Face Detection System to Monitor Property Security

ECE 492 Winter 2020
Progress Report

April 10, 2020

Group 1

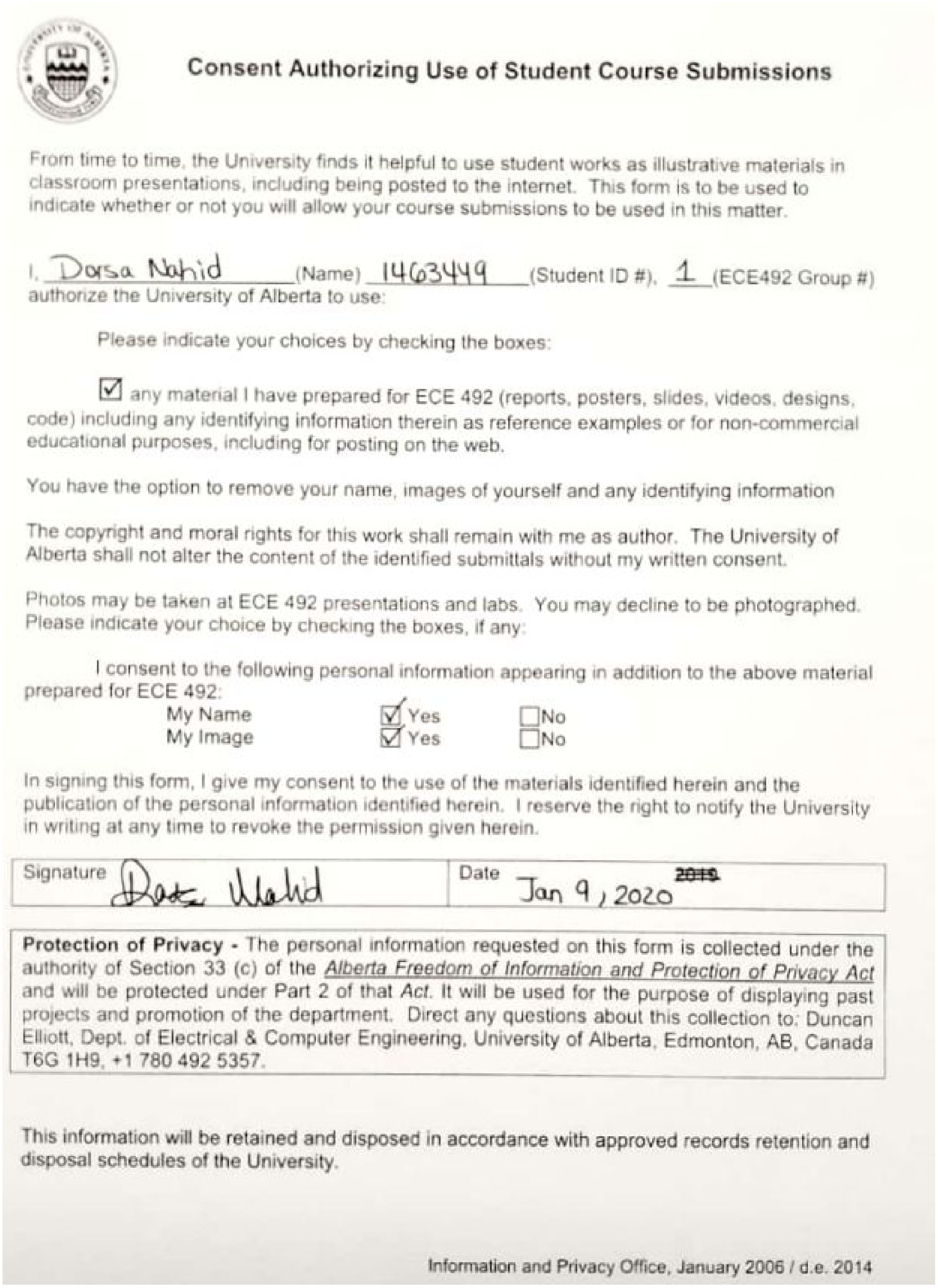
Declaration of Original Content

The design elements of this project and report are entirely the original work of the authors and have not been submitted for credit in any other course, except as follows:

1. Web form template provided by Colorlib [29]

Tymoore Jamal	1452978	[illegible]	March 28, 2020
Bryn Leonard-Fortune	1469246	[illegible]	March 28, 2020
Jessica D'Cunha	1463456	[illegible]	March 28, 2020
Dorsa Nahid	1463449	[illegible]	March 28, 2020

Consent Forms



Consent Authorizing Use of Student Course Submissions

From time to time, the University finds it helpful to use student works as illustrative materials in classroom presentations, including being posted to the internet. This form is to be used to indicate whether or not you will allow your course submissions to be used in this matter.

I, Dorsa Nahid (Name) 1463449 (Student ID #), 1 (ECE492 Group #) authorize the University of Alberta to use:

Please indicate your choices by checking the boxes:

☑ any material I have prepared for ECE 492 (reports, posters, slides, videos, designs, code) including any identifying information therein as reference examples or for non-commercial educational purposes, including for posting on the web.

You have the option to remove your name, images of yourself and any identifying information

The copyright and moral rights for this work shall remain with me as author. The University of Alberta shall not alter the content of the identified submittals without my written consent.

Photos may be taken at ECE 492 presentations and labs. You may decline to be photographed. Please indicate your choice by checking the boxes, if any:

I consent to the following personal information appearing in addition to the above material prepared for ECE 492:

My Name	[✓ Yes]	No
My Image	[✓ Yes]	No

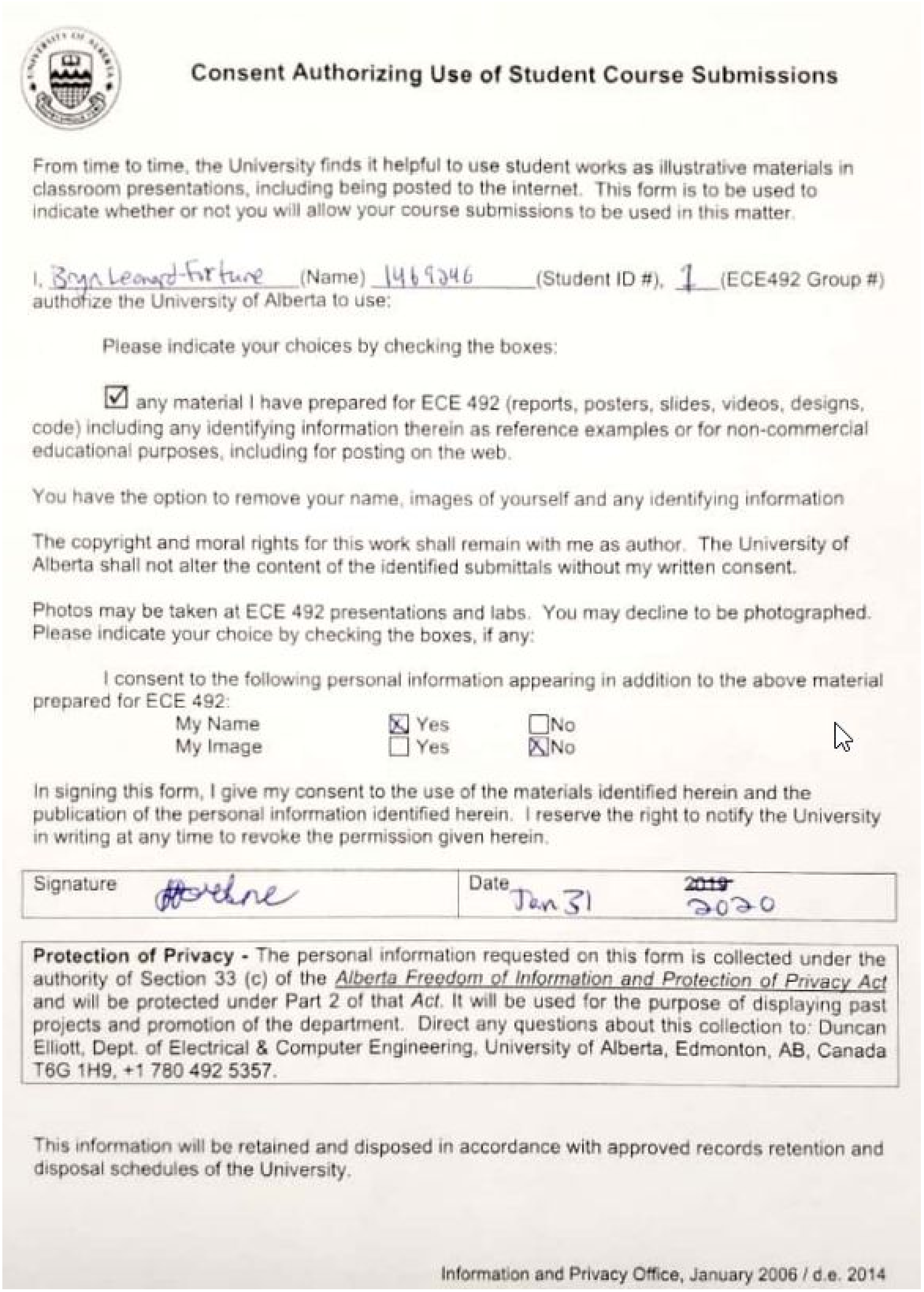
In signing this form, I give my consent to the use of the materials identified herein and the publication of the personal information identified herein. I reserve the right to notify the University in writing at any time to revoke the permission given herein.

Signature [illegible]	Date Jan 9, 2020 ~~2019~~

Protection of Privacy - The personal information requested on this form is collected under the authority of Section 33 (c) of the *Alberta Freedom of Information and Protection of Privacy Act* and will be protected under Part 2 of that *Act*. It will be used for the purpose of displaying past projects and promotion of the department. Direct any questions about this collection to: Duncan Elliott, Dept. of Electrical & Computer Engineering, University of Alberta, Edmonton, AB, Canada T6G 1H9, +1 780 492 5357.

This information will be retained and disposed in accordance with approved records retention and disposal schedules of the University.

Information and Privacy Office, January 2006 / d.e. 2014



Consent Authorizing Use of Student Course Submissions

From time to time, the University finds it helpful to use student works as illustrative materials in classroom presentations, including being posted to the internet. This form is to be used to indicate whether or not you will allow your course submissions to be used in this matter.

I, Bryn Leonard-Fortune (Name) 1469246 (Student ID #), 1 (ECE492 Group #) authorize the University of Alberta to use:

Please indicate your choices by checking the boxes:

☑ any material I have prepared for ECE 492 (reports, posters, slides, videos, designs, code) including any identifying information therein as reference examples or for non-commercial educational purposes, including for posting on the web.

You have the option to remove your name, images of yourself and any identifying information

The copyright and moral rights for this work shall remain with me as author. The University of Alberta shall not alter the content of the identified submittals without my written consent.

Photos may be taken at ECE 492 presentations and labs. You may decline to be photographed. Please indicate your choice by checking the boxes, if any:

I consent to the following personal information appearing in addition to the above material prepared for ECE 492:

My Name	[✓ Yes]	No
My Image	Yes	[✓ No]

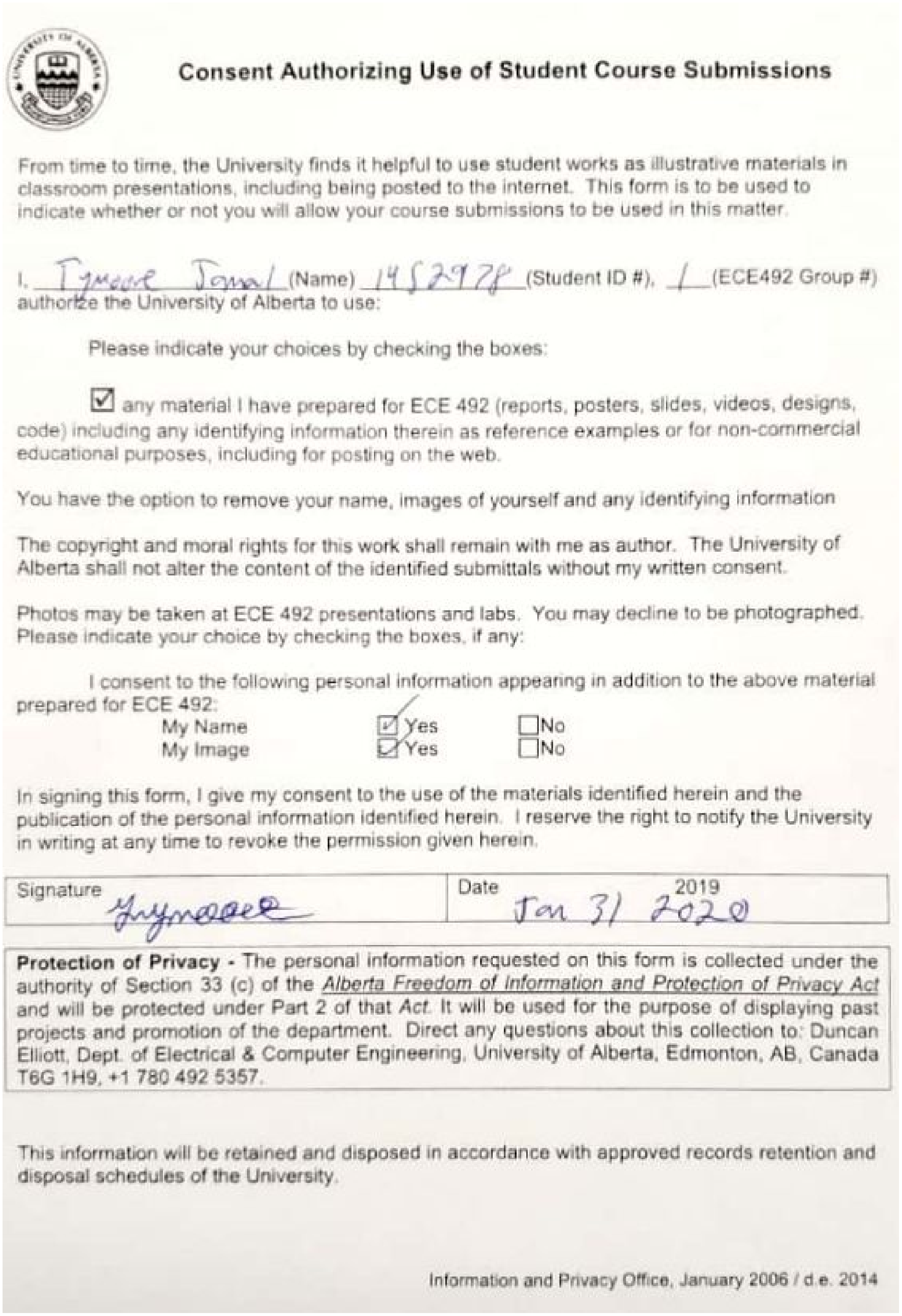
In signing this form, I give my consent to the use of the materials identified herein and the publication of the personal information identified herein. I reserve the right to notify the University in writing at any time to revoke the permission given herein.

Signature [illegible]	Date Jan 31 ~~2019~~ 2020

Protection of Privacy - The personal information requested on this form is collected under the authority of Section 33 (c) of the *Alberta Freedom of Information and Protection of Privacy Act* and will be protected under Part 2 of that *Act*. It will be used for the purpose of displaying past projects and promotion of the department. Direct any questions about this collection to: Duncan Elliott, Dept. of Electrical & Computer Engineering, University of Alberta, Edmonton, AB, Canada T6G 1H9, +1 780 492 5357.

This information will be retained and disposed in accordance with approved records retention and disposal schedules of the University.

Information and Privacy Office, January 2006 / d.e. 2014



Consent Authorizing Use of Student Course Submissions

From time to time, the University finds it helpful to use student works as illustrative materials in classroom presentations, including being posted to the internet. This form is to be used to indicate whether or not you will allow your course submissions to be used in this matter.

I, Tymour Jamal (Name) 1452978 (Student ID #), 1 (ECE492 Group #) authorize the University of Alberta to use:

Please indicate your choices by checking the boxes:

☑ any material I have prepared for ECE 492 (reports, posters, slides, videos, designs, code) including any identifying information therein as reference examples or for non-commercial educational purposes, including for posting on the web.

You have the option to remove your name, images of yourself and any identifying information

The copyright and moral rights for this work shall remain with me as author. The University of Alberta shall not alter the content of the identified submittals without my written consent.

Photos may be taken at ECE 492 presentations and labs. You may decline to be photographed. Please indicate your choice by checking the boxes, if any:

I consent to the following personal information appearing in addition to the above material prepared for ECE 492:

My Name	[✓ Yes]	No
My Image	[✓ Yes]	No

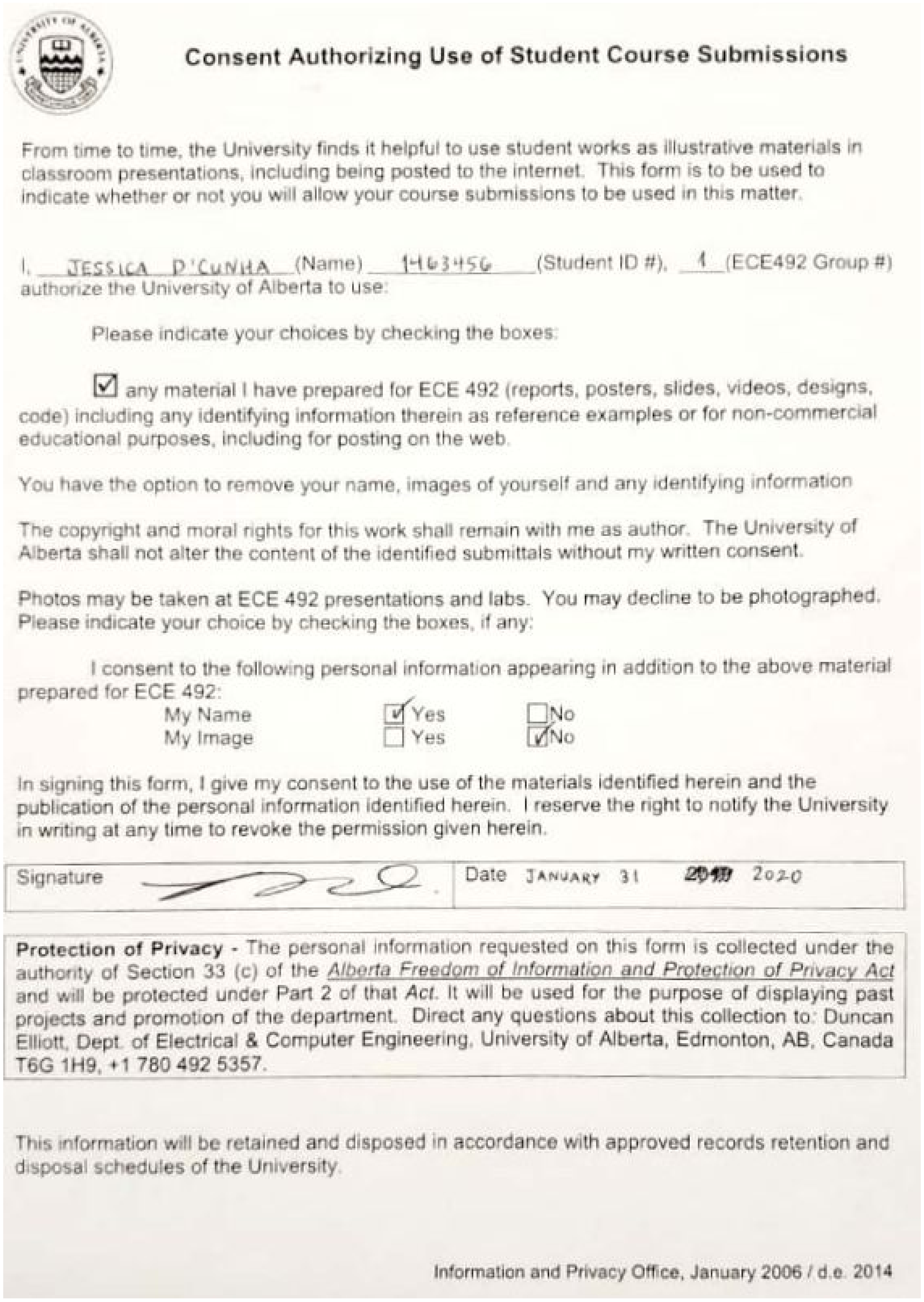
In signing this form, I give my consent to the use of the materials identified herein and the publication of the personal information identified herein. I reserve the right to notify the University in writing at any time to revoke the permission given herein.

Signature [illegible]	Date ~~2019~~ Jan 31 2020

Protection of Privacy - The personal information requested on this form is collected under the authority of Section 33 (c) of the *Alberta Freedom of Information and Protection of Privacy Act* and will be protected under Part 2 of that *Act*. It will be used for the purpose of displaying past projects and promotion of the department. Direct any questions about this collection to: Duncan Elliott, Dept. of Electrical & Computer Engineering, University of Alberta, Edmonton, AB, Canada T6G 1H9, +1 780 492 5357.

This information will be retained and disposed in accordance with approved records retention and disposal schedules of the University.

Information and Privacy Office, January 2006 / d.e. 2014



Consent Authorizing Use of Student Course Submissions

From time to time, the University finds it helpful to use student works as illustrative materials in classroom presentations, including being posted to the internet. This form is to be used to indicate whether or not you will allow your course submissions to be used in this matter.

I, JESSICA D'CUNHA (Name) 1463456 (Student ID #), 1 (ECE492 Group #) authorize the University of Alberta to use:

Please indicate your choices by checking the boxes:

[✓] any material I have prepared for ECE 492 (reports, posters, slides, videos, designs, code) including any identifying information therein as reference examples or for non-commercial educational purposes, including for posting on the web.

You have the option to remove your name, images of yourself and any identifying information

The copyright and moral rights for this work shall remain with me as author. The University of Alberta shall not alter the content of the identified submittals without my written consent.

Photos may be taken at ECE 492 presentations and labs. You may decline to be photographed. Please indicate your choice by checking the boxes, if any:

I consent to the following personal information appearing in addition to the above material prepared for ECE 492:

My Name	[✓ Yes]	No
My Image	Yes	[✓ No]

In signing this form, I give my consent to the use of the materials identified herein and the publication of the personal information identified herein. I reserve the right to notify the University in writing at any time to revoke the permission given herein.

Signature [illegible]	Date JANUARY 31 ~~2019~~ 2020

Protection of Privacy - The personal information requested on this form is collected under the authority of Section 33 (c) of the *Alberta Freedom of Information and Protection of Privacy Act* and will be protected under Part 2 of that *Act*. It will be used for the purpose of displaying past projects and promotion of the department. Direct any questions about this collection to: Duncan Elliott, Dept. of Electrical & Computer Engineering, University of Alberta, Edmonton, AB, Canada T6G 1H9, +1 780 492 5357.

This information will be retained and disposed in accordance with approved records retention and disposal schedules of the University.

Information and Privacy Office, January 2006 / d.e. 2014

Abstract

Property security is an everyday issue in which property owners and residents often struggle to find a balance between ensuring that their property and its inhabitants are safe and reducing the number of false alarms. Our team designed and built a camera-based security system that uses motion detection and face detection to determine whether an intruder is present on a user's property. With our Face Detection Security system, properties can be monitored using a motion detector and security camera. Our security system performs face detection on the security feed, and users are notified of intrusions through an application on their phone. Users are able to take action following a notification of intrusion, as well as view a live feed of their property from which they can control the position of the camera, among other features.

Our security system was developed using an Agile approach. Using scrum, the team worked on the iterative and incremental delivery of the product. This project management framework allowed the team to deliver value quickly. Daily scrums and weekly sprints aided in determining tasks for each team member. We paired Agile Development with Test Driven Development to ensure that issues were detected early and the quality of software was maximized.

Table of Contents

Statement of Problem

When we are away from our properties and think of our loved ones and possessions that inhabit them, we want to be assured that they are safe and secure. If an intruder is present on a property, being alerted to the intrusion by means of a security system is crucial to providing an opportunity for the property owner to take appropriate action. However, false alarms from security systems can be unnecessarily stressful and, if frequent, can result in the property owner taking alerts from their security system less seriously.

User-friendly home security systems have increased in popularity. Ring Video Doorbell is one such system, which leverages motion detection to alert users of possible suspicious behaviour on their property. However, this system relies on motion detection, and cannot detect a human intruder accurately. Our team wants to meet the need for a more technologically advanced and accurate solution by designing and implementing a camera-based security system that also uses face detection in addition to motion detection to more confidently determine if an intruder is present on a user's property. Thus, our system aims to minimize the number of false alarms that solely motion detection-based security systems often create, as motion detectors can easily be set off by harmless triggers such as pets, wind, or falling objects.

Objectives/Goals

Scope of Work

For the purpose of this project, we required that our security system was able to detect motion and perform face detection on a camera feed using a trained machine-learning model. To achieve this, the camera and motion detector interfaced with a Raspberry Pi microcomputer which ran a client application with the face detection machine learning model loaded onto it. The Raspberry Pi is able to communicate via Wi-Fi to a Firebase cloud-based NoSQL database, which is able to, in turn, communicate with an Android application to notify users of intruders on their property. The Android application is paired with the camera by scanning a QR code on the camera's enclosure and will be able to log alerts and allow users to view images of intruders associated with a particular alert. The application also offers features that allow users to view a live camera feed, as well as take action following a break-in by allowing users to customize and contact emergency contacts.

Assumptions

The scope of this project is based on several assumptions. We assume that the camera will be used indoors only and will be placed in a standard-sized room no larger than 7m x 7m to allow for motion anywhere in the room to be detected. [I] As the system relies on face detection, an assumption is that intruders will not be masked. It is also assumed that the system will be plugged in at all times; the scope of the project does not include considering scenarios involving power outages. It also assumes that when the user is at or near the property, the user doesn't want our system activated.

Quantitative Design Specifications

Regarding quantitative design specifications, the camera is able to capture and transmit images to the application at least 5 images of size 1280 px by 780 px per second. The system will be able to capture motion within 7m of the camera. Face detection will only be attempted when motion is detected and will have an accuracy of at least 85%. Successful face detection will cause an alert to be delivered to the user's application in at most 1 minute.

Background Research

Many elements of this project have been researched by experts before us. Face detection and encryption are two well-researched elements. We used the research done on both face detection and encryption to help us determine the most effective methods of detecting faces and ensuring users' privacy when handling camera feed.

Our research on implementation methods for a fast and non-power-intensive system that detects human faces accurately brought us to machine learning. Specifically, deep learning, where we use a wealth of images to train a model to do image classification (what is a face and what is not) using a convolutional neural network (CNN).

The convolutional neural network method of image classification is very popular, and can be very accurate depending on how the machine learning model is trained. We have learned that in essence, convolutional neural networks take in a series of inputs (in our case, pictures), and feed it layers of connected nodes, where each connection between nodes has a weighting. Since these images are labeled as part of our training data with whether they are faces or not, the paths through the convolutional network (more specifically, the node connections) are assigned weights. A convolutional neural network evaluates an image and determines with a certain probability whether an image is a face or not. However, the process is not as simple as just feeding images to a model. Pre-processing the images we will use as our training data is a vital part to ensure accurate results, as is configuring the structure of a neural network.

Determining what the best data is for performing facial detection is still unclear. Several sources agree that in face detection the colors used in images are a key part. Usually, normalizing images to feed into the CNN involves converting the images to black and white. Several sources suggest that using an Adaptive Color to Grayscale (ACGS) conversion algorithm is more effective than traditional conversion methods, which is why we chose to use this method [1][2]. Image segmentation is also an area we explored, which helped us gain more training data[3]. Image segmentation involves separating visual elements in an image so that similar shapes can be identified. A third approach that we took for training our machine learning model was exploring variations of CNN structures (for example, variations in layers, nodes per layer, dropout rate, activation functions, etc.) while heavily leaning towards the simple methods suggested by Shin, Kim, and Kwon in their study on CNN structures as it relates to processing faces [4].

Encryption is also an area that our project wants to expand into. As we are storing personal information and images of a user's home, as well as streaming this data using wifi between our Firebase online server, our android application, and our Raspberry Pi, we need to ensure that security is our top priority. Encryption is one method of doing this. Using this study on image encryption for multimedia applications, we have found that in order to balance the speed and security both required by our security system, that we must encrypt data before storage and that we should compress and encrypt images at the same time using standardized techniques, like Data Encryption Standard (DES) [5].

Hardware Requirements

Data Communication Requirements

For our live-feed functionality, we are transmitting a minimum of 5 images each second. These images will be 720p HD images. These images have a size of 1280 x 720 pixels each with 3 bytes for each RGB pixel (one byte for red, one byte for green, and one byte for blue).

<u>Transmission Rate Needed for Uncompressed Images:</u>
(5 images/second) x (1280 x 720) pixels x (3 bytes / pixel) = 13,824,000 bytes/second

<u>Transmission Rate Needed for Uncompressed Images With (90% compression):</u>
(5 images/second) x (1280 x 720) pixels x (3 bytes / pixel) x 0.1 =1,382,400 bytes/second

Therefore when compressing images, we would have a transmission rate of 1,382,400 bytes/second from the Raspberry Pi 4 microcomputer to the Firebase cloud database.

Power Requirements

At maximum power, our Raspberry Pi 4 microcomputer uses 5V and 3A [10]
Pmax = Imax x Vmax = 5V x 3A = 15 W
Therefore, we need a power source that could sustain at least 15W.

Memory Requirements

The main memory requirements include loading the face detection model in RAM and storing an image queue to handle incoming images.

Experimentally we have found that the face detection machine learning model has a size of 80 MB. Each second we are receiving 5 images, therefore we have elected to make our queue have a size of 5 images to give our face detection system a full second to catch up to the camera at any point. Before any processing is done, each image is of size 2592 x 1944 pixels, with each pixel having a size of 3 bytes. Therefore we need a buffer size of 2592 x 1944 x 3 x 5 = 76 MB

Therefore in total, we need a memory size of at least 80 MB + 76 MB = 156 MB.

Bill of Materials

Material	Supplier link or Source	Datasheet link	Cost	Status
Raspberry Pi 4 Kit	https://www.amazon.ca/LABISTS-Raspberry-Heatsinks-Necessary-Accessories/dp/B07XLMVYJJ/ref=sr_1_1_sspa?gclid=EAIaIQobChMI_YK9qcik5wIV0xZ9Ch0xFwPGEAAYASAAEgKPzfD_BwE&hvadid=372957068131&hvdev=c&hvlocphy=9001384&hvnetw=g&hvpos=1t1&hvqmt=b&hvrand=15533687259358706 65&hvtargid=kwd-849202219089&hydadcr=24917_10283276&keywords=labists+raspberry+pi+4+kit&qid=1580155064&smid=A3ED0UV10SQ3A&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUFUSEM4VFdBQVVTQ0EmZW5jcnlwdGVkSWQ9QTA3NzY2OTQyM0wzSktHU1pVTk5HJmVuY3J5cHRlZEFkSWQ9QTA5ODEyNDUzSFZFSU5VM1FOTlI3JndpZGdldE5hbWU9c3BfYXRmJmFjdGlvbj1jbGlja1JlZGlyZWN0JmRvTm90TG9nQ2xpY2s9dHJ1ZQ==	https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2711/rpi_DATA_2711_1p0_preliminary.pdf	$121.54, free shipping, with taxes: $127.62	Received
HC-SR501 PIR Motion Detector (3)	https://www.amazon.ca/Aukru-HC-SR501-Pyroelectric-Infrared-Detector/dp/B019SX6ZR6/ref=pd_sbs_469_2/137-0968925-7166753?_encoding=UTF8&pd_rd_i=B019SX6ZR6&pd_rd_r=0b	https://www.mpja.com/download/31227sc.pdf	$10.99, free shipping, with taxes: $11.54	Received

	445a56-5f0a-458b-b577-c1048faa83d8&pd_rd_w=WFahj&pd_rd_wg=3Lwfk&pf_rd_p=dbebb38c-0e3d-4a67-ac15-432d7c7a2789&pf_rd_r=VJFA25M07BA4RFJ4M0P6&psc=1&refRID=VJFA25M07BA4RFJ4M0P6			
Raspberry Pi Camera Module v2	https://www.amazon.ca/Raspberry-Camera-Official-8-Megapixel-LABISTS/dp/B07W921DCT/ref=sr_1_1_sspa?gclid=EAIaIQobChMI99jOrs6k5wIVbiCtBh3hhQ3rEAAYASAAEgKCT_D_BwE&hvadid=208410826925&hvdev=c&hvlocphy=9001384&hvnetw=g&hvqmt=e&hvrand=15657036253844672351&hvtargid=kwd-296167130300&hydadcr=1532_9454498&keywords=raspberry+pi+camera+noir&qid=1580156695&sr=8-1-spons&psc=1&spLa=ZW5jcnlwdGVkUXVhbGlmaWVyPUEyWUlCN1o5OTg1SjZLJmVuY3J5cHRlZElkPUExMDE3MjA0MUlZUE5LNlRRVVNTNiZlbmNyeXB0ZWRBZElkPUEwMjI0MTEzMU5IUjhLOUg3TFhRTiZ3aWRnZXROYW1lPXNwX2F0ZiZhY3Rpb249Y2xpY2tSZWRpcmVjdCZkb05vdExvZ0NsaWNrPXRydWU=	https://docs.rs-online.com/80e5/0900766b8127db33.pdf	$39.99, free shipping, with taxes: $41.98	Received
Wires, breadboard, soldering equipment, and LEDs	Breadboard and LEDs are supplied from the teams members' personal stock. Our wires are sourced from the lab supplies.	https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/239_Web.pdf	Free	Received

Pan Tilt 2 Axis FPV Camera Gimbal Mount Bracket w/ 2 Servos	https://www.amazon.ca/Bracket-Support-Ultrasonic-Mounting-Fixed-Wing/dp/B07QZ3NSKX/ref=sr_1_15?keywords=gimbal%20servo&qid=1580501386&sr=8-15	http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf	$38.30, free shipping, with taxes: $40.22	Received
3pcs 15 Pin 30cm 50cm 100cm FFC Ribbon Flexible Flat Cable for Raspberry Pi Module Camera	https://www.amazon.ca/Ribbon-Flexible-Raspberry-Module-Camera/dp/B07DNYM8KC/ref=sr_1_3?crid=47C9EZ5R9ZEC&keywords=raspberry+pi+camera+ribbon&qid=1582952343&sprefix=rasp%2Caps%2C181&sr=8-3	N/A	$13.89, free shipping, with taxes: $14.58	Received
All elements	***All parts available from Amazon broker***	***All schematics available***	***Total = $235.94***	***All received***

Table 1: Parts List and Aggregate

Datasheet

User-Perspective Block Diagram

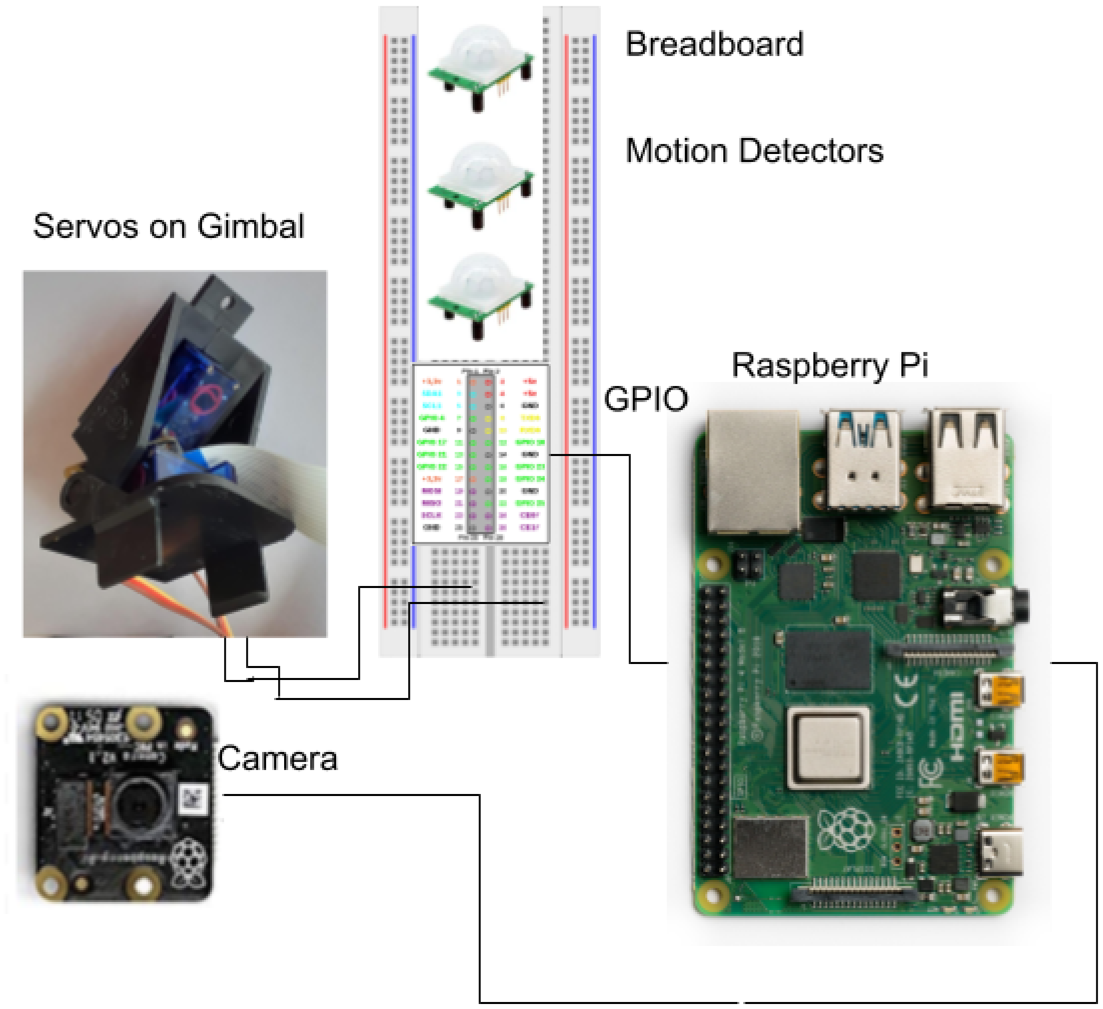


Figure 11.User Block Diagram

Operating Conditions Power/Performance Calculations

See the Sustainability section for more information on how this is calculated. Please note that due to the COVID-19 Pandemic, we were unable to calculate experimental measured power.

	Raspberry Pi (polling + wifi) [33]	Shooting Video [32]	LEDs	Motion Detectors [30]	Motors [31]	Total power
Standby State	575mA*5V = 2.875W	0	0	50uA*5V*3 =750mW	5v*10mA*2 =0.1W	=2.92W

Max	1200mA*5V =6W	250mA*5V = 1.25W	3.3V*9mA *5 LEDs = 0.149W	5V*65mA*3 =0.0975W	5V*250mA*2 =2.5W	=8.68W
Best Guess	95% standby +5% max = 0.95*(2.88W) +0.05*(6W) =3.03W	95% standby +5% max =0.95*(0)+0.05 *(1.25W) =0.0625W	99% standby + 1% max =0.015W	92% standby + 8% max =0.07869W	=99% standby + 1%max =0.124W	=**4.02W**

Table 6. Power Calculations for Our System

IO Signals

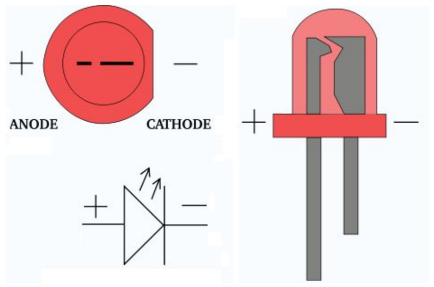
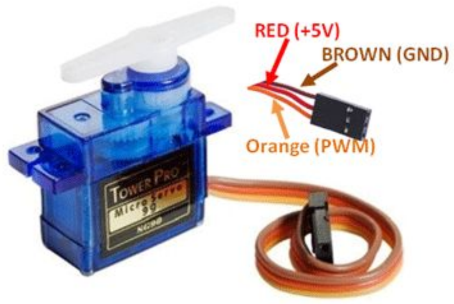
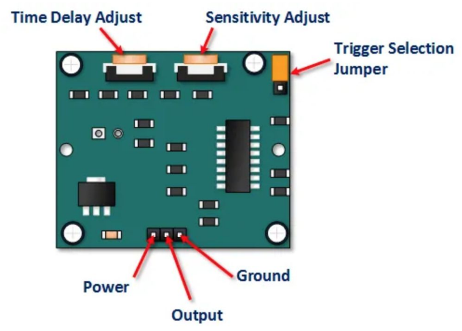
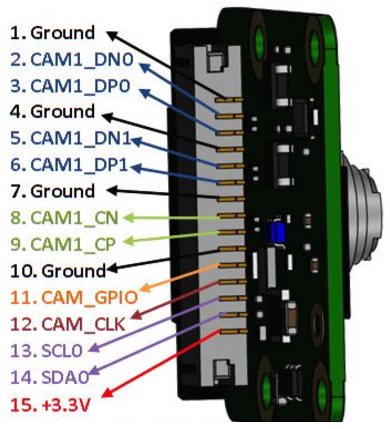
Connection	Name	Voltage	Signal Description
Figure 1: LED 	VCC	2V-4V Using (3.3V)	Power to turn on or off the LED
	GND	0V	Ground
Figure 2: SG90 Servo Motor  RED (+5V) BROWN (GND) Orange (PWM) TowerPro Micro Servo 9g SG90	VCC	4.8V - 6 V (5V)	Provides power to the servo motor
	PWM	3.3V - 5V	Pulse width modulation signal. The duration of the signal indicates the desired rotation of the motor
	GND	0V	Ground

Figure 3: HC-SR501 Motion Detector - Raspberry Pi GPIO [21]  Time Delay Adjust Sensitivity Adjust Trigger Selection Jumper Power Output Ground	VCC	5V-20V. Raspberry Pi provides 5 volts, so we are using 5V	Provides power to the Motion Detector
	OUT	0V - 3.3V, where 0V is no motion detection	High voltage (3.3V) indicates that motion has been detected within 3-7m.
	GND	0V	Ground
Figure 4: Raspberry Pi Camera Module V2.1 - Raspberry Pi Camera Slot [22]  1. Ground 2. CAM1_DN0 3. CAM1_DP0 4. Ground 5. CAM1_DN1 6. CAM1_DP1 7. Ground 8. CAM1_CN 9. CAM1_CP 10. Ground 11. CAM_GPIO 12. CAM_CLK 13. SCL0 14. SDA0 15. +3.3V	GND (x4)	0V	Ground
	CAM1_DN 0	Uses Z, so anything above 0V (not more than 3.3V sent)	MIPI Data Negative of data lane 0
	CAM1_DP 0	Uses Z, so anything above 0V (not more than 3.3V sent)	MIPI Data Positive of data lane 0
	CAM1_DN 1	Uses Z, so anything above 0V (not more than 3.3V sent)	MIPI Data Positive of data lane 1

	CAM1_DP 1	Uses Z, so anything above 0V (not more than 3.3V sent)	MIPI Data Positive of data lane 1
	CAM1_CN	Uses Z, so anything above 0V (not more than 3.3V sent)	MIPI Clock Negative Pulse
	CAM1_CP	Uses Z, so anything above 0V (not more than 3.3V sent)	MIPI Clock Positive Pulse
	CAM_GPI O	Uses Z, so anything above 0V (not more than 3.3V sent)	GPIO Pin (optional)
	CAM_CLK	Uses Z, so anything above 0V (not more than 3.3V sent)	Clock Pin (optional, CN, CP used typically)

	SCL0	Uses Z, so anything above 0V (not more than 3.3V sent)	Serial interface clock input for I2C communication
	SDA0	Uses Z, so anything above 0V (not more than 3.3V sent)	Standard serial interface for I2C communication
	VCC	3.3 V	Provides Power to the camera

Table 2: Consolidated IO Signal Table

Printed Circuit Board Design

Our hardware involved the Raspberry Pi 4, three HC-SR501 PIR Motion Detectors, the Raspberry Pi Camera Module v2 Noir, two Tower Pro servo motors, and connecting wires. Our design is simple in that our only connections are from the camera to the Raspberry Pi (using a camera module slot), the Motion Detectors to the Raspberry Pi, and the motors to the Raspberry Pi, using wires.

Figure 5 illustrates the schematic for the PCB, designed in EAGLE. This schematic can then be converted to a board design using EAGLE and used for manufacturing the board in the ELKO Engineering Garage, which would subsequently be soldered to component connections. Due to the COVID-19 pandemic restricting access to the necessary equipment, the PCB could not be finished.

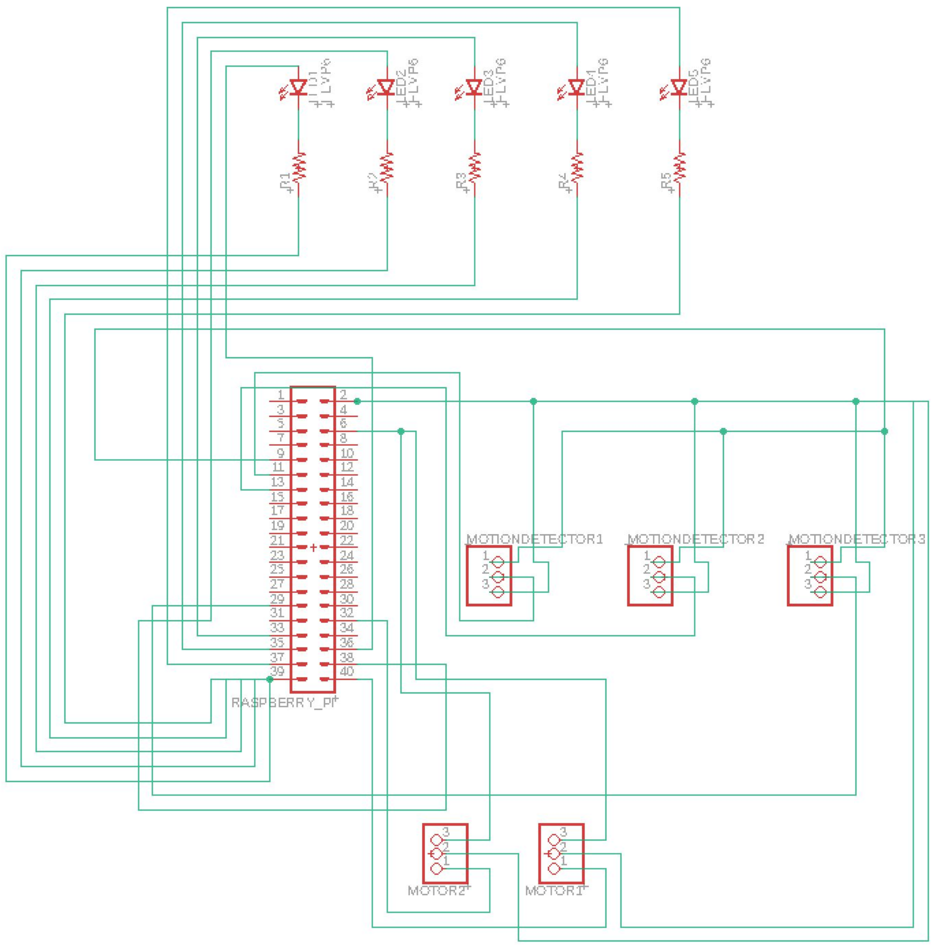


Figure 5: PCB Schematic

Software Design

Our project connects a video feed, motion detectors, a Raspberry Pi, a Firebase database, and an Android application to analyze camera feed upon detection of motion, to determine if the camera feed contains faces of humans, and keep users informed of face detections and break-ins through messages and stored logs.

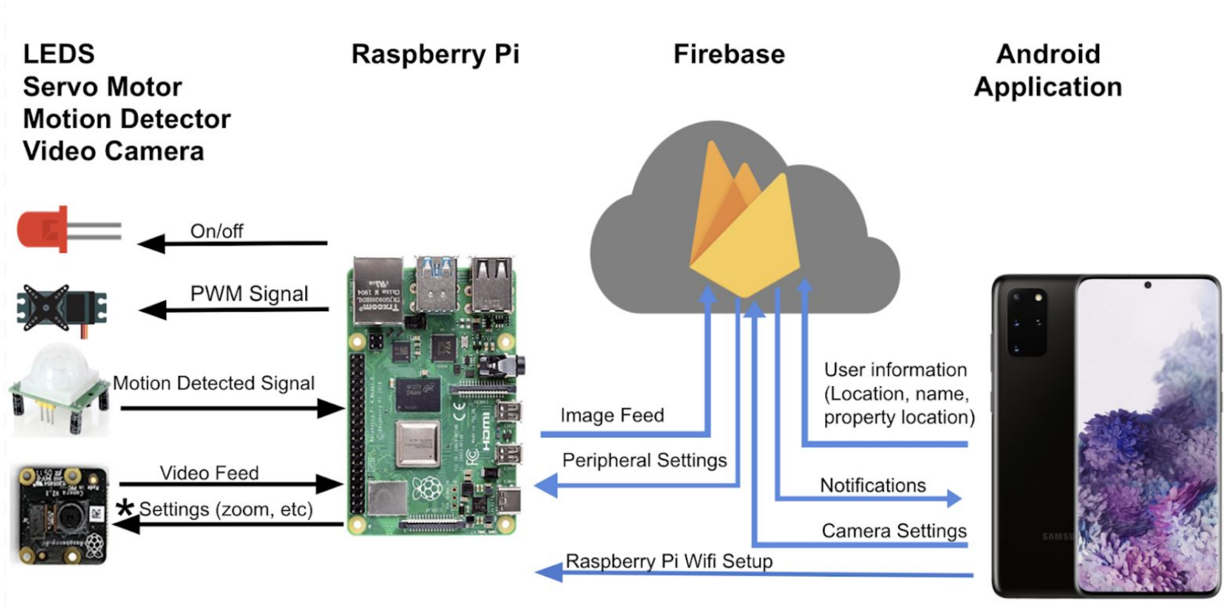


Figure 6. Data flow between project elements

Our face detection system can be broken down into peripheral control through the Raspberry Pi, deep learning model creation, running the face detection model, Firebase, and Android application design.

Deep Learning Training

Our deep learning model has been made using the popular Tensorflow library and leverages the Keras library, and python code has been written to interface with these libraries. Our initial models have been trained on our personal computers, since they have enough RAM and GPU to train models with a few thousand training images. The next step is to use a ‘multiple fitting’ method that reuses the same dataset (subset of training images) to build our models and recreate the CNN node weightings that will be used to determine whether an image contains a face or not, if initial experimentation with multiple fitting doesn’t lead to overfitting and improves our accuracy.

Another option we explored was using the University of Alberta ECE department's AI centre, which maintains supercomputers with the high RAM and GPU that we would need to train our models on a large set of training data in a reasonable amount of time. Our hope with these supercomputer VMs was that we can achieve higher accuracy of the model and higher efficiency in terms of time when developing the model due to their large RAM allowing us to train the model with significantly more images. However, through model size optimizations we eliminated the need for these supercomputers as we were able to achieve a face detection accuracy of 97% without them.

Peripheral Control

The Raspberry Pi runs python scripts to control the motion detector, the motors, and the camera stream. For the motion detector, we use the RPi.GPIO library to pick up the motion detector's motion detected signal through the GPIO pins wired to our motion detector. Once a movement threshold has been met, the script instructs our camera to begin taking a stream of pictures at regular intervals through the camera's python script, which uses the PiCamera library.

Face Detection

These captured photos are converted to PIL pictures using the Pillow library, where we can easily alter the resolution and colour control of the images. Using PIL images, we preprocess the images by altering the colouring and splitting the images into manageable sections that we will pass through our trained machine learning model. We gather the result that indicates whether there is a face present or not.

Raspberry Pi Communication

After getting the result of our model (face or no face), we convert the photos to a base 64 string, then use our Firebase.py file to transmit the image data. This string is encrypted through HTTPS; as all communication with Firebase uses HTTPS.

Firebase

Google's Firebase platform includes eighteen products that support web and mobile app development. We are using several of these solutions, including the Firebase real-time database, Firebase Auth, and Firebase In-App Messaging. Firebase's main roles in our project involve acting as a secure database for logging face detection instances and user settings and information, as well as acting as a communications relayer between the user application and the Raspberry Pi. It transmits camera feeds, notifications, and configuration settings.

Below is a UML diagram of our Firebase database schema, which translates to a similar class model for the Android application. The security camera and user each have their own settings information that we will store. Each user will also have a series of notifications (alerts of intruders) with a notification image (the face detected) associated with a notification. Our database schema allows for easy reading and writing of settings, as well as allowing easy calls to get a user’s notification logs without slowing down to load images unnecessarily. In this way, our database design compliments our app in that it focuses on user experience in utility as well as speed.

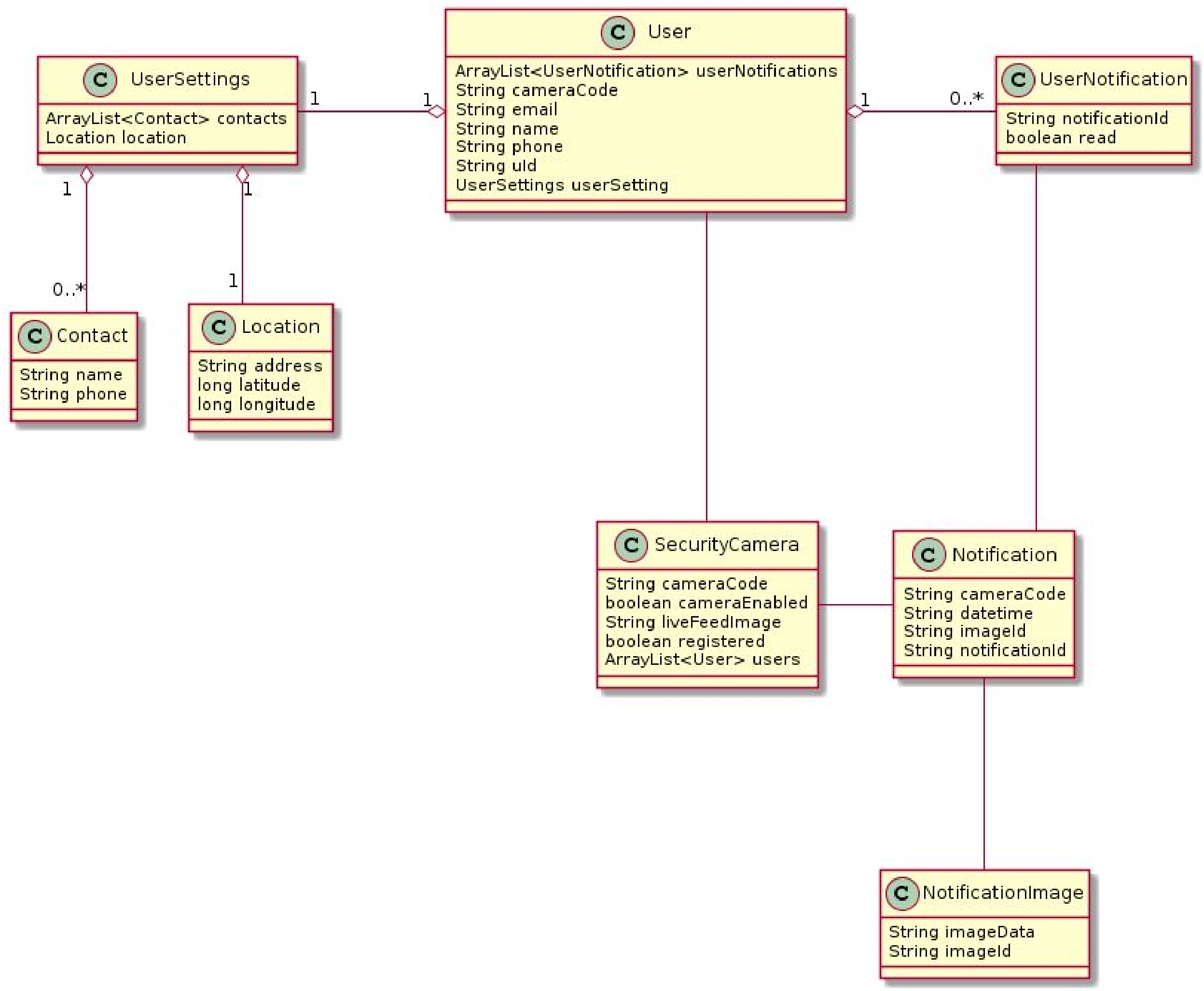


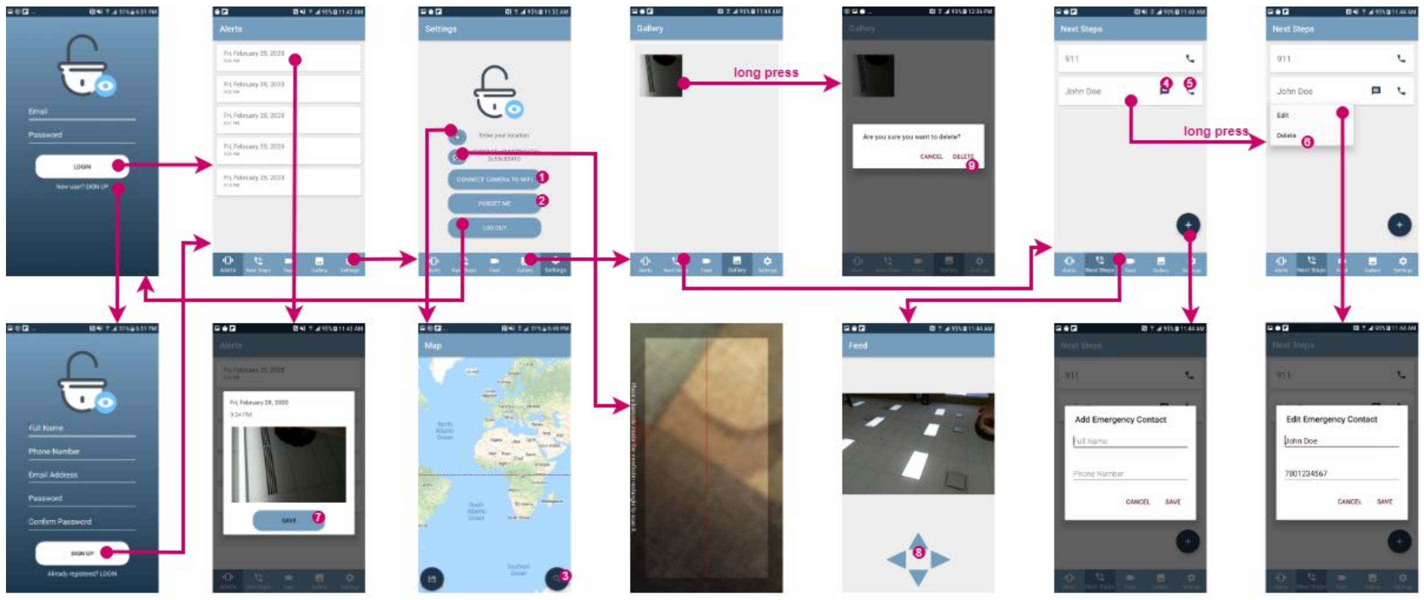
Figure 7. Firebase database UML Diagram

Android Application

We chose to create an AndroidX application using Android Studio. Androids hold 74% of the mobile device market share, so choosing an Android application is thus a good choice [6]. Our team members also have experience developing in Android Studio, making it an ideal IDE for our project.

Our application needs to elegantly and quickly meet the needs of our users, while allowing for interactions with Firebase and maintaining security. Users are able to view logs of potential break-ins, with each log associated with a push notification that uses Firebase In-App Messaging. Each log associated with a notification contains information about the break-in, including the time and an image of the face detected. Users can choose to save a notification's image into a gallery, where the image will be encrypted. Users will also be able to view a pseudo-real time video stream, which will be a series of images streamed to the user through Firebase. Users can also alter settings through the app, like adding contacts, initially setting up their camera, or altering their address. We use the Zebra Crossing Barcode Reader library to recognize and set up our camera [25]. To edit the location information, we use Places API provided by the Google Cloud Platform. As well, users have access to a feature that allows them to react quickly to a break-in through the maintenance of a contact list that they can message or call directly following a notification of a break-in.

At this stage of the project, all aforementioned features have been implemented on the application side and are ready to be integrated with the Raspberry Pi side, with the exception of the live feed feature, which is a work-in-progress and is currently able to fetch a single image from the camera. The next step for this feature is to implement the ability for the feed to update in real-time, as well as to implement an interface to control the rotation of the camera. Figure 8 shows the storyboard for our application, whose user interface has been designed to be as simple and self-explanatory as possible. This storyboard was made from actual screenshots of the application.



1. Prompts the user to connect to enter their Wi-Fi credentials to connect the Raspberry Pi to the internet.
2. Removes the user's information from the application.
3. Allows the user to search for a location.
4. Passes the contact's phone number to the phone's native call screen.
5. Passes the contact's phone number to the phone's native SMS screen.
6. Removes the contact from the list.
7. Adds the notification image to the gallery.
8. Controls the up/down and left/right rotation of the camera remotely
9. Removes the image from the gallery.

Figure 8. Mobile Application Storyboard

Sources of Reusable Design

Our codebase is split into machine learning, Raspberry Pi interfacing, Raspberry Pi to Firebase communication, and Android application development categories.

Machine Learning

Machine learning is an accurate and fast way of detecting faces. To create the image-based machine learning model we want to run video feed images against, we need a reliable library that supports machine learning with neural networks. Google's Tensorflow is widely viewed as a leading tool in machine learning and differentiates itself from other deep learning software by interfacing with several different languages, its large community base, visualization capabilities through Tensorboard, and high tunability of layers. Tensorflow has extensive documentation of its open-source code. Because of the large community, there are many examples, tutorials, and repositories available that we can reference. For example, the "Basic classification: Classify images of clothing" example from Tensorflow's tutorials page is a good starting spot for our project.

Tensorflow also runs on Raspbian and all other main operating systems, which gives us flexibility. [11]

Raspberry Pi Interfacing

There are many different options for languages for interfacing with a motion detector and a camera via the GPIO and camera slot respectively. A good option is Python, the PiCamera Python library allows us to record images and video, as well as alter some camera settings with ease. Examples like RaspberryPi.org's "Getting Started with PiCamera" example project demonstrates how to use this library. RaspberryPi.org recommends using the RPi.GPIO library, which we are picking over the more popular but now deprecated WiringPi library. The RPi.GPIO library also has extensive documentation and a few examples to kickstart work, making it the ideal library for Raspberry Pi interfacing with our motion detector [12][13].

The Python Imaging Library, which adds image processing capabilities to the Python interpreter and provides extensive file format support, will provide fast access to data stored in a few basic pixel formats. It will be used to compress images taken from the camera feed before sending them to the Android application for the pseudo-live feed to reduce the transmission rate [14].

Firebase communication

Firebase is supported in various programming languages. To save/send images or a video feed, we would push the saved images to the Firebase database from the code running on the Raspberry Pi. Firebase has easy to use SDKs that we can leverage.

Android Application Development

We are using Android Studio to make our Android application using the new AndroidX extension library, which comprises of the jetpack libraries, instead of the older Android support library, which will no longer be maintained. Luckily, Android and AndroidX are still very similar, since a main goal of AndroidX was to make naming conventions more clear, so any example code for Android is easily converted to AndroidX. Source code and community support of Android is very prevalent, which has helped in speeding up the development process. ZXing Project is used for scanning barcodes that will uniquely identify our cameras. AwesomeOpenSource.com has a list of open-source projects related to Firebase that we can refer to, which include projects focusing on messaging, UI, etc. [15].

We are also using AES-256 encryption to encrypt images that are associated with notifications if the user chooses to save them into a gallery on the Android application. AES-256 uses a 256-bit key to encrypt and decrypt the data. A 256-bit encryption key is significantly more difficult for brute-force attacks to guess than something like a 128-bit key. This was achieved by using an Android library, called Security-Crypto. This allows for easy implementation of cryptography and therefore securing the data of the user.

In the Settings page of the Android application, users are able to facilitate connection of the Raspberry Pi to Wi-Fi by entering their Wi-Fi credentials. A template from Colorlib [29] was used as the basis for making the credentials form for this feature. The Raspberry Pi collects the credentials of the Wi-Fi through the app by creating a hotspot and then proceeds to connect to the internet using the provided credentials. The hotspot will always be on so the user is always able to change their password if the password of their Wi-Fi changes. Also, the user is required to enter the code associated with the camera, this was put in place as a security measure so the neighbours of the property are not just able to login and take over the access.

Test Plan and Results

As we are not using VHDL in our projects, we are not including any testbenches.

Software Test Plan and Results

Memory Management

In Android Studio, it is possible to create a memory leak, despite using Java's garbage collector. Memory leaks will occur when objects are no longer being used by the application and the garbage collector is unable to remove them from working memory, this will be prevented by making sure that no objects are being left without a reference. The easiest way to prevent these memory leaks is by making the classes static.

If they do occur, we will use the Android Leak Canary library, which creates weak references to objects, then checks them after garbage collection is run [23].

The following is the result of running the Android LeakCanary library on our code:

```
====================================
HEAP ANALYSIS RESULT
====================================
0 APPLICATION LEAKS
References underlined with "~~~" are likely causes.
Learn more at https://squ.re/leaks.
====================================
0 LIBRARY LEAKS
Library Leaks are leaks coming from the Android Framework or Google libraries.
====================================
METADATA
Please include this in bug reports and Stack Overflow questions.
Build.VERSION.SDK_INT: 29
Build.MANUFACTURER: samsung
LeakCanary version: 2.2
App process name: ca.ualberta.dorsa.seccam
Analysis duration: 5830 ms
Heap dump file path:
/data/user/0/ca.ualberta.dorsa.seccam/files/leakcanary/2020-03-24_13-20-49_901.hprof
Heap dump timestamp: 1585077657408
====================================
```

For python code, we can use the Pympler library, which tracks memory usage during run time, and is used to recognize and identify objects that are leaking memory [24].

Peripheral Control

To test our peripheral control, we ran two simple Python scripts that used the RPPi.GPIO library and the PiCamera library to manage our motion detectors and camera. Through manual testing we have found that our motion detector and camera do function as required. Our method of reading from the GPIO pins works, as does the PiCamera library.

We did find that the PiCamera library cannot correctly filter out IR light when we need it to, so we altered the Automatic White Balance (AWB) settings to include the greyworld setting that reduces purple tinting of the image.

To test the function of the two servo motors, we assembled the gimbal to hold the motors and connected them to a power supply set to 5 V via breadboard. The PWM signals were connected to a function generator outputting a 50 Hz square wave. By manipulating the current output on the power supply, we were able to observe the motors moving causing the gimbal to move both side-to-side and up-and-down at speeds that varied with the amount of current, showing that the motors function as required.

Raspberry Pi Process Control

To ensure that the Raspberry Pi could perform all required tasks simultaneously, we ran the machine learning model, the peripheral control script, and transmitted data to Firebase concurrently. By checking the model’s expected return value (is a face), checking Firebase for streamed images, and comparing printed motion detected indicators, we ensured that the Raspberry Pi could simultaneously handle all the processes we need to complete the project.

Through this ‘concurrency’ testing process, we noted the camera cannot be used by multiple processes without mutexes, which we promptly implemented.

Machine Learning Training

To view accuracy of our machine learning models, we use Tensorflow’s toolkit Tensorboard for result data visualizations. We can view graphs of accuracy for different models, which helps us find the most successful models. View Figure 9 for an example which provides context to how machine learning testing as performed.

Android Application

Planned testing activities for the Android application included unit testing and acceptance testing.

Unit testing involved writing unit tests for functions in the Java code of the Android application using the JUnit 4 framework, focusing on those of the model and control classes. This method of testing aimed to ensure that each function performed correctly. Unit tests for the model classes and control classes have been completed. Refer to Appendix C for a sample of our unit tests.

As the Android application is in its final stages of development, acceptance tests have been performed. This involved having the members of the group as well as individuals not involved with the project use the application. Acceptance testing conducted by the members of the group evaluated the functional and non-functional requirements of the application that have been created by the group, while acceptance testing conducted by individuals not involved with the project evaluated both the usability of the application and the experience of an average user of the product.

During integration testing in Android, the relationships to the server were mocked and each component was tested individually. This allowed future expansions of the code to be done more effectively while catching any potential errors that may have been caused as a result of introducing new methods.

Hardware Test Plan

Motion Detector

For the motion detector, our test plan involved first using the datasheet to determine what we expect on motion, and comparing these expected values to our motion detectors.

According to the datasheet, we should expect the motion detector to work between three to seven meters. When motion is detected, a peak of 3.3V will be outputted, with 0 indicating no motion. Our test involved a member of our team waving at the motion detectors straight on, then staying still. This test was repeated in varying light levels.

Distance	Motion detected correctly in normal light	No motion detected correctly in normal light	Motion detected correctly in low light	No motion detected correctly in low light
1m	100%	100%	100%	100%
2m	80%	80%	100%	60%
3m	100%	60%	100%	80%
4m	60%	80%	100%	80%
5m	100%	100%	100%	100%
6m	100%	100%	80%	100%
7m	80%	100%	40%	100%
8m	60%	100%	0%	100%

Table 3. Motion Detector Tests using three motion detectors

Camera

We expect our noir camera to be able to capture images clearly in normal light, as well as Infrared (IR) images in low light. Our test results showed us that with more IR light, the image will appear more purple, and with less IR light, the image will appear more green. The camera functions as expected in terms of its ability to capture pictures, sequences of pictures, and videos of an acceptably high resolution (1280 pixels x 720 pixels).

Raspberry Pi

The Raspberry Pi had several hardware elements we needed to test to ensure that our project would function. These included the wifi functionality, the ability to connect to a monitor, a mouse, and a keyboard. We also tested select GPIO ports. Our Raspberry Pi passed all hardware tests.

Motors

We performed manual motor testing to ensure that we could move the motor in all needed directions. A table with our results are found below:

Angle	**Completed (Yes, No)**
Can move up to 90°	Yes
Can move down to 90°	Yes
Can move left to 90°	Yes
Can move right to 90°	Yes

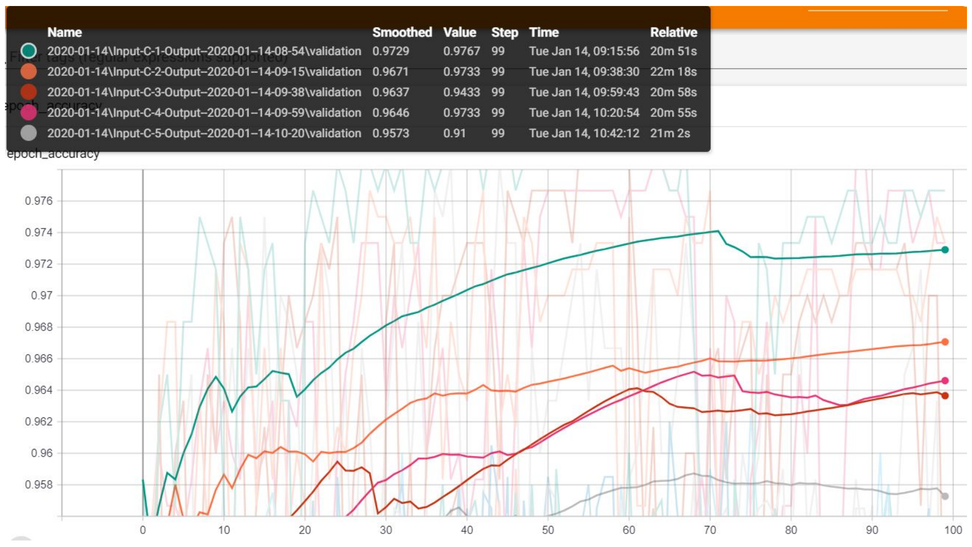
Table 4. Motion Detector Tests using three motion detectors

Results of Experiments and Characterization

Machine Learning

After training the model against random 1500 data subsets of faces and rooms without faces from a 175000 face and 175000 non face data set, our machine learning model was tested against the validation set, which consisted of random photos from the data set of face and non face images that were used to train the model. Through the analysis of validation testing results, we saw that the model achieved approximately 97% accuracy in both determining if a photo contained a face and determining if a photo did not contain a face, as an equal number of photos were used for each data set in training the model.

A machine learning tradeoff is finding a good balance between the size of the model, and the overfitting that can occur. With overfitting, if we don't have enough training data, the model will be inaccurate, but if we fit the model too much, by training it too much, then our model will again be inaccurate. Overfitting is a big concern for machine learning, where overfitting is when the model memorizes the training data set. This is typical of small data sets, where there is not enough variation and too much repetition in the training images or data. However, since Figure 9 shows that there were no major drops in accuracy for the validation data, we know that overfitting did not occur, meaning that our model will be prepared to confidently label new data images.



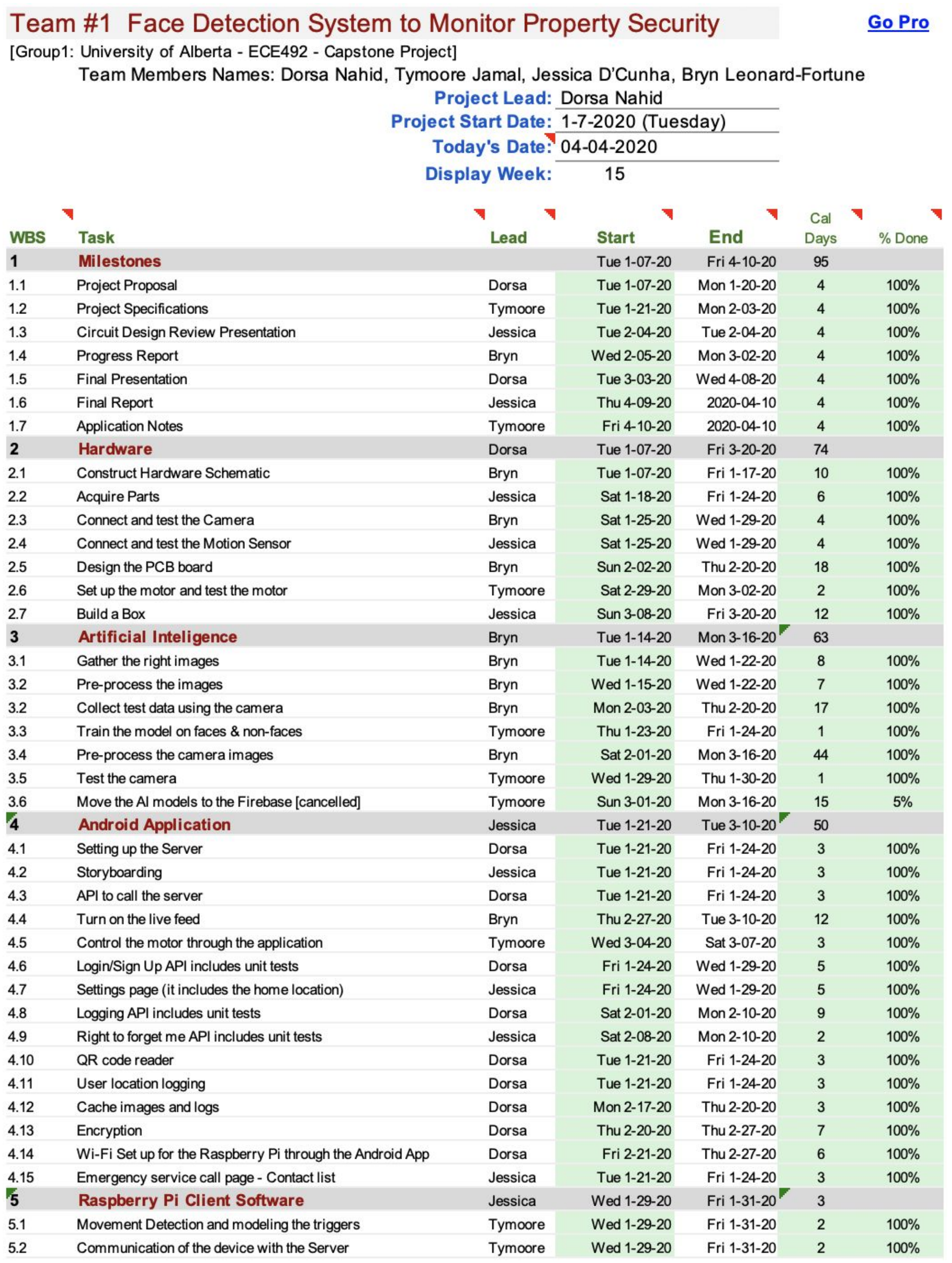
	Name	Smoothed	Value	Step	Time	Relative
●	2020-01-14\Input-C-1-Output--2020-01--14-08-54\validation	0.9729	0.9767	99	Tue Jan 14, 09:15:56	20m 51s
●	2020-01-14\Input-C-2-Output--2020-01--14-09-15\validation	0.9671	0.9733	99	Tue Jan 14, 09:38:30	22m 18s
●	2020-01-14\Input-C-3-Output--2020-01--14-09-38\validation	0.9637	0.9433	99	Tue Jan 14, 09:59:43	20m 58s
●	2020-01-14\Input-C-4-Output--2020-01--14-09-59\validation	0.9646	0.9733	99	Tue Jan 14, 10:20:54	20m 55s
●	2020-01-14\Input-C-5-Output--2020-01--14-10-20\validation	0.9573	0.91	99	Tue Jan 14, 10:42:12	21m 2s

Figure 9: Face Detection Model Validation Accuracy

Simulation of Hardware

Initially, before our ordered hardware arrived we needed to simulate the hardware functionality of our Raspberry Pi and its tasks. This was to ensure that we could test our code and continue to develop our functionality on the Raspberry Pi side. Our hardware consisted of a camera and motion detectors. To simulate them, we created a motion detector class and a camera class. We also created a mocked motion detector class that inherited from the motion detector class and overwrote the method which detected motion. Instead of actually reading the motion sensor, this mock function generated a random floating-point number between 0 and 1. If the number was greater than 0.8 then it would return that motion was detected. We settled on 0.8 as the threshold as we expect that in a house motion would not be present more than 20% of the time. To mimic the camera we required a more complex approach. We created a directory of 100 images; 50 of faces, and 50 of internal environments without faces. We had a mocked camera class that overwrote our image grabbing method and instead returned a random image from this directory. Together, mocking the motion detector and camera allowed us to continue our development before our ordered parts arrived.

Project Management



Team #1 Face Detection System to Monitor Property Security

Go Pro

[Group1: University of Alberta - ECE492 - Capstone Project]

Team Members Names: Dorsa Nahid, Tymoore Jamal, Jessica D'Cunha, Bryn Leonard-Fortune

Project Lead:	Dorsa Nahid
Project Start Date:	1-7-2020 (Tuesday)
Today's Date:	04-04-2020
Display Week:	15

WBS	Task	Lead	Start	End	Cal Days	% Done
1	**Milestones**		Tue 1-07-20	Fri 4-10-20	95	
1.1	Project Proposal	Dorsa	Tue 1-07-20	Mon 1-20-20	4	100%
1.2	Project Specifications	Tymoore	Tue 1-21-20	Mon 2-03-20	4	100%
1.3	Circuit Design Review Presentation	Jessica	Tue 2-04-20	Tue 2-04-20	4	100%
1.4	Progress Report	Bryn	Wed 2-05-20	Mon 3-02-20	4	100%
1.5	Final Presentation	Dorsa	Tue 3-03-20	Wed 4-08-20	4	100%
1.6	Final Report	Jessica	Thu 4-09-20	2020-04-10	4	100%
1.7	Application Notes	Tymoore	Fri 4-10-20	2020-04-10	4	100%
2	**Hardware**	Dorsa	Tue 1-07-20	Fri 3-20-20	74	
2.1	Construct Hardware Schematic	Bryn	Tue 1-07-20	Fri 1-17-20	10	100%
2.2	Acquire Parts	Jessica	Sat 1-18-20	Fri 1-24-20	6	100%
2.3	Connect and test the Camera	Bryn	Sat 1-25-20	Wed 1-29-20	4	100%
2.4	Connect and test the Motion Sensor	Jessica	Sat 1-25-20	Wed 1-29-20	4	100%
2.5	Design the PCB board	Bryn	Sun 2-02-20	Thu 2-20-20	18	100%
2.6	Set up the motor and test the motor	Tymoore	Sat 2-29-20	Mon 3-02-20	2	100%
2.7	Build a Box	Jessica	Sun 3-08-20	Fri 3-20-20	12	100%
3	**Artificial Inteligence**	Bryn	Tue 1-14-20	Mon 3-16-20	63	
3.1	Gather the right images	Bryn	Tue 1-14-20	Wed 1-22-20	8	100%
3.2	Pre-process the images	Bryn	Wed 1-15-20	Wed 1-22-20	7	100%
3.2	Collect test data using the camera	Bryn	Mon 2-03-20	Thu 2-20-20	17	100%
3.3	Train the model on faces & non-faces	Tymoore	Thu 1-23-20	Fri 1-24-20	1	100%
3.4	Pre-process the camera images	Bryn	Sat 2-01-20	Mon 3-16-20	44	100%
3.5	Test the camera	Tymoore	Wed 1-29-20	Thu 1-30-20	1	100%
3.6	Move the AI models to the Firebase [cancelled]	Tymoore	Sun 3-01-20	Mon 3-16-20	15	5%
4	**Android Application**	Jessica	Tue 1-21-20	Tue 3-10-20	50	
4.1	Setting up the Server	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%
4.2	Storyboarding	Jessica	Tue 1-21-20	Fri 1-24-20	3	100%
4.3	API to call the server	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%
4.4	Turn on the live feed	Bryn	Thu 2-27-20	Tue 3-10-20	12	100%
4.5	Control the motor through the application	Tymoore	Wed 3-04-20	Sat 3-07-20	3	100%
4.6	Login/Sign Up API includes unit tests	Dorsa	Fri 1-24-20	Wed 1-29-20	5	100%
4.7	Settings page (it includes the home location)	Jessica	Fri 1-24-20	Wed 1-29-20	5	100%
4.8	Logging API includes unit tests	Dorsa	Sat 2-01-20	Mon 2-10-20	9	100%
4.9	Right to forget me API includes unit tests	Jessica	Sat 2-08-20	Mon 2-10-20	2	100%
4.10	QR code reader	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%
4.11	User location logging	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%
4.12	Cache images and logs	Dorsa	Mon 2-17-20	Thu 2-20-20	3	100%
4.13	Encryption	Dorsa	Thu 2-20-20	Thu 2-27-20	7	100%
4.14	Wi-Fi Set up for the Raspberry Pi through the Android App	Dorsa	Fri 2-21-20	Thu 2-27-20	6	100%
4.15	Emergency service call page - Contact list	Jessica	Tue 1-21-20	Fri 1-24-20	3	100%
5	**Raspberry Pi Client Software**	Jessica	Wed 1-29-20	Fri 1-31-20	3	
5.1	Movement Detection and modeling the triggers	Tymoore	Wed 1-29-20	Fri 1-31-20	2	100%
5.2	Communication of the device with the Server	Tymoore	Wed 1-29-20	Fri 1-31-20	2	100%

Figure 10: Schedule (see Appendix A for full Gantt chart, Appendix B for old chart)

Safety

Team Safety

Safety is a value of our team as we are working in hardware labs with electrical and mechanical equipment as designers, experimenters, engineers, testers, and operators. Our risk and safety procedures can be divided into subcategories: mechanical, electrical, software, miscellaneous.

Electrical

Our project involved connecting various electrical components to each other. Electrical shocks, scrapes or punctures or pinches from sharp wires, and burns from soldering or hot components are all dangers that we had the potential to face. Our team strived to always work with another team member, use equipment properly, note and know how to use emergency equipment (first aid kit, cell phones with emergency phone numbers, AED), and be careful when applying voltage to boards and circuits in our project. This project also involved creating a PCB which was done in the ELKO Engineering Garage. Our team complied with the rules of the garage to ensure that safe procedures and practices were followed when accomplishing this task.

Software

There aren't physical risks to using software, but ensuring that we do not load malware onto any University machines as well as complying with University ITS policies are precautions that we will keep in the back of our minds. We also tried not to hog or waste university resources.

Product Safety

Our design aims to avert exposing users to any risk, and any possible burns or shocks users could experience from our circuitry. The maximum voltage of the Raspberry Pi is 5.0V, and has a maximum Power SupplU current of 3.0A, making the maximum power 15W [7]. This is not a lot of power and cannot cause a lot of damage as our hardware is separate from sensitive systems. The Raspberry Pi's operational temperature ranges from 40°C to 85°C, with the Noir Camera operating between -20°C and 60°C, and the motion detector operational between -15°C and 70°C degrees [7][8][9]. If our product should fail, there would be no physical consequences. On failure, the software would stop working.

Regulatory and Society

Regulations

Our security system involves analyzing, transmitting, and storing sensitive information. This includes images of the inside of users' homes and personal information such as users' location, property location, name, and contact information, as well as the contact information of users' emergency contacts. As this is all sensitive material, society as a whole is fairly scrutinous and critical of the use of such material, and regulations exist for handling it.

In Canada, the regulations that affect how we must handle, use, and explain the information collected through our security system includes laws for businesses and individuals, as it relates to security cameras and personal identifying information. For home users who capture no image data other than their own property, there is no jurisdiction or regulation. Regulations apply to commercial use of cameras (for example, in a corporate building or a home with tenants), and any filming of other property owners (for example in the case where one films their backyard and ends up filming a small section of another's property as well).

These two uses of security systems are regulated on a municipal, provincial and federal level. The vast majority of municipalities in Canada do not have regulations. However, if these cameras were placed in select municipalities with existing regulation, like Hamilton, Ontario, then those bylaws would apply as well [16]. For example, Alberta's provincial legislation would apply in these two cases, and because we are storing and transmitting personal information using Firebase, whose physical servers are not within Alberta, federal legislation will also apply.

The Office of the Privacy Commissioner of Canada (OPC) handles all regulations related to security cameras in Canada [17]. OPC requires that corporations who use security cameras balance the need for security and the need for privacy, and that they are clear (except in the case of covert security) about how they will use collected images, why they are collecting images, and who is collecting images. The OPC manages complaints and papers related to Canada's Privacy Act, PIPEDA; the Personal Information Protection and Electronic Documents Act which applies to Canada, and PIPA; the Personal Information Protection Act, which is Alberta's specific regulatory act regarding commercial use of personal information [18][19][20].

Under these acts, names, addresses, and images of people (which fall under "any identifying number, symbol or other particular assigned to the individual") are all sensitive information. Thus, our system must secure the collection and use of such information, and corporations that use our product must secure and justify the use of it as well. The images should also remain in our database, in case the people filmed request their own personal information (their own identifying images of their faces).

Risks

Some risks that we need to be aware of include hacking of our Raspberry Pi, database attacks, and using or storing people's personal information (contact information, name, images) in a way that doesn't comply with regulations. All of these risks could result in legal consequences for our face detection security.

Mitigation

To mitigate the risk of using or storing personal information in a non-complicit way to current regulations, management for our face detection security needs to keep apprised of current laws. To mitigate the risk of hacking, instead of implementing authentication ourselves, we chose to use Firebase's authentication service. To mitigate the risk of intercepting sensitive information, we encrypted our transmitted data.

Environmental Impact

The Restriction of Hazardous Substances Directive (RoHS) restricts the use of certain hazardous materials such as lead, cadmium, and mercury in the manufacture of electrical/electronic equipment. Examples of components that may contain such substances are solder, PCB finishes, and paints [26].

Our project included a custom PCB that needed to have components soldered onto it. In addition, the Raspberry Pi 4 already has solder on it. However, all commercial PCBs are required to have less than 1000 ppm of lead in them [26], and all Raspberry Pi products have undergone compliance testing to ensure they are RoHS compliant [27]. The custom PCB design has far fewer components than the Raspberry Pi. Thus, if the board is properly soldered (i.e. excessive amount of solder is not used), it is expected to also be compliant. The type of finish used on the custom PCB will also affect RoHS compliance; the board is expected to be constructed in the ELKO Engineering Garage, which offers copper blanks that do not have a non-compliant finish.

In addition to the physical components that make up our project, our cloud based database run through firebase's cloud platform also has an impact on the environment. Data centers worldwide use 416 terawatts of energy per year, which is 3% of the total global energy production. This is a huge amount of energy, and considering that most electricity is not produced cleanly (produces greenhouse gasses), data centers globally (which include our firebase servers) do contribute to negatively impacting the atmosphere and environment.

Sustainability

Our project has three main components which consume power: the android mobile app, the firebase server, and our Raspberry Pi, which includes all of the components attached to it.

We can’t accurately predict the power consumption of our total project, since this is dependent on how much each user uses the app, as well as the amount of data they wish to keep in our database. Since we cannot determine the power consumption of the app or the firebase server, we will only cover the sustainability of the Raspberry Pi and its peripherals.

Below, we calculate the power consumption on standby and max for when each component is being run fully or standing by. To calculate the typical power consumption over the lifetime of the face detection system, we consider that motors and warning LEDs will be used less than one percent (here, we round up to one percent). We assume that motion will be detected by all three motion detectors 5% of the time, and we approximate that each motion detector will detect motion around 8% of the time.

The Raspberry Pi draws 2.5A at 5V with regular power supply cables. See Table 5 in the appendix for power consumption.

	Raspberry Pi (polling + wifi) [33]	Shooting Video [32]	LEDs	Motion Detectors [30]	Motors [31]	Total power
Standby State	575mA*5V = 2.875W	0	0	50uA*5V*3 =750mW	5v*10mA*2 =0.1W	=2.92W
Max	1200mA*5V =6W	250mA*5V = 1.25W	3.3V*9mA *5 LEDs = 0.149W	5V*65mA*3 =0.0975W	5V*250mA*2 =2.5W	=8.68W
Best Guess	95% standby +5% max = 0.95*(2.88W) +0.05*(6W) =3.03W	95% standby +5% max =0.95*(0)+0.05 *(1.25W) =0.0625W	99% standby + 1% max =0.015W	92% standby + 8% max =0.07869W	=99% standby + 1%max =0.124W	=**4.02W**

Table 5. Power Consumption of Raspberry Pi and Peripherals

Here, we see that for an hour, our Raspberry Pi and its peripherals will use 4.02W. In a day, this translates to 96.48W. Since our device will be constantly running, this means that in a year, our Raspberry Pi and its peripherals will consume 35kW.

By using Canada's greenhouse gas emissions data, we can see that since Canada emitted 722 mega tonnes of CO2 in 2017 and 10.9% came from electrical energy production, that 78.7 mega tonnes of CO2 was produced to create Canada's Electrical energy [34]. Since Canada's electricity generation was 80.6% non-emitting, this means that of the 652 375 GWh produced, 652 375 GWh*(100% - 80.6%) = 126,560.75GWh emit CO2 [35]. This translates to 1,108,672,170GW in a year. So 78.7 mega tonnes of CO2 was a byproduct of the production of 1,108,672,170GWh, which means that a watt creates 0.00000000065122 tonnes of CO2. If this is the case, then a watthour produces 0.65mg of greenhouse gas emission. In a year, our project creates 22.9 grams of CO2 gas per year, per product.

If we were looking for a cleaner solution, we could look to solar panels. A solar panel generates 15Wh/square foot on average [36]. Since we only need to generate 4.02 Watts, a solar panel of 0.268 square feet would work nicely to power a single Raspberry Pi and its peripherals.

Quick Start Manual

To use the Face Detection Security Camera System, a user needs to perform the following steps:

1. Plug in the camera.
2. Install the Android application onto an Android phone and create an account. (see Figure 11, screen 2).
3. When signed in to the application, navigate to the settings screen and click on the camera icon (see Figure 11, screen 5) to open the phone's camera as a code scanner (see Figure 11, screen 7). Scan the camera's QR code to bind the camera with the user's account.
4. Connect the user's phone to the Wi-Fi hotspot called "iSecurity."
5. Navigate again to the settings screen and click on the "Connect Camera to Wi-Fi" button (see Figure 11, screen 5). Press "OK."
6. On the resulting screen, enter the details in the fields (see Figure 11, screen 15). Enter the credentials for your home Wi-Fi to allow the camera to be connected to the internet and transmit data and notifications to the user when an intruder is present on the user's property.
7. Use the Android application as follows:
    - On the "Alerts" screen , any intruder alerts will be logged (see Figure 11, screen 3). Click on a logged alert to view its associated image in a pop-up dialog (see Figure 11, screen 4). Click "Save" on this dialog to save the encrypted image into the Gallery.
    - On the "Next Steps" screen, emergency contacts can be maintained. Click on the "+" button to add a new contact (see Figure 11, screen 11). Long-click on a contact other than 911 to choose to edit or delete it (see Figure 11, screen 10 and 12). Click on the message and phone icons beside a contact to send an SMS or call the contact, respectively (see Figure 11, screen 9).
    - On the "Feed" screen, the user is able to watch a live feed of the events that are occurring on their property. Click on the up, down, left, and right buttons to move the camera to the desired position and watch the live feed (see Figure 11, screen 8).
    - On the "Gallery" screen (see Figure 11, screen 13), the user is able to view the decrypted images (see Figure 11, screen 16) that they had

wished to store for viewing later. Long-click on an image to bring up the option to delete it (see Figure 11, screen 14).

- On the “Settings” screen, the user is able to maintain various user settings (see Figure 11, screen 5).
    1. Click on the compass icon to bring up a Map screen (see Figure 11, screen 6) where the user can search for a desired location by clicking the magnifying glass icon and saving the location by clicking the disk button. Click on the camera icon to rescan a camera’s QR code.
    2. Click on the “Connect Camera to Wi-Fi” button to reconnect the camera to Wi-Fi.
    3. Click on the “Forget Me” button to delete all user data from the application.
    4. Click on the “Logout” button to log the user out of the application.

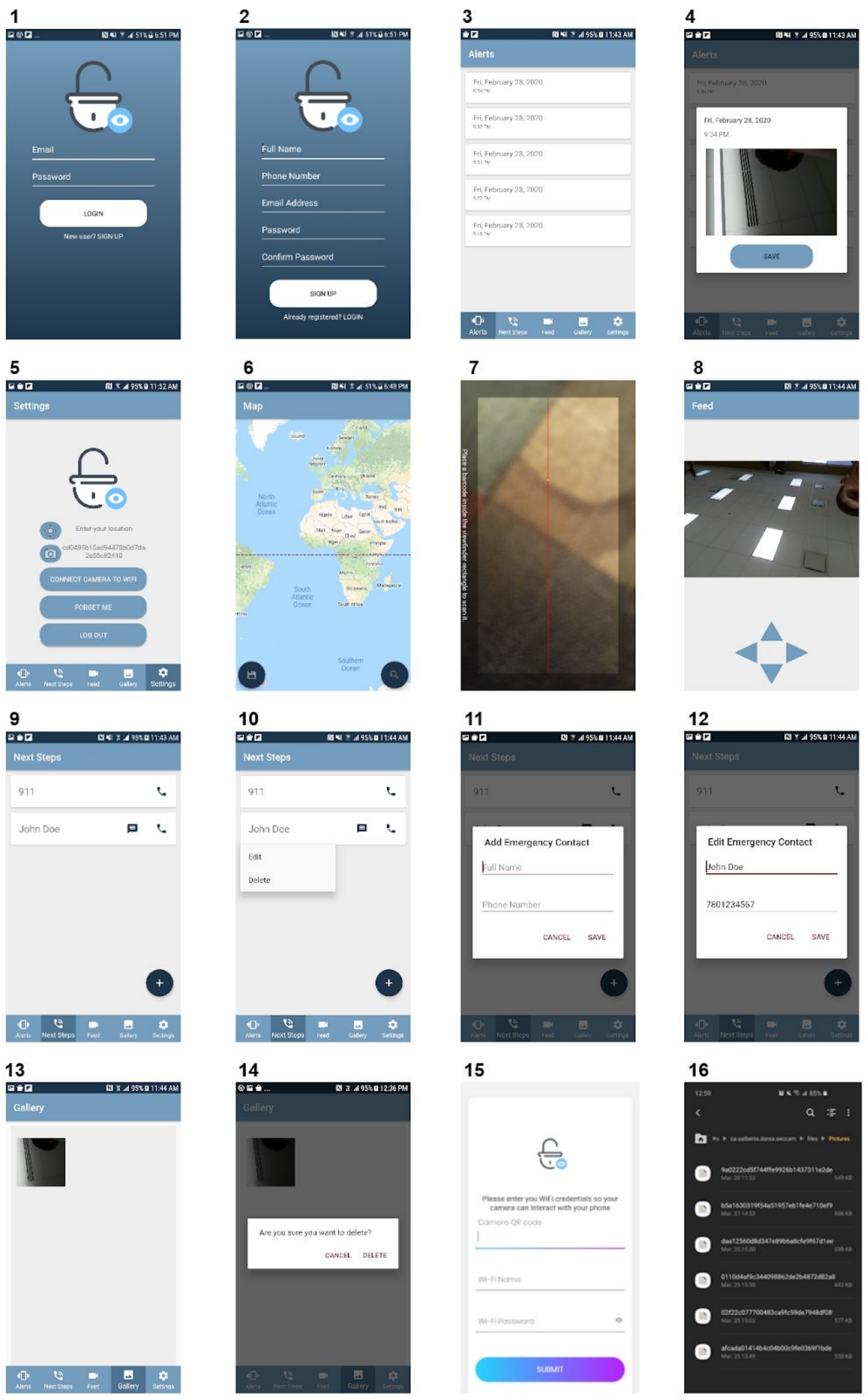


Figure 11: Android Application Screens

Conclusions

Our project goals involve meeting a home owner’s use need for accurate and fast security protection, ease of use, and privacy. To create a feasible solution for the home owner’s needs, our project requires it to be built with safe and easily sourceable materials with low effects on the environment, and developed with portable and maintainable code that can be easily expanded in the future.

The basic tasks of our project involved the ability to detect faces, inform users of break ins, allow users control over their camera, and store face detection and user information. With our requirements in mind, we chose to create a system to record images, train and use a machine learning face detection model, create a user-controllable application, and leverage an online server with security, database, and notification services. Our final project design uses motion detectors and a raspberry pi camera connected to a raspberry pi that controls the peripherals, creates a hotspot to connect to wifi with our android application, and manages communication between the raspberry pi and our Firebase Server. Our firebase server liaises between the raspberry pi and the user application while also acting as a real-time database, secure login manager, and notifications service. Our android application uses encrypted file storage as well as a connection to firebase to store private information such as photos of their homes or burglars, display a live feed, control the orientation of the camera, link the user to their camera, connect the raspberry pi to wifi, and manage user contacts to act in case of an emergency.

Overall, we were able to meet most of the requirements of the user, and all of the requirements to build and develop the project. The user experiences a very secure and private, highly accurate, and relatively fast security system.

The system is secure; 100% of our communication with Firebase is AES-256 encrypted through Firebase and all images are AES-256 encrypted through the Android Application. The National Security Agency uses this encryption to encrypt their documents up to the Top Secret classification.

The system is accurate; our final model had an accuracy of 97.7% according to our Tensorflow tests, which means that when a face is present in the 0.5 - 7m range of our security system, our model will be able to determine a face is present.

The system is relatively fast; our speed goals involves creating a seamless environment for a user and having users informed in real time of an intrusion. User experience through the application was nearly instantaneous and thus acceptable, except for the Raspberry Pi's ability to send and receive frames, affecting the live feed and camera orientation control through the application; instead of reaching 5 frames per second, the Raspberry Pi can send or receive about 1 frame per second. This is due to Firebase acting as a bottleneck from our lower tier account, which slows down communication entering and exiting our server to the Raspberry Pi.

All of our hardware materials were acceptably safe to acquire and use, with no negative effects other than the 35kW and 22.9 grams of carbon per year. With solar panels, we can use 0.268 square feet and reduce the carbon footprint to near zero.

Our software is made from portable and popular technologies, including Android Studio, Tensorflow, and Python. Our group members followed most coding conventions in Python, and used a combination of testing and adherence to Firebase coding conventions in Java. Our code has been tested experimentally and with unit tests, as well as for memory management using the Android Leak Canary and Pympler libraries we found 0 memory leaks. Our code also passes all of our JUnit unit tests.

Having had success in meeting the vast majority of the user, builder, and developer needs of our project, we look forward to exploring future expansions; our next steps involve completing the PCB board design for our peripherals, creating more CNN models and moving them to a higher-tiered Firebase for increased accuracy and speed, and creating a 3D printed case for our project.

Future Work

This project can be expanded in the future by creating a customized PCB board. At the current moment a breadboard is used to create the connections however to move into production it would be more reliable and cost effective to move towards creating a custom PCB board.

The set up is protected inside a temporary enclosure which is not stable nor feasible if the project goes to production. In order to make the project a feasible one for production, an enclosure needs to be created. This can be done by 3D printing a case and placing the set up inside.

One way to increase the security of the authentication of the Android application would be to have the password and the data double encrypted. RIght now the FireBase Authentication package does AES encryption, however the application can also encrypt the data prior to sending it to the cloud for firebase to encrypt. One other way to increase the security of the application is to block the feature that allows the user to take screenshots since that would defeat the purpose of being able to save the picture into a gallery encrypted.

This project can be expanded in terms of improving the machine learning in the face detection component. It can be improved using the following two methods, increasing the number of images used in training the machine learning model and moving the execution of the machine learning model to the cloud.In our final state we had the machine learning model be trained with a total of 3000 images, this provided a high accuracy, however it could be improved upon by increasing the total number of images used in training. Through the increase of training images we could reduce the number of false positives and false negatives produced by our machine learning model. Moving the machine learning model to the cloud instead of running it on the Raspberry Pi would also be an advantage once implemented. Through having the machine learning model in the cloud, we could reduce the workload of the Raspberry Pi, leading to overall higher performance.

Another improvement which could be made in this project is error detection and reporting to the user if the Raspberry Pi were to fail. This could be done through having several LEDs which turn on when the Raspberry Pi is running as intended, but might turn off if something goes wrong. We could potentially include the following LEDs:

power, Wi-Fi, script running, database available, and phone connection available. These would provide the user with a better understanding if the application is working correctly or if an issue is present.

Source Code

Github Organization (https://github.com/FDSMPS)

- Android Application:
  https://github.com/FDSMPS/FDSMPS-android/tree/final
- Raspberry Pi Client Application:
  https://github.com/FDSMPS/Security-Camera-App/tree/final
- Image classification training:
  https://github.com/FDSMPS/Image-Classification-Training/tree/final
- Wi-Fi set up for Raspberry Pi
  https://github.com/FDSMPS/InitialConfigWifiSetUp/tree/final

Our source code is original, except for where we have taken advantage of source code that we modified and described in our Sources of Reusable Design Section. Our third party libraries have remained unmodified.

Peer Review

Group number: 1

Group member name: Dorsa Nahid
Member's contribution: 25%
Comments: Dorsa worked heavily on getting the Android application functional, encryption, sending push notifications, setting up the Wi-Fi for the Raspberry Pi, gallery, retrieving notifications and displaying them.

Group member name: Jessica D'Cunha
Member's contribution: 25%
Comments: Jessica did a fantastic job with designing the Android application, writing unit tests, and creating the contact list portion of the application. Her contact list looks great and is very OOP. She also worked heavily on designing and manufacturing a PCB, although a final good copy was not able to be made due to the circumstances surrounding COVID-19.

Group member name: Hannah Leonard Fortune
Member's contribution: 25%
Comments: Bryn has done a great job of setting up the model images and getting the PCB working, she has taken on some Android tasks as well and I cannot wait for her to get more involved with Android.

Group member name: Tymoore Jamal
Member's contribution: 25%
Comment: Tymoore has done a great job of setting up the AI model, training it, setting up the motor and camera, and it all works great.

References

[1] Lu, J., & Plataniotis, K. (2009, June). ResearchGate. Retrieved January 31, 2020, from https://www.researchgate.net/publication/232627417_On_conversion_from_color_to_gray-scale_images_for_face_detection

[2] Lim, W. H., & Isa, N. A. M. (2012, August). ResearchGate. Retrieved January 31, 2020, from https://www.researchgate.net/publication/259293860_A_novel_adaptive_color_to_grayscale_conversion_algorithm_for_digital_images

[3] Image Segmentation in Deep Learning: Methods and Applications. (n.d.). Retrieved January 31, 2020, from https://missinglink.ai/guides/computer-vision/image-segmentation-deep-learning-methods-applications/

[4] Shin, M., Kim, M., & Kwon, D.-S. (2016, November 17). IEEE Xplore. Retrieved January 31, 2020, from https://ieeexplore.ieee.org/abstract/document/7745199

[5] Dongare, A. S., Alvi, A. S., & Tarbani, N. M. (2017). An Efficient Technique for Image Encryption and Decryption for Secured Multimedia Application . International Research Journal of Engineering and Technology, 4(4). Retrieved from https://pdfs.semanticscholar.org/6c09/ae0a58c7577c66f5a183f4c6f97712a69d08.pdf

[6] Mobile Operating System Market Share Worldwide. (n.d.). Retrieved January 31, 2020, from https://gs.statcounter.com/os-market-share/mobile/worldwide

[7] FAQs - Raspberry Pi Documentation. (n.d.). Retrieved January 31, 2020, from https://www.raspberrypi.org/documentation/faqs/

[8] Raspberry Pi PiNoir Camera Module V2 - Seeed Studio. (n.d.). Retrieved January 31, 2020, from https://www.seeedstudio.com/Raspberry-Pi-PiNoir-Camera-Module-V2-p-4166.html

[9] HC-SR501 PIR MOTION DETECTOR. (n.d.). Retrieved January 31, 2020, from https://www.mpja.com/download/31227sc.pdf

[10] Raspberry Pi hardware - Raspberry Pi Documentation. (n.d.). Retrieved January 31, 2020, from https://www.raspberrypi.org/documentation/hardware/raspberrypi/

[11] Basic classification: Classify images of clothing : TensorFlow Core. (n.d.). Retrieved January 17, 2020, from https://www.tensorflow.org/tutorials/keras/classification

[12] 13. API - Input Devices. (n.d.). Retrieved January 17, 2020, from https://gpiozero.readthedocs.io/en/stable/api_input.html

[13] (n.d.). Retrieved January 17, 2020, from https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/

[14] Pillow (PIL Fork) Overview. (n.d.). Retrieved January 17, 2020, from https://pillow.readthedocs.io/en/5.1.x/handbook/overview.html

[15] (n.d.). Retrieved January 17, 2020, from https://projects.raspberrypi.org/en/projects/getting-started-with-picamera/

[16] Dongen, Matthew Van. “Hamilton to Keep Bylaw Ban on Security Cameras Facing the Street.” TheSpec.com, 17 Jan. 2019, www.thespec.com/news-story/9131134-hamilton-to-keep-bylaw-ban-on-security-cameras-facing-the-street/.

[17] Office of the Privacy Commissioner of Canada. “About the OPC.” Office of the Privacy Commissioner of Canada, 26 Sept. 2019, www.priv.gc.ca/en/about-the-opc/.

[18] Office of the Privacy Commissioner of Canada. “Consultation on the OPC's Proposals for Ensuring Appropriate Regulation of Artificial Intelligence.” Office of the Privacy Commissioner of Canada, 28 Jan. 2020, www.priv.gc.ca/en/about-the-opc/what-we-do/consultations/consultation-ai/pos_ai_202001/.

[19] Office of the Privacy Commissioner of Canada. “The Personal Information Protection and Electronic Documents Act (PIPEDA).” *Office of the Privacy Commissioner of Canada*, 4 Sept. 2019, www.priv.gc.ca/en/privacy-topics/privacy-laws-in-canada/the-personal-information-protection-and-electronic-documents-act-pipeda/.

[20] “Personal Information Protection Act.” *Alberta.ca*, www.alberta.ca/personal-information-protection-act.aspx.

[21] (n.d.).Retrieved January 16, 2020, from http://henrysbench.capnfatz.com/henrys-bench/arduino-sensors-and-input/arduino-hc-sr501-motion-sensor-tutorial/

[22](n.d.).Retrieved January 16, 2020, from https://www.petervis.com/Raspberry_PI/Raspberry_Pi_CSI/raspberry-pi-csi-interface-connector-pinout.html

[23] Square. “Square/Leakcanary.” *GitHub*, 30 Jan. 2020, github.com/square/leakcanary.

[24] “Identifying Memory Leaks.” *Identifying Memory Leaks - Pympler 0.5 Documentation*, pythonhosted.org/Pympler/muppy.html.

[25] Zxing. “Zxing/Zxing.” *GitHub*, 1 Feb. 2020, github.com/zxing/zxing.

[26] Restriction of Hazardous Substances Directive. (n.d.). Retrieved February 28, 2020, from https://en.wikipedia.org/wiki/Restriction_of_Hazardous_Substances_Directive

[27] Product compliance and safety. (n.d.). Retrieved February 28, 2020, from https://www.raspberrypi.org/documentation/hardware/raspberrypi/conformity.md

[28] Power Consumption Benchmarks. (n.d.). Retrieved February 28, 2020, from https://www.pidramble.com/wiki/benchmarks/power-consumption

[29] Aigars. (2018, April 21). Login Form v2 by Colorlib. Retrieved February 28, 2020, from https://colorlib.com/wp/template/login-form-v2/

[30] Components 101 (2017, September 18). HC-SR501 PIR Sensor Retrieved March 1, 2020, from https://components101.com/hc-sr501-pir-sensor

[31] ProtoSupplies. (Unknown). Servo Motor Micro SG90. Retrieved March 1, 2020, from https://protosupplies.com/product/servo-motor-micro-sg90/

[32] Raspberry Pi (Unknown). Raspberry Pi Power Supply. Retrieved March 1, 2020, from
https://www.raspberrypi.org/documentation/hardware/raspberrypi/power/README.md

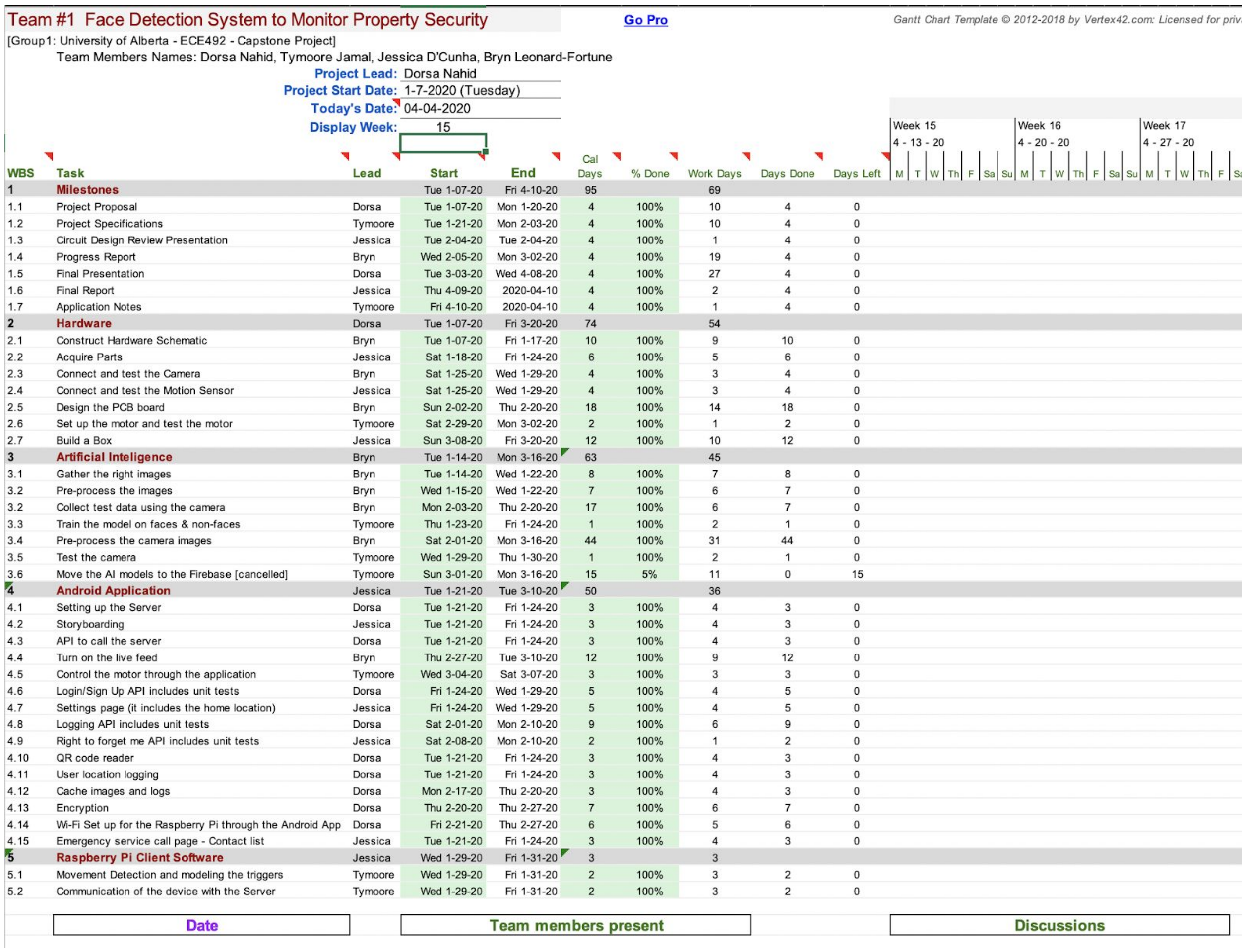
[33] Raspberry Pi. (Unknown). Frequently Asked Questions. Retrieved March 1, 2020, from https://www.raspberrypi.org/documentation/faqs/

[34] Government of Canada. (2019, August 28). Canada's Renewable Power Landscape 2017 – Energy Market Analysis. Retrieved March 1, 2020, from
https://www.cer-rec.gc.ca/nrg/sttstc/lctrct/rprt/2017cndrnwblpwr/ghgmssn-eng.html

[35] Government of Canada. (2019, August 28). Canada's Renewable Power Landscape 2017 – Energy Market Analysis. Retrieved March 1, 2020, from

https://www.cer-rec.gc.ca/nrg/sttstc/lctrct/rprt/2017cndrnwblpwr/cndnvrvw-eng.html
[36] Zientara, Ben. (2019 January 1). How much electricity does a solar panel produce?
https://www.solarpowerrocks.com/solar-basics/how-much-electricity-does-a-solar-panel-produce/

Appendix A: Gantt Chart - New Schedule



Team #1 Face Detection System to Monitor Property Security

Go Pro

Gantt Chart Template © 2012-2018 by Vertex42.com: Licensed for priv

[Group1: University of Alberta - ECE492 - Capstone Project]

Team Members Names: Dorsa Nahid, Tymoore Jamal, Jessica D'Cunha, Bryn Leonard-Fortune

Project Lead: Dorsa Nahid

Project Start Date: 1-7-2020 (Tuesday)

Today's Date: 04-04-2020

Display Week: 15

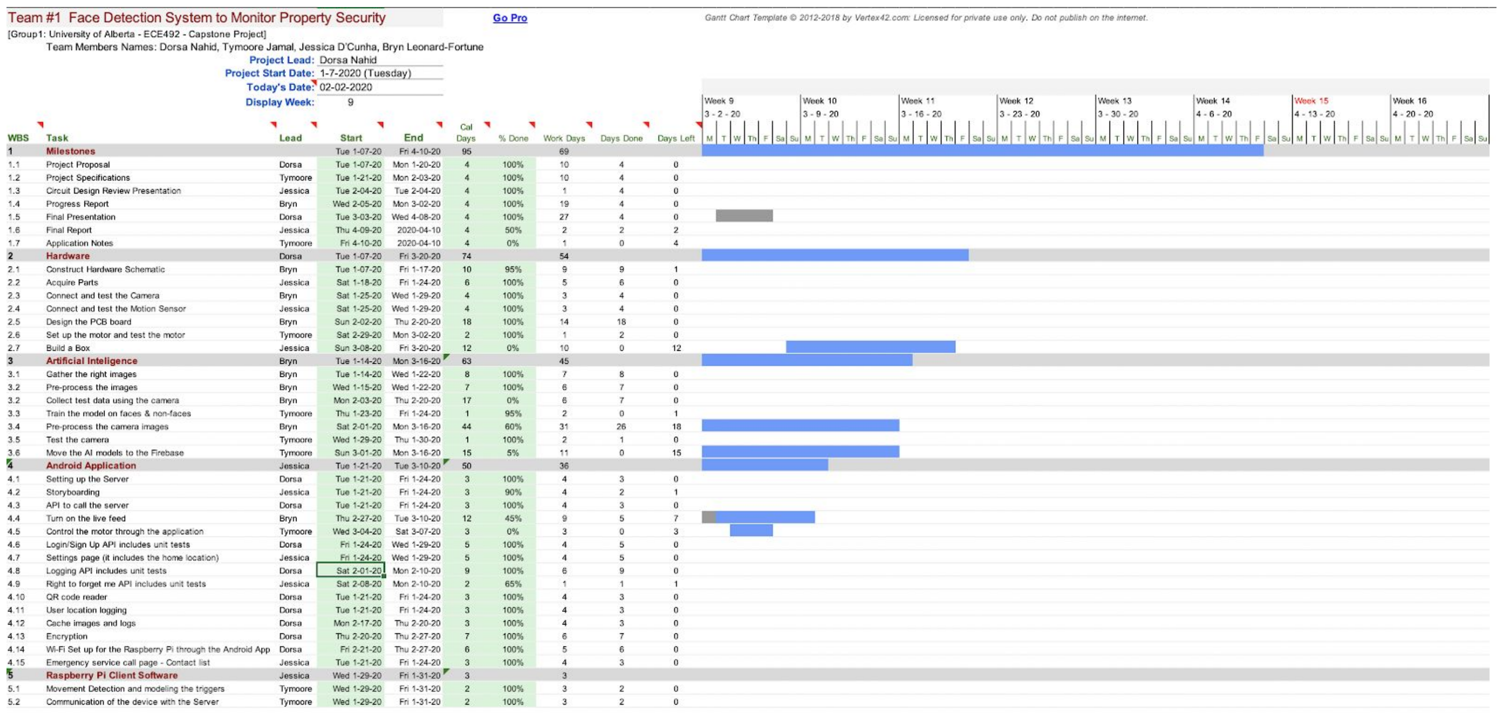
Week 15 4 - 13 - 20 | Week 16 4 - 20 - 20 | Week 17 4 - 27 - 20

WBS	Task	Lead	Start	End	Cal Days	% Done	Work Days	Days Done	Days Left
1	**Milestones**		Tue 1-07-20	Fri 4-10-20	95		69		
1.1	Project Proposal	Dorsa	Tue 1-07-20	Mon 1-20-20	4	100%	10	4	0
1.2	Project Specifications	Tymoore	Tue 1-21-20	Mon 2-03-20	4	100%	10	4	0
1.3	Circuit Design Review Presentation	Jessica	Tue 2-04-20	Tue 2-04-20	4	100%	1	4	0
1.4	Progress Report	Bryn	Wed 2-05-20	Mon 3-02-20	4	100%	19	4	0
1.5	Final Presentation	Dorsa	Tue 3-03-20	Wed 4-08-20	4	100%	27	4	0
1.6	Final Report	Jessica	Thu 4-09-20	2020-04-10	4	100%	2	4	0
1.7	Application Notes	Tymoore	Fri 4-10-20	2020-04-10	4	100%	1	4	0
2	**Hardware**	Dorsa	Tue 1-07-20	Fri 3-20-20	74		54		
2.1	Construct Hardware Schematic	Bryn	Tue 1-07-20	Fri 1-17-20	10	100%	9	10	0
2.2	Acquire Parts	Jessica	Sat 1-18-20	Fri 1-24-20	6	100%	5	6	0
2.3	Connect and test the Camera	Bryn	Sat 1-25-20	Wed 1-29-20	4	100%	3	4	0
2.4	Connect and test the Motion Sensor	Jessica	Sat 1-25-20	Wed 1-29-20	4	100%	3	4	0
2.5	Design the PCB board	Bryn	Sun 2-02-20	Thu 2-20-20	18	100%	14	18	0
2.6	Set up the motor and test the motor	Tymoore	Sat 2-29-20	Mon 3-02-20	2	100%	1	2	0
2.7	Build a Box	Jessica	Sun 3-08-20	Fri 3-20-20	12	100%	10	12	0
3	**Artificial Inteligence**	Bryn	Tue 1-14-20	Mon 3-16-20	63		45		
3.1	Gather the right images	Bryn	Tue 1-14-20	Wed 1-22-20	8	100%	7	8	0
3.2	Pre-process the images	Bryn	Wed 1-15-20	Wed 1-22-20	7	100%	6	7	0
3.2	Collect test data using the camera	Bryn	Mon 2-03-20	Thu 2-20-20	17	100%	6	7	0
3.3	Train the model on faces & non-faces	Tymoore	Thu 1-23-20	Fri 1-24-20	1	100%	2	1	0
3.4	Pre-process the camera images	Bryn	Sat 2-01-20	Mon 3-16-20	44	100%	31	44	0
3.5	Test the camera	Tymoore	Wed 1-29-20	Thu 1-30-20	1	100%	2	1	0
3.6	Move the AI models to the Firebase [cancelled]	Tymoore	Sun 3-01-20	Mon 3-16-20	15	5%	11	0	15
4	**Android Application**	Jessica	Tue 1-21-20	Tue 3-10-20	50		36		
4.1	Setting up the Server	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
4.2	Storyboarding	Jessica	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
4.3	API to call the server	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
4.4	Turn on the live feed	Bryn	Thu 2-27-20	Tue 3-10-20	12	100%	9	12	0
4.5	Control the motor through the application	Tymoore	Wed 3-04-20	Sat 3-07-20	3	100%	3	3	0
4.6	Login/Sign Up API includes unit tests	Dorsa	Fri 1-24-20	Wed 1-29-20	5	100%	4	5	0
4.7	Settings page (it includes the home location)	Jessica	Fri 1-24-20	Wed 1-29-20	5	100%	4	5	0
4.8	Logging API includes unit tests	Dorsa	Sat 2-01-20	Mon 2-10-20	9	100%	6	9	0
4.9	Right to forget me API includes unit tests	Jessica	Sat 2-08-20	Mon 2-10-20	2	100%	1	2	0
4.10	QR code reader	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
4.11	User location logging	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
4.12	Cache images and logs	Dorsa	Mon 2-17-20	Thu 2-20-20	3	100%	4	3	0
4.13	Encryption	Dorsa	Thu 2-20-20	Thu 2-27-20	7	100%	6	7	0
4.14	Wi-Fi Set up for the Raspberry Pi through the Android App	Dorsa	Fri 2-21-20	Thu 2-27-20	6	100%	5	6	0
4.15	Emergency service call page - Contact list	Jessica	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
5	**Raspberry Pi Client Software**	Jessica	Wed 1-29-20	Fri 1-31-20	3		3		
5.1	Movement Detection and modeling the triggers	Tymoore	Wed 1-29-20	Fri 1-31-20	2	100%	3	2	0
5.2	Communication of the device with the Server	Tymoore	Wed 1-29-20	Fri 1-31-20	2	100%	3	2	0

Date	Team members present	Discussions

All goals were achieved except moving the AI models to the Firebase was cancelled.

Appendix B: Gantt Chart - Old Schedule



Team #1 Face Detection System to Monitor Property Security

Go Pro

[Group1: University of Alberta - ECE492 - Capstone Project]

Team Members Names: Dorsa Nahid, Tymoore Jamal, Jessica D'Cunha, Bryn Leonard-Fortune

Project Lead: Dorsa Nahid

Project Start Date: 1-7-2020 (Tuesday)

Today's Date: 02-02-2020

Display Week: 9

Week 9 3 - 2 - 20 | Week 10 3 - 9 - 20 | Week 11 3 - 16 - 20 | Week 12 3 - 23 - 20 | Week 13 3 - 30 - 20 | Week 14 4 - 6 - 20 | Week 15 4 - 13 - 20 | Week 16 4 - 20 - 20

M T W Th F Sa Su

WBS	Task	Lead	Start	End	Cal Days	% Done	Work Days	Days Done	Days Left
1	**Milestones**		Tue 1-07-20	Fri 4-10-20	95		69		
1.1	Project Proposal	Dorsa	Tue 1-07-20	Mon 1-20-20	4	100%	10	4	0
1.2	Project Specifications	Tymoore	Tue 1-21-20	Mon 2-03-20	4	100%	10	4	0
1.3	Circuit Design Review Presentation	Jessica	Tue 2-04-20	Tue 2-04-20	4	100%	1	4	0
1.4	Progress Report	Bryn	Wed 2-05-20	Mon 3-02-20	4	100%	19	4	0
1.5	Final Presentation	Dorsa	Tue 3-03-20	Wed 4-08-20	4	100%	27	4	0
1.6	Final Report	Jessica	Thu 4-09-20	2020-04-10	4	50%	2	2	2
1.7	Application Notes	Tymoore	Fri 4-10-20	2020-04-10	4	0%	1	0	4
2	**Hardware**	Dorsa	Tue 1-07-20	Fri 3-20-20	74		54		
2.1	Construct Hardware Schematic	Bryn	Tue 1-07-20	Fri 1-17-20	10	95%	9	9	1
2.2	Acquire Parts	Jessica	Sat 1-18-20	Fri 1-24-20	6	100%	5	6	0
2.3	Connect and test the Camera	Bryn	Sat 1-25-20	Wed 1-29-20	4	100%	3	4	0
2.4	Connect and test the Motion Sensor	Jessica	Sat 1-25-20	Wed 1-29-20	4	100%	3	4	0
2.5	Design the PCB board	Bryn	Sun 2-02-20	Thu 2-20-20	18	100%	14	18	0
2.6	Set up the motor and test the motor	Tymoore	Sat 2-29-20	Mon 3-02-20	2	100%	1	2	0
2.7	Build a Box	Jessica	Sun 3-08-20	Fri 3-20-20	12	0%	10	0	12
3	**Artificial Inteligence**	Bryn	Tue 1-14-20	Mon 3-16-20	63		45		
3.1	Gather the right images	Bryn	Tue 1-14-20	Wed 1-22-20	8	100%	7	8	0
3.2	Pre-process the images	Bryn	Wed 1-15-20	Wed 1-22-20	7	100%	6	7	0
3.2	Collect test data using the camera	Bryn	Mon 2-03-20	Thu 2-20-20	17	0%	6	7	0
3.3	Train the model on faces & non-faces	Tymoore	Thu 1-23-20	Fri 1-24-20	1	95%	2	0	1
3.4	Pre-process the camera images	Bryn	Sat 2-01-20	Mon 3-16-20	44	60%	31	26	18
3.5	Test the camera	Tymoore	Wed 1-29-20	Thu 1-30-20	1	100%	2	1	0
3.6	Move the AI models to the Firebase	Tymoore	Sun 3-01-20	Mon 3-16-20	15	5%	11	0	15
4	**Android Application**	Jessica	Tue 1-21-20	Tue 3-10-20	50		36		
4.1	Setting up the Server	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
4.2	Storyboarding	Jessica	Tue 1-21-20	Fri 1-24-20	3	90%	4	2	1
4.3	API to call the server	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
4.4	Turn on the live feed	Bryn	Thu 2-27-20	Tue 3-10-20	12	45%	9	5	7
4.5	Control the motor through the application	Tymoore	Wed 3-04-20	Sat 3-07-20	3	0%	3	0	3
4.6	Login/Sign Up API includes unit tests	Dorsa	Fri 1-24-20	Wed 1-29-20	5	100%	4	5	0
4.7	Settings page (it includes the home location)	Jessica	Fri 1-24-20	Wed 1-29-20	5	100%	4	5	0
4.8	Logging API includes unit tests	Dorsa	Sat 2-01-20	Mon 2-10-20	9	100%	6	9	0
4.9	Right to forget me API includes unit tests	Jessica	Sat 2-08-20	Mon 2-10-20	2	65%	1	1	1
4.10	QR code reader	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
4.11	User location logging	Dorsa	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
4.12	Cache images and logs	Dorsa	Mon 2-17-20	Thu 2-20-20	3	100%	4	3	0
4.13	Encryption	Dorsa	Thu 2-20-20	Thu 2-27-20	7	100%	6	7	0
4.14	Wi-Fi Set up for the Raspberry Pi through the Android App	Dorsa	Fri 2-21-20	Thu 2-27-20	6	100%	5	6	0
4.15	Emergency service call page - Contact list	Jessica	Tue 1-21-20	Fri 1-24-20	3	100%	4	3	0
5	**Raspberry Pi Client Software**	Jessica	Wed 1-29-20	Fri 1-31-20	3		3		
5.1	Movement Detection and modeling the triggers	Tymoore	Wed 1-29-20	Fri 1-31-20	2	100%	3	2	0
5.2	Communication of the device with the Server	Tymoore	Wed 1-29-20	Fri 1-31-20	2	100%	3	2	0

Appendix C: Android Sample JUnit Tests

Testing of SecurityCamera.java

```java
package ca.ualberta.dorsa.seccam.entities;

import org.junit.Before;
import org.junit.Test;

import static junit.framework.TestCase.assertEquals;
import static junit.framework.TestCase.assertFalse;
import static junit.framework.TestCase.assertTrue;

/**
 * To test the methods in the SecurityCamera file
 */
public class SecurityCameraTest {
    private SecurityCamera sc;

    /**
     * Populate SecurityCamera object for testing.
     */
    @Before
    public void setUp() {
        sc = new SecurityCamera(
                "code",
                true,
                "image",
                true,
                "12345"
        );
    }

    /**
     * Checks that getting cameraCode is correct.
     */
    @Test
    public void getCameraCode() {
        assertEquals("code", sc.getCameraCode());
    }

    /**
     * Checks that setting currentImageString is correct.
     */
    @Test
    public void setCameraCode() {
        sc.setCameraCode("another");
        assertEquals("another", sc.getCameraCode());
    }

    /**
     * Checks that getting cameraEnabled is correct.
     */
    @Test
    public void getCameraEnabled() {
        assertTrue(sc.getCameraEnabled());
    }

    /**
     * Checks that setting cameraEnabled is correct.
     */
    @Test
    public void setCameraEnabled() {
        sc.setCameraEnabled(Boolean.FALSE);
        assertFalse(sc.getCameraEnabled());
    }

    /**
     * Checks that getting registered is correct.
     */
    @Test
    public void getRegistered() {
        assertTrue(sc.getRegistered());
    }

    /**
     * Checks that setting registered is correct.
     */
    @Test
    public void setRegistered() {
        sc.setRegistered(Boolean.FALSE);
        assertFalse(sc.getRegistered());
    }
}
```

Testing of LoginActivity.java

```
/**
* To test that login activity will switch activities after a successful login
*/
@RunWith(AndroidJUnit4ClassRunner.class)
@LargeTest
public class LoginActivityTest {
   private Context context;
```

```
    /**
     * Set up the context
     */
    @Before
    public void setup() {

        context = ApplicationProvider.getApplicationContext();
        assertNotNull(context);
    }
    /**
     * Use LoginActivity.class
     */
    @Rule
    public ActivityTestRule<LoginActivity> mActivityRule =
            new ActivityTestRule<>(LoginActivity.class);
    /**
     * Ensure activity change.
     */
    @Test
    public void ensureActivityChange() {
        // Type text and then press the button.
        onView(withId(R.id.emailLogin))
                .perform(typeText("d@yahoo.com"), closeSoftKeyboard());
        onView(withId(R.id.passwordLogin))
                .perform(typeText("123456"), closeSoftKeyboard());
        onView(withId(R.id.signInButton)).perform(click());
        Instrumentation.ActivityMonitor activityMonitor =
getInstrumentation().addMonitor(LogActivity.class.getName(), null, false);

        LoginActivity myActivity = this.mActivityRule.getActivity();

        LogActivity nextActivity = (LogActivity)
getInstrumentation().waitForMonitorWithTimeout(activityMonitor, 5000);
        assertEquals(nextActivity.getClass().getName(), LogActivity.class.getName());
        nextActivity.finish();
    }
}
```