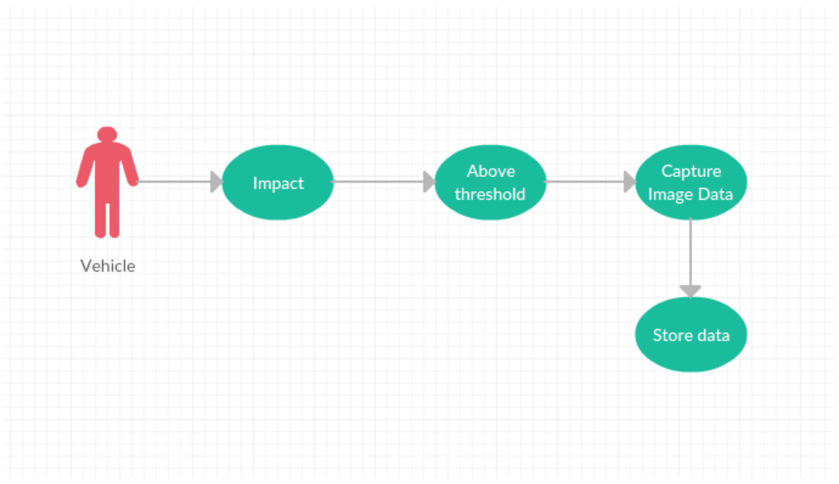
System continues with routine

3.1.3 Requirement 3 <Capture Image>

3.1.3.1 Description & Priority

Images captured in event of system activation

3.1.3.2 Use Case



Scope

Image captured from camera input, encoded using base64 and stored in database

Description

This use case describes the capture of Image data after triggering of system

Flow Description

Precondition

System detected impact above set threshold

Activation

Image captured from camera input

Main flow

1. Sensor activation
2. Systems determines impact is above threshold
3. System captures image
4. System encodes image

Termination

System stores image

Post condition

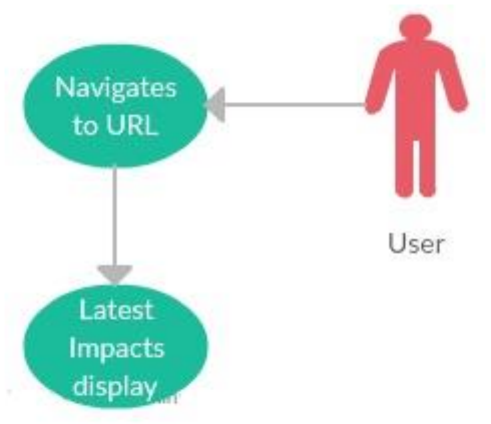
System continues with routine

3.1.4 Requirement 4 <Retrieving and Viewing Stored Data>

3.1.4.1 Description & Priority

User can view stored data using a web app

3.1.4.2 Use Case



Scope

Allow user to view detected impacts

Description

This use case describes how retrieval and viewing of stored data will work

Flow Description

Precondition

User has connection to internet/a device to access the internet (Phone, PC, Laptop)

Activation

User navigates web app

Main flow

1. User navigates to web app URL
2. UI displays users impact history

Termination

User moves away from website

Post condition

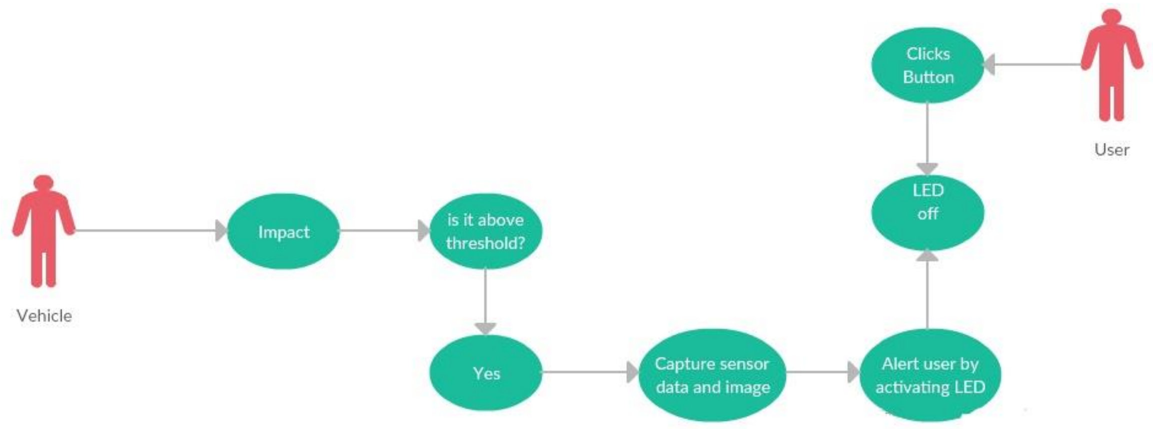
System awaits connection

3.1.5 Requirement 5 <Reset warning>

3.1.5.1 Description & Priority

User can reset warning system by pressing button connected to device

3.1.5.2 Use Case



Scope

Allow user to reset system

Description

The use case describes the user resetting the system

Flow Description

Precondition

severe impact has occurred

Activation

LED is on

Main flow

1. LED is on
2. User clicks button
3. LED off

Termination

System turns off LED

Post condition

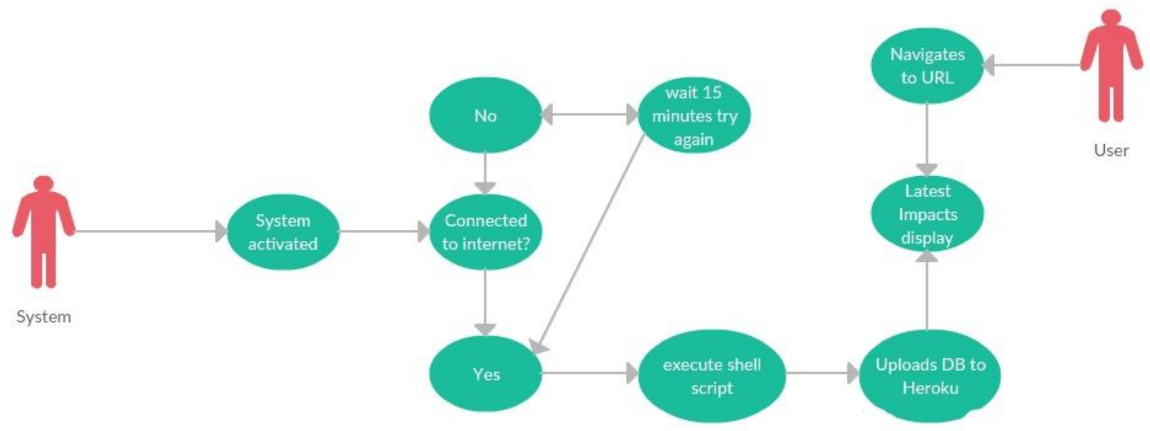
System continues monitoring

3.1.6 Requirement 5 <Database Upload>

3.1.6.1 Description & Priority

Database is upload to Heroku when device is connected to internet

3.1.6.2 Use Case



Scope

Keeps web app updated with impacts

Description

The use case describes the system uploading the database to cloud hosting site

Flow Description

Precondition

severe impact has occurred, device connected to internet

Activation

System activated

Main flow

1. System is activated
2. Script checks if connected to internet
3. Executes shell script
4. Uploads database

Termination

System monitors for impacts

Post condition

System waits 30 minutes before trying to upload again

3.2 Functional requirements

System must alert user if impact occurs

System must capture user's geolocation data

System must capture image via camera

System must allow user to view stored data from device

System must allow user to reset warning system

System must upload database to Cloud hosting site

System must store impacts in database

3.3 Non-Functional requirements

3.3.1 Reliability requirements

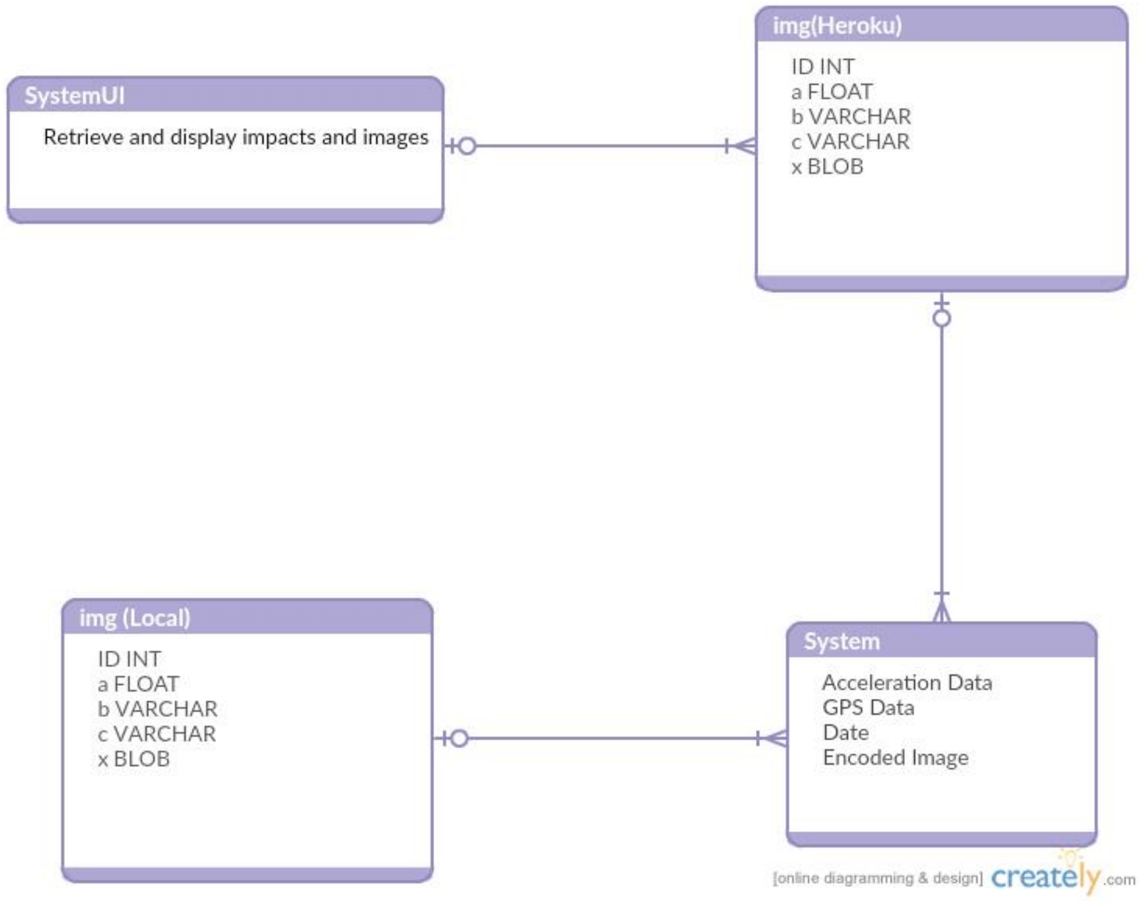
System should be reliable enough to handle multiple impacts without failing.

Multiple impacts could occur in succession and each one must be captured individually

3.3.2 Reusability requirements

System should continue to monitor for more impacts even if a severe one has occurred and LED has been activated. Without this data loss could occur

3.3.3 Data requirements



A SQLite database consisting of one table, image, is created when an initial impact occurs. This table holds all the information of an impact, Acceleration data from the accelerometer, GPS Data, Date of incident and a base64 encoded image. When captured this is stored in a database on the local machine.

When connected to the internet a shell script pushes this database to the cloud hosting service Heroku. This Database is then read by the system UI and displays each records detail to the user via a web app.

3.3.4 User requirements

User requirements can be quantified as:

- System display alert if impact threshold is above a certain value
- System will capture users Geolocation data, Acceleration data, date and images in the event of sensor activation being above defined threshold

- User will have ability to reset warning system
- User can view stored events
- Values will be stored in a database
- Values can be read through a UI

3.3.5 Hardware requirements

Microcontroller – Raspberry Pi 3.0

Sensor Control – GrovePi+

Sensor – Grove 16g 3 axis accelerometer

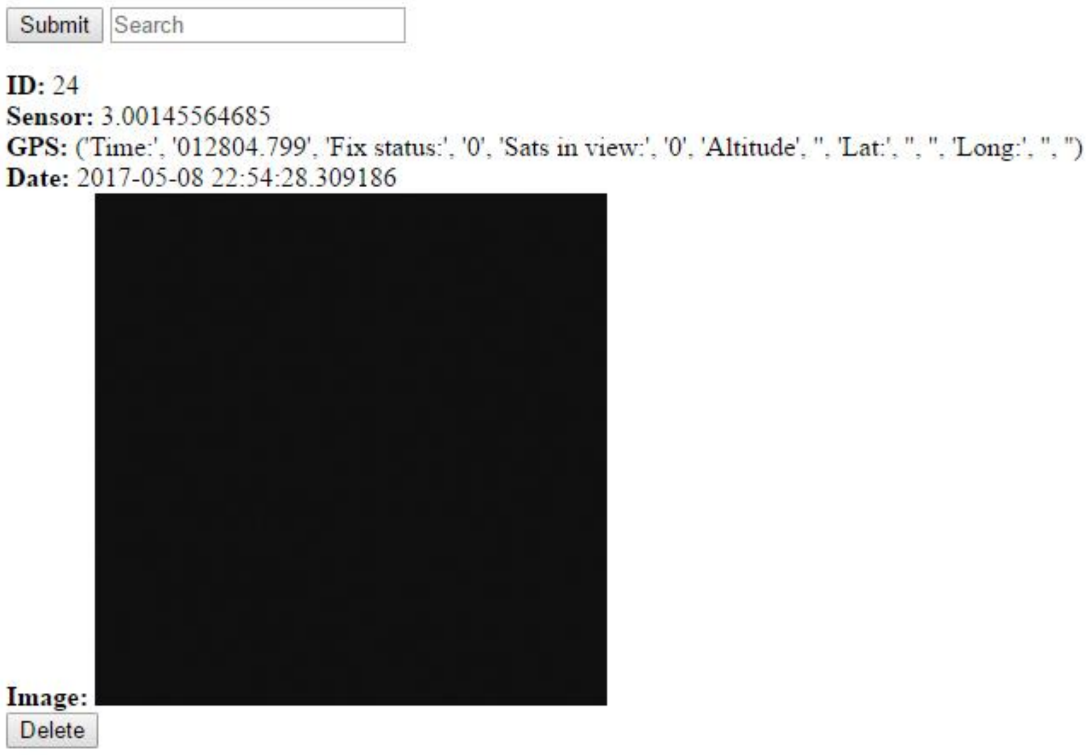
GPS – Grove GPS sensor

Warning system – Grove button and LED

Camera – Raspberry Pi Camera V2

Laptop/Desktop Computer – Interact with Raspberry Pi during development

3.4 Graphical User Interface (GUI) Layout



ID: 24
Sensor: 3.00145564685
GPS: ('Time:', '012804.799', 'Fix status:', '0', 'Sats in view:', '0', 'Altitude', '', 'Lat:', '', '', 'Long:', '', '')
Date: 2017-05-08 22:54:28.309186
Image:

A secondary web app was developed alongside the python application to enable the user to view the impacts from anywhere via a mobile device or desktop client. This was decided to be a better approach than having to have the user connect a display to the device.

As this decision was taken later in development once the main applications functionality had been finalized and testing had been complete, the UI is a bare bones HTML document at the time of writing. Development is still ongoing so this will look different in Version 1 of the product.

3.5 Testing

The Test plan for this project includes the following test cases which were executed upon completion of development and the calling of a code freeze on the main development branch. Sanity testing of the final build was also conducted by manual testers.

ID	Title	Steps	Expected result
1	End to End test	1. Activate sensor with reading above threshold 2. Navigate to view impacts page 3. View record created at step 1	**LED should turn on** **Record should exist** **Should contain location data, image, date and Acceleration Data**
2	Image capture test	1. Point camera at identifiable object 2. Activate sensor with reading above severe threshold 3. Navigate to impacts page 4. View record created in step 2	**LED should Turn on** **Record should exist** **Image in record should match object from step 1**
3	Warning system test	1. Activate sensor with reading above severe threshold 2. Click Button	**LED should turn on** **LED should turn off**
3	Location capture test	1. Identify Location 2. Activate sensor with reading above severe threshold 3. Navigate to impacts page 4. View record created in step 2 5. Copy location data into Google Earth	**Record should exist** **Location should match location identified at step 1**
5	Concurrency Test	1. Activate sensor with reading above severe threshold more than once in succession	**LED should turn on**

		2. Navigate to impacts page	**Record should exist for each activation**
6	Database Upload test	1. Navigate to impacts page in web app 2. Make note of impacts that exist 3. Navigate to device 4. Activate system 5. Trigger system 6. Navigate back to web app	**New Records Exist**

To avoid the need to purposefully damage a motor vehicle for testing, the above tests will be performed in a lab environment and activation will be simulated.

Concurrency testing will be vital for this project to function properly, if impacts occur in quick succession the system should record details of both impacts.

Testing had originally been planned to be achieved using unit tests but I was unfamiliar with Python at the time and a better approach is to use print statements to ensure that the code is operating as it should. This is a better approach due to the nature of Python

Thus, during development code was tested using print statements to ensure that everything was behaving in expected fashion and that variables had the appropriate data. This is a common testing procedure when developing in python and was adopted to ensure good quality of code.

3.6 Test Results

5/6 test cases passed on first run. The failing test case related to database upload and a bug was raised for this defect. This was fixed through a patch and a new build was given for testing. Upon 2nd run 6/6 tests passed and the build was deemed to be stable. Test cases will continue to be executed daily up until final release on the 24th of may

As mentioned above the UI is still in development at time of writing, The UI for this application needs to be as simple as possible due to the nature of it. The device could be used by a vehicle owner who is unfamiliar with computers in general. This needs to be considered when designing the UI. The following usability testing methodologies will be considered during this time to create a UI that is acceptable to users.

1.Effectiveness

Effectiveness is about whether users can complete their goals with a high degree of accuracy

2.Efficiency

Efficiency is all about speed. How fast can the user get the job done?

3.Engaging

Engagement occurs when the user finds the product pleasant and gratifying to use

4.Error Tolerance

minimize errors from occurring

5.Easy to Learn

Keep the UI simple

4 Conclusions

One aspect that is not covered in this project is power. Power would be considered a limitation. The Raspberry Pi gets its power from a Micro USB and has an input voltage of 5v and consumes 2Amps. When the vehicle is running, the alternator will be able to handle this no problem as most cars systems are regulated to produce in and around 12v power.

A cars battery outputs 12.7V and an average amp hours of 45-50Amps which would be more than adequate for powering the device. However, this will put a drain on the battery and if the car is stationary for long periods of time, will result in the battery dropping below 11.7v and at this voltage the car will not be able to start, and the damage monitoring system will be deactivated. This will need to be considered when applying it to a vehicle. The system would also have to be fused to prevent a power surge from destroying the hardware components of the system.

This will not impact development of the project but it would be something that needs to be considered if it is put into live production. Solutions to this could be an auxiliary battery (this is common practice in many cars) or a “Sleep Mode” being developed for the system to minimise power.

One of the main advantages of this project is its size and universal fit. The components are all small and the sensor system does not be built into the car or put behind panels that would be expensive to remove. It can be located in any place on the vehicles frame (boot for example). If the owner buys a new car the system can be moved over for use in that vehicle with minimal effort.

One of the disadvantages is this universal factor. Different vehicles will have different impact thresholds so the accuracy of such a project cannot be determined without extensive testing.

With almost no competition there are plenty of opportunities for this project. As mentioned in the research section below the only similar kind of system acts in a completely different way and is only implemented on new cars designed to use it.

Due to the universal approach to this project it can be implemented as an aftermarket accessory even without support of car manufacturers.

4.1 Research

Research into existing systems

There is currently one similar system in development by Hella. Hellas system puts a sensor on each of the cars body panels to alert a user if one has been damaged. Hellas system is designed to detect scratches to the cars surface panels and any dents or impacts to the panels. This is achieved by placing anywhere from 2-12 sensors on the cars body to monitor for damage. The system is also hooked into the cars on board GPS and camera systems to capture location data and images of the event.

Hellas system proposes to use the vehicles on board GPS and cameras but what if a car doesn’t have these? This system would work well with new cars that come with this technology but even in this case not all new cars will have GPS or Cameras depending on the specification of the car. This also excludes many older models of cars that do not have either of these technologies and prevents them from accessing major functionality of the system.

Even if a car has these technologies there is still a challenge facing Hellas system which is using a cars camera to capture images. Even if a car has camera technology most cars only use cameras as an aid when reversing. This wouldn’t be much use if the car is hit anywhere other than the back. The system outlined in this document would overcome this by placing a camera at A and C pillars of the car both Nearside and Offside to cover the entirety of the car. These cameras will be linked to an independent microcontroller that will store images and capture geolocation data in the event of an impact.

With the use of an independent microcontroller the system outlined in this project can be deployed on any vehicle regardless of how old it is or it’s specification and still monitor impacts that could damage not only the panels but the structure of the car itself.

4.2 Further development

The data collected by the program could be used in a data analysis program to determine the total wear to the vehicles suspension. This could then be incorporated into Vehicle Diagnostic software and give measures of when shock absorbers and dampers should be changed. This would optimize a vehicles road handling and safety. To do this a second threshold would need to be determined that if detected the impact data is collected and stored in the DB without having to alert the user as the impact would not be severe.

System could be evolved over time to include a bank of sensors on each of a vehicles panels. The application could then determine where damage has occurred to a localized section of the car. With finer sensors, not only could impacts be detected but also scratches to the vehicles surface.

Another advancement option would be using fiber optic cable to trace the cars frame and various mechanical components and if bends or warping occur a second monitoring system could alert the user providing more detailed damage feedback.

This application could move beyond the use in light vehicle motor industry into heavy vehicle and even the aerospace industry. Wing clips are a common occurrence in airports where plane's wings contact each other on the runway. A damage monitoring system like this could assess the damage that may have occurred and provide an ability to triage the situation

5 References

References

"Hella Working On System So Cars Can Tell Owners When Body Panels Are Damaged". Autoblog. N.p., 2016. Web. 11 Dec. 2016.

"Motors.Co.Uk - Buy And Sell New & Used Cars Safely". Motors.co.uk. N.p., 2016. Web. 11 Dec. 2016.

"Raspberry Pi • Index Page". Raspberrypi.org. N.p., 2016. Web. 11 Dec. 2016.

"Seeed Studio Bazaar, Boost Ideas, Extend The Reach". Seeedstudio.com. N.p., 2016. Web. 11 Dec. 2016.

"About Sqlite". *Sqlite.org*. N.p., 2017. Web. 7 May 2017.

6 Appendix

6.1 Project Proposal

6.1.1 Car damage monitoring system

Graham Kinsella

X13558787

Graham_kinsella3@hotmail.com

BSc (Hons) in Computing Evening

Software Development

16/10/2016

6.1.2 Objectives

The intent of this project is to build an application that takes input from impact sensors and in the event of an impact displays a warning that the vehicle may have been badly damaged.

The broken-down objectives can be quantified as:

- Research the appropriate technologies required for the intent outlined and which ones will work best with each other
- Create a link between sensors and Raspberry PI
- Create a prototype that raises an alert when sensor input is above a certain threshold
- Add functionality for users Geolocation data to be monitored
- Create second prototype with previous functionality and ability to capture users Geolocation Data if sensor input is above certain threshold
- Create link to Raspberry PI and Cameras to capture a photo in event of sensor input being above a certain threshold (**Optional Objective depending on time constraints/access to resources**)

6.1.3 Background

Many road users are involved in minor collisions daily. Most of these may not seem to have caused much damage as motorists cannot see the immediate damage to a car without being able to access areas often hidden behind panelling or body trim such as bumpers.

Unbeknownst to most people bumpers and other car parts are plastic and reform to their original shape after an impact even though damage to body components behind them may have occurred. This leads to many motorists leaving the scene of an accident thinking no damage bar superficial surface scratches have occurred only to find out at a later date that their car is badly damaged/wrote off.

Another reason for the development of this application is in regards to another common issue faced by motorists. A car could be parked at one place for a number of hours and hit by another vehicle. When the owner returns they may not immediately notice and travel to many other destinations before finally realising their car has been damaged at some point in the journey.

With this application the location of the impact will be recorded and the user can then view it and return to the location and possibly access witnesses or CCTV footage. There will also be the possibility of a camera being hooked to the system that will also record a picture of the impact with the hopes of capturing a picture of the perpetrating vehicle which could be used to trace them.

6.1.4 Technical Approach

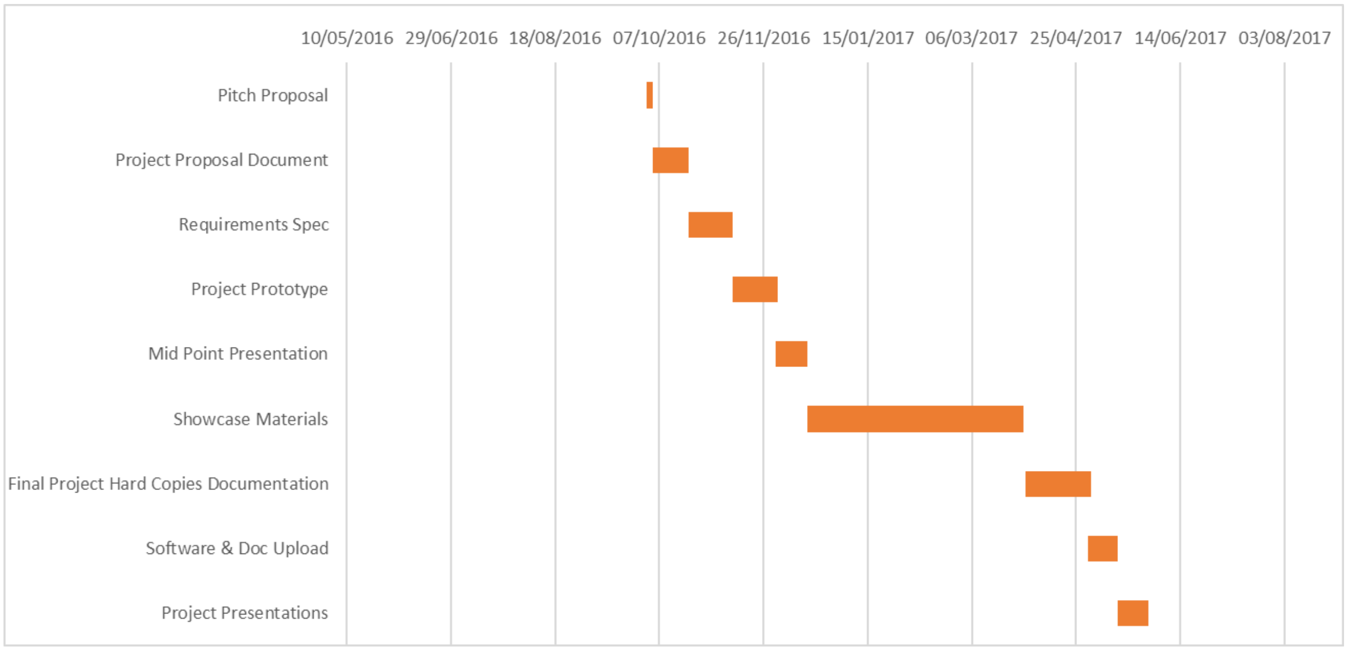
The application will be built on a Raspberry Pi with input from sensors and cameras. It will be written in Python as this is one of the most common languages used in Raspberry Pi development as it can be used to easily interact with the boards GPIO pins. Python is used by many Raspberry Pi developers so will provide access to plenty of online resources. GitHub will be used for version control of code

6.1.5 Special resources required

Hardware components:

- Raspberry Pi
- Impact sensor
- Camera

6.1.6 Project Plan



6.1.7 Technical Details

Software: Written in Python. Plan on using a web based IDE such as Adafruit WebIDE. This will allow me to develop on any computer I am at without installing an IDE. This will be useful when working from home, college and work.

Hardware: Raspberry Pi, Impact sensor, Camera

6.1.8 Evaluation

Unit tests will be used to test code. Unit tests will ensure any defects resulting from changes made to code are noticed and can save time in refactoring code after big changes.

Test cases will be written to ensure input by actors causes appropriate responses. Test cases describe the process flows through a system based on its most likely use. This makes the test cases good for finding defects in the real-world use of the system (i.e. the defects that the users are most likely to come across when first using the system).

In the event sensor data cannot be obtained. A variable will be written and passed into the system to trigger activation. This will provide a proof of concept.

Graham Kinsella 18/10/2016

Signature of student and date

6.2 Monthly Journals

6.2.1 September

Reflective Journal

Student name: Graham Kinsella

Programme: BSHCE4

Month: September 2016

My Achievements

This month I came up with a project proposal and pitched it to a panel of lecturers. They were happy with my idea but my current idea is IOT based and I do not have a module in this so told me to contact a lecturer, Dominic Carr, to help me finalise my idea and point me towards suitable technologies. I contacted Dominic and he pointed out that there are some viability questions to be answered in regards to my idea. I plan on completing my proposal in the coming month and forwarding it to him to see if I can start to develop this project or start looking at other ideas

Intended Changes

Next month, I will try to finalise my idea

Supervisor Meetings

Have not been assigned a supervisor yet

6.2.2 October

Reflective Journal

This month I pitched my idea to a panel of people from the college. My initial research showed that the idea I had come up with might be too hard to do but the lecturers of the panel suggested I pursue it as it was a good idea and told me to get in touch with an IoT lecturer as I don't have an IoT module in my stream.

I emailed a lecturer, Dominic Carr, in regards to my project to see if I would need any more components than I had foreseen. He got back to me and suggested I send him my proposal when I completed it and he would have a look over it. I did this and a couple of weeks later he got back to me saying that it looked fine. Next month I will meet with my project supervisor and complete my requirements spec.

I will also begin to use and get familiar with Raspberry Pi as I have never developed with one before. I am intending to use Python as my primary language but one of my colleagues in work mentioned that C# can now be used with a Raspberry Pi so I will investigate that this month as I primarily use C# in work and I am very familiar with it.

Supervisor Meetings

Have not been assigned a supervisor yet

6.2.3 November:

Reflective Journal

Student name: Graham Kinsella

Programme BSc in Computing

Month: November

This month we were working on completing the requirements specification. I finally nailed down the components I'd need after posting in a couple of forums. So far I'm going with a Raspberry Pi, accelerometer and camera. I may need to add to this but this will be a start.

I met with my supervisor, Eamon, and he made some suggestions for me to include in my requirements spec. I knew my spec was lacking already but due to the sheer volume of work NCI has thrown at us this year I've been struggling to find time to do the requirements spec

From now I plan on putting together a prototype to see if I'm missing any other components but with 5 other projects and numerous CAs due before the prototype for this module that might go on the backburner for now.

My Achievements

This month I was able to determine some of the components I'll need for this project

My Reflection

Need a 28 hour day

Intended Changes

Next month I will continue my research into what components are best suited for this project

6.2.4 December

Reflective Journal

Student name: Graham Kinsella

Programme BSHC hons in Computing

Month: December

This month was spent prepping the technical report for upload and developing a prototype to demo at the end of semester during the mid-point presentations.

I spent a bit of time doing both to make sure they were ok. I’d neglected my original requirements a bit due to all the other uploads we had around the same time for our other 5 modules but I wanted to make sure this final upload was good quality.

I contacted a couple of the mechanics I’ve worked with/served my apprenticeship with and asked them for feedback on the project. All who replied thought it was a great idea.

I started to get familiar with the Raspberry Pi and other components this month. I’ve never used one or even seen one being used before. Developing on them is interesting. I wrote my demo via Python Shell and made it executable through command line but as it starts to get bigger I’ll swap to using an IDE (Ninja).

Next month I’m going to start writing scripts to capture the geolocation data and vibration strength as variables that I can use and messing with the vibration sensor I have. It seems to be what I need but I want to make sure it is 100% fit for purpose, if not I’ll order another.

I also need to figure out how to use GitHub as source control on my Raspberry Pi this month.

My Achievements

- Constructed a working prototype
- Learned how to use a Raspberry Pi
- Learned how to use the GrovePi+ and associated sensors
- Wrote a script to capture image at a certain threshold

Supervisor Meetings

Date of Meeting:

29/11/2016

Items discussed:

technical report, prototype

Action Items:

Continue with technical report.

Outlined what would be expected for a prototype of this software

6.2.5 January

Reflective Journal

Student name: Graham Kinsella

Programme BSHC hons in Computing

Month: January

My Reflection

This month was spent mostly preparing for exams. Last semesters workload was fairly heavy so I'm going to take a week sanity break after the exams are over. I've done a couple of python tutorials to get up to speed with the language. I thought with being familiar with Java/C# it'd be easy to pick up but I'm being caught out with how to extend classes or even with indenting so I thought doing tutorials would be a good place to start before heavy coding begins

My Achievements

- Got all other projects done/exams
- Survived to the second semester
- Learned some more advanced python

Supervisor Meetings

Due to exams/college being closed a supervisor meeting didn't occur this month

6.2.6 February

Reflective Journal

Student name: Graham Kinsella

Programme BSHC hons in Computing

Month: February

My Reflection

This month proper development started on the project. I ordered a new GPS sensor so I didn't have to rely on network connectivity. The idea here is that cars don't have internet so I needed a way of monitoring location independent of it. Got it set up and have the GPS data, sensor data and image storing in a DB. GPS data is coming back blank because I'm in an apartment but I've tested it outside and it works fine so it should work perfectly in a car.

I'm still not convince that this vibration sensor I'm using is right for the job at hand so I'm going to have to do some extensive testing to make sure because I don't want any last-minute surprises. It's working grand at the minute for development purposes but when it comes to a working prototype I think I'll need something else.

Also, managed to get Git set up on the Raspberry Pi. Wasn't working on first install due to a corrupt package so I cleared it and uploaded my code to GitHub.

Thinking about how to display the records to a user at the minute. I originally had intended on creating a GUI myself locally on the device but I don't think it's the best approach as the user must plug a monitor in and have one at hand anytime they want to check the device. Thinking a web app will be better so the records can be accessed from a mobile device as well as a pc.

Got images storing in the DB instead of the file system as was the approach in my demonstration in December. Have them stored as a blob in the database and just need to figure out how to serialize them now. I'm still debating leaving them in the file system and storing the path in the DB but this defeats my original purpose of using a sqlite db to hold everything due to the convenience of it.

Leaves me with a couple of tasks for next month.

My Achievements

- Found a flaw in my Vibration sensor early in development
- Identified and implemented a working GPS sensor
- Storing images as blobs in database just need to figure out how to retrieve them now

Supervisor Meetings

Date of Meeting: 27/02/2017

Items discussed: Current issues with sensors, ongoing research

Action Items: Continue with current course of action

6.2.7 March

Reflective Journal

Student name: Graham Kinsella

Programme BSHC hons in Computing

Month: March

My Reflection

Development was going well with the project but as suspected my Vibration sensor is not fit for purpose. I did extensive research into other types of sensors and finally decided on a 3d accelerometer. It should work perfectly for this project; I had hoped to measure vibration intensity but it doesn't look like I can do that with sensors at hand.

The accelerometer will measure movement of the devices environment(vehicle) across 3 axes and I can write code that determines acceleration from this. If I measure the sudden change in acceleration I can determine if an impact has occurred.

There are numerous types of accelerometers but I settled on a 16g one. I researched car crashes and it looks like the most they ever get to is around 8gs so this should be more than adequate. If the person is in a crash that puts out over 8gs they won't need my device to tell them their car is broken, the paramedics will take care of that job…

Also, ordered a new better LED and button for my device. I had originally intended on having the warning and reset button displayed on the GUI but I think this goes against the idea of the project. So now when an impact occurs the LED will be lit up until the user pushes the button to reset it. The idea here is that this can be integrated into the cars LED warning systems on the dash and stay lit up until the user goes to a garage to get the car inspected. The button logic can then be replaced with being reset with an OBD (OnBoard Diagnostic) reader that is used in all garages.

As for last month's action items I solved my image problem by changing how they are stored, I was storing them as a raw text blob in the database but now I am storing them as a Base64 encoded string. This has allowed me to serialize them in either python or as my next point will elaborate on, in HTML.

I also scrapped the idea of developing GUI on the device and instead opted for using a web app. With becoming more familiar with developing in Python I could make a web app in python with a html template. Each record from the DB is displayed on this web app and the images are serialized here to become images again. This is pretty much my main functionality finished.

Next month I want to get the site hosted on a cloud hosting site such as pythonanywhere or Heroku but I will need to research which is best. Next month in work brings one of the biggest releases the company has ever had so I'm expecting to have to work a lot of overtime which I know is going to affect this project. During our last release, I was doing 12 hour days and working weekends and the date this time coincides with the end of college which is unfortunate as it will be my busiest time in NCI too. Having the project in a good state already should help.

My Achievements

- Got main functionality working
- Identified and ordered a new sensor
- Created Web App
- Improved application design
- Identified how to deal with images

Supervisor Meetings

Date of Meeting: 14/03/2017

Items discussed: New sensors, ideas for web app, end of college

Action Items: remember to do work for upcoming showcase (poster, report, profile)

Continue work on application

6.3 Other Material Used

An email was sent to 20 garages explaining the general Idea of this project and asking the mechanics employed there for their opinion on this technology. The reasoning behind this was that mechanics would interact with more vehicles, especially damaged ones, than the general public. While a vehicle owner might have 1-2 cars in a household the average mechanic would see 10-20 cars per day and interact with their owners. Generally, the owners will relay how damage occurred (e.g. crash, dent and run). At the time of writing, of these 20, 6 replied and the results of this survey are listed below.

"*This would be a great idea, I've had more people than I can count whose cars had been hit in shopping centres or that and then they're left to pay for the damage. Sure it happened to me a few weeks ago*"

"*That'd be savage, I had a Range Rover in a few weeks ago and the owner had been rear ended on the M50 on the way to work but he seen it looked alright when he checked so he just drove on, had to replace his whole sub frame all in all cost himself about 1800 euro"*

"I'd install it on my car, be a good idea to try hook it into the cars multifunction display if you could"

"…Car manufacturers are always looking for one ups on each other this would be snapped up I reckon, something new and sounds like it's cheap enough to make"

"Don't know how many times I've heard people saying their cars have been hit or scratched when it was parked and nobody was around"

"…Might help catch a few of them clowns that are robbing cars and always crashing into parked cars…."

Of the 6 replies all 6 were positive.