**CT4003 – Assignment 2 - Report**

**Arduino Security Robot – The Kenobis**

**Contents**

[**Team Details and GitHub Account** 2](#_Toc383516921)

[**Project Proposal – Arduino Security Bot** 2](#_Toc383516922)

[**Requirements** 2](#_Toc383516923)

[**Running System Screenshots** 3](#_Toc383516924)

[**Java Application** (Eclipse Developer) 3](#_Toc383516925)

[**Java Application** (.jar Version) 3](#_Toc383516926)

[**Arduino** (Serial Monitor Pre Set-up) 4](#_Toc383516927)

[**Arduino Circuit** 4](#_Toc383516928)

[**Email Received** 5](#_Toc383516929)

[**Flow Diagrams** 6](#_Toc383516930)

[**Production Issues** 7](#_Toc383516931)

[**Arduino to Java Serial Communication** 7](#_Toc383516932)

[**Ultrasonic Sensor Calibration** 7](#_Toc383516933)

[**GitHub Issues** 7](#_Toc383516934)

[**Simple Date Format** 8](#_Toc383516935)

[**Reflection** 8](#_Toc383516936)

[**Test Cases** 9](#_Toc383516937)

[**Trace Tables** 9](#_Toc383516938)

[**Review Meetings** 9](#_Toc383516939)

[**Team Member Contribution** 10](#_Toc383516940)

[**Andrew** 10](#_Toc383516941)

[**Carl** 11](#_Toc383516942)

[**Josh** 11](#_Toc383516943)

[**Reece** 11](#_Toc383516944)

[**References** 11](#_Toc383516945)

# Team Details and GitHub Account

**Team Name**

The Kenobis

|  |  |
| --- | --- |
| **Team Members** | **Student Number** |
| Andrew Spackman | s1309454 |
| Carl Berquist | s1310712 |
| Josh Ravenscroft | s1303892 |
| Reece Kelly | s1304388 |

**GitHub Account Details**

Username: TheKenobis

Email: s1309454@gmail.com

Password: TheKenobis1

URL: https://github.com/TheKenobis/ArduinoSecurityBot

**Project Proposal – Arduino Security Bot**

Our proposed idea for assignment 2 of CT4010 is the Arduino Security Bot. The security bot will be used to detect motion in a specified area and when motion is detected for a specific period of time, the security bot will then send an email to a specified email address as a response and keep a textual log of activity.

**Requirements**

Arduino with various components attached:

* Requirement 1: Ultrasonic Sensor for motion detection
* Requirement 2: LEDs for confirmation of Security Bot status
* Requirement 3: Button for switching the system on and off

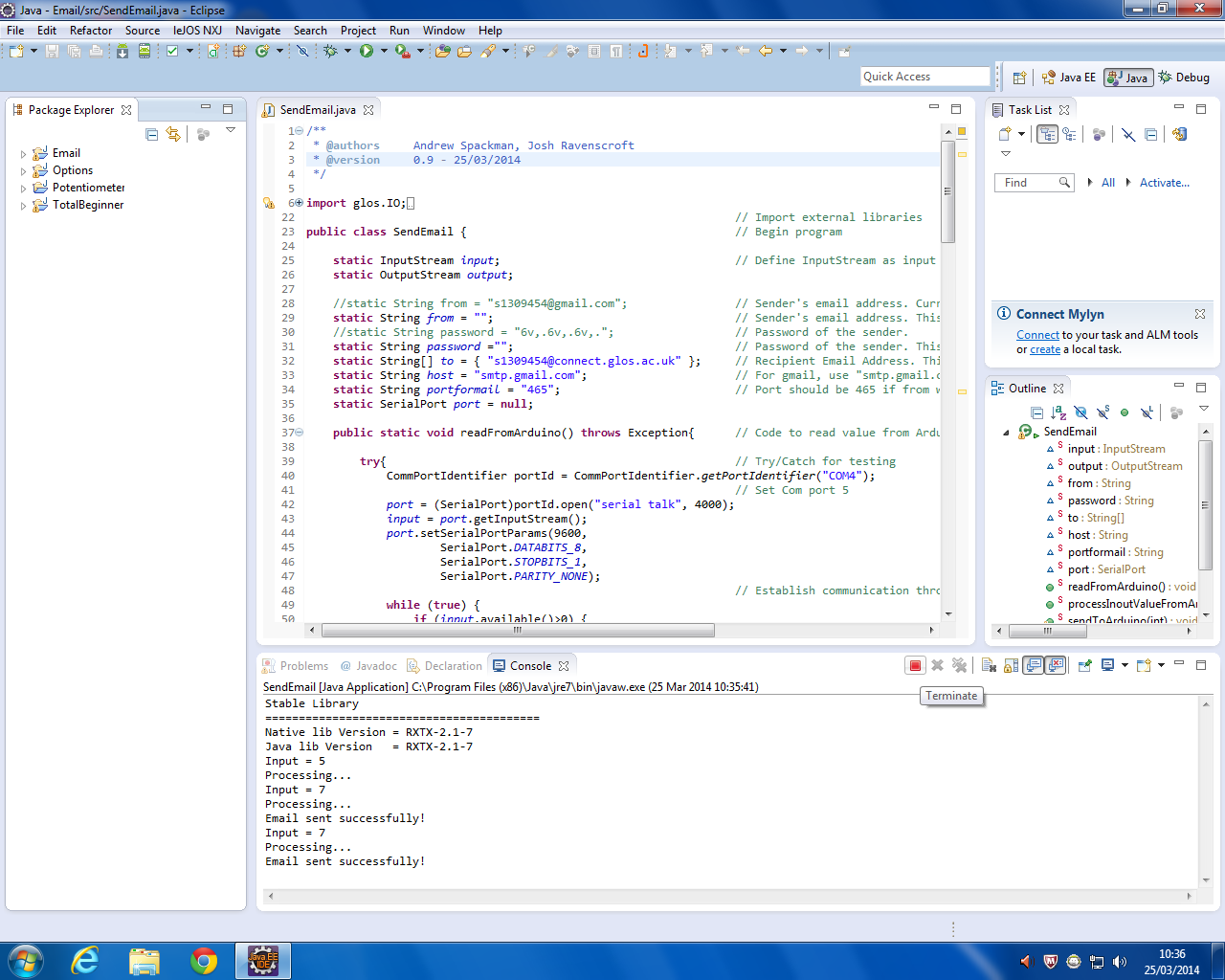
Java Application which will send and receive status to and from the Arduino, with automatic port finder, including:

* Requirement 4: Receive Arduino status through serial port
* Requirement 5: Use serial port input as part of the program
* Requirement 6: Determine next steps in relation to input
* Requirement 7: Send an email to a recipient
* Requirement 8: Confirm sending status

The Java application will also send an email to a specified recipient in response to the input received from the Arduino. This will include the email recipient’s address, subject header and email contents and timestamp. The email component will work through java to ensure that is sent successfully in relation to the Arduino’s input. It will also maintain a text log of activity as a fail-safe to ensure that if an email isn’t sent successfully, a textual log can be accessed from the computer system. The textual log will also confirm email sending status (sent or not sent).

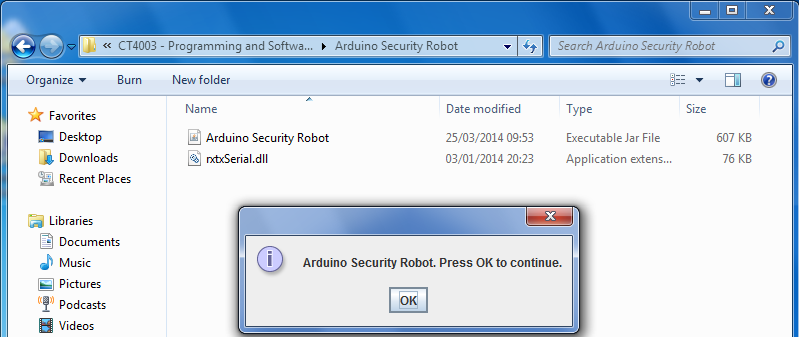
**Running System Screenshots**

## Java Application (Eclipse Developer)

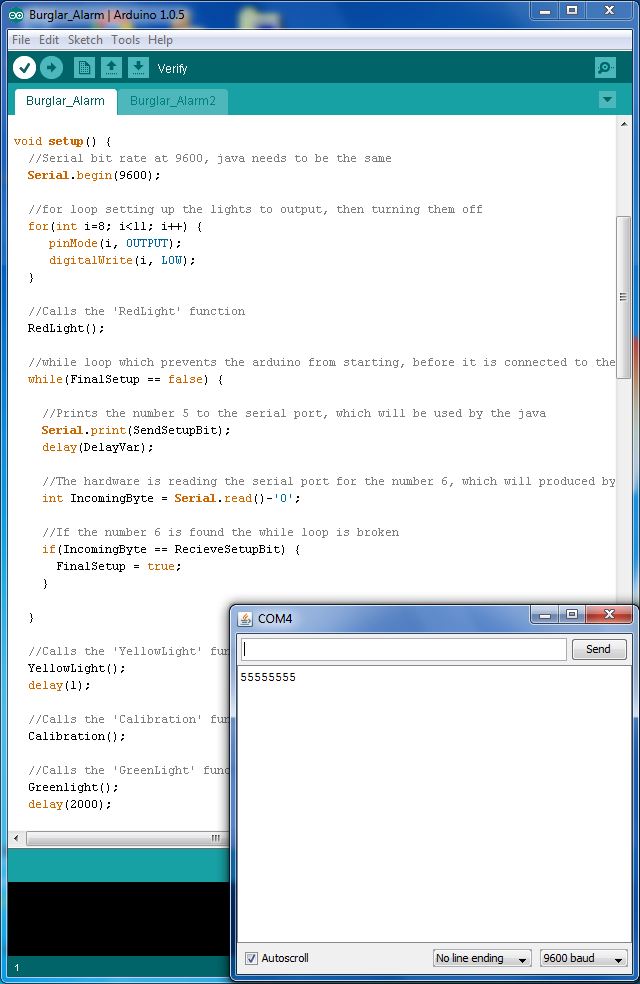


## 

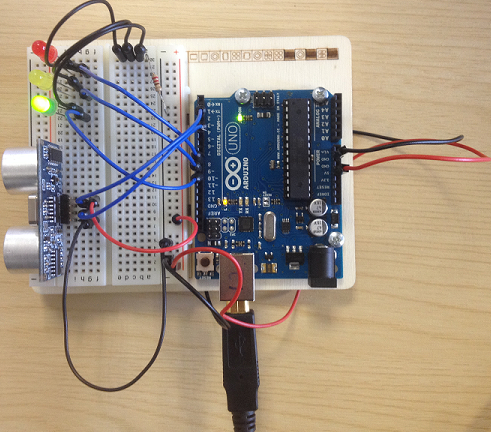
## Java Application (.jar Version)



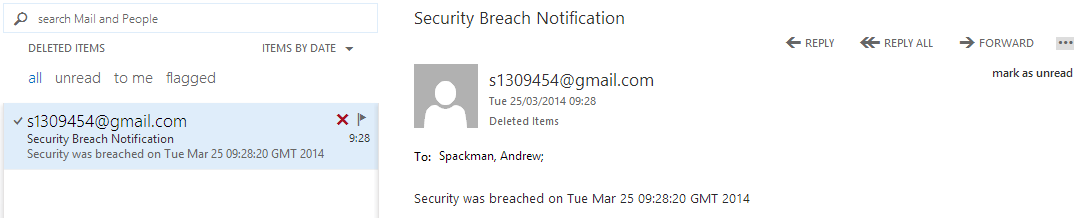
**Arduino** (Serial Monitor Pre Set-up)



## Arduino Circuit

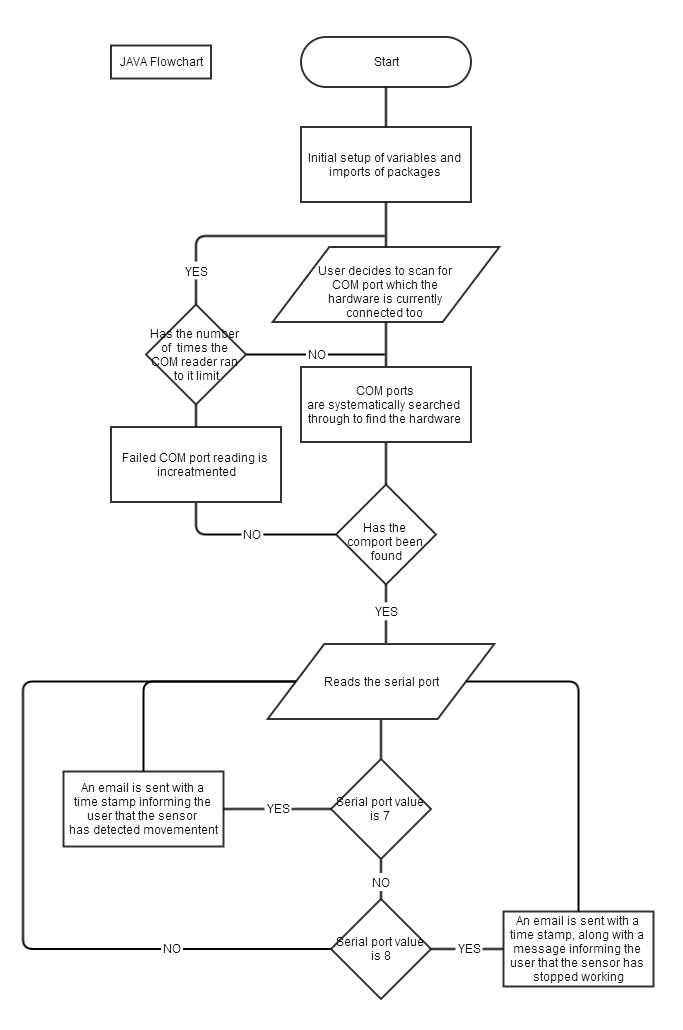


## Email Received



# Flow Diagrams

**Arduino Java**



# N:\GitHub\ArduinoSecurityBot\Arduino Flow.pngProduction Issues

## Arduino to Java Serial Communication

One of the main problems that we experienced during the programming of the Arduino Security Robot was an issue with the serial communication between the Arduino programming and the Java application. We were led to believe that the java program wasn’t functioning as intended and wasn’t executing specific sections of code. This culminated in 4 hours of testing which, unfortunately, didn’t provide an answer. We then implemented a small piece of coding which printed the value of the variables that were being sent to and from the 2 programs and found that the values were different from what we were expecting. We were expecting the number “5” to be printed and sent but instead we were getting “53”. After some discussion with our lecturer, we discovered that the value “53” was correct because it was printing an ASCII character of “53”, which is the numerical 5. After including a small piece of code to fix this (- '0') the problem was corrected and the java program then printed “5” as intended.

## Ultrasonic Sensor Calibration

At one point in the development of the project, the security bot wasn’t calculating an initial distance correctly, which was to be used in detecting possible movement. This issue was the result of the Arduino skipping code due to a lack of delay commands. This problem was then resolved by adding in multiple ‘delay (1)’ commands were needed (where 1 is one millisecond). This provided small breaks within the code, allowing the Arduino to process all commands, one at a time, without issues.

Another issue, which involved skipping, occurred when using the serial monitor. This was only a problem when trying to debug the system without the java program. The simple solution we used to fix this was to test the code without using the serial monitor and instead, using LEDs to show the input and output.

## GitHub Issues

We used Github to transfer files between each member and keep track of the work that each member of the group completed. However, we experienced a few issues while using Github:

* Eclipse to Github Uploading

The first issue was using Eclipse to upload new versions to Github. We couldn’t connect Eclipse to Github so we had to use Githubs own software to upload the new versions. It would have been faster to be able to upload the java through eclipse as that would remove an extra step.

* Images and Thumbnails

The Github uploading software loads hidden files that are in the folders such as the thumbs.db files that are hidden in Windows. The thumbnails that were uploaded caused a conflict when images were uploaded as the thumbs.db was trying to reference different images. To fix this we had to manually delete all the thumbs.db files that got uploaded to Github.

* Failed to Sync

The main error that the Github uploading software reported was that it failed to update the repository with new commits. To resolve this error we had to delete the local repository on the computer and re-download the repository to the local machine.

## Simple Date Format

During the beginning of the Java development process, we had declared in our proposal that we would include a time stamp on emails that were to be sent to the recipient. This would state when the breach had occurred. While undertaking research, I found the simple date format code, which was an easy function to implement into our code. However to use this function we would have had to use an external file: the “java.simpletimeformat” library, like so:

* import java.text.SimpleDateFormat;

As this was the case, we did not feel it was necessary as we would have to include this external library. It would also be counter-intuitive because java already provides a similar function which is already available.

**Reflection**

As a collective, we all agree that the project was a success in the end. We have successfully created an Arduino program which connects to a java application, sends and receives data via serial port, collects and uses values from an Ultrasonic sensor and then sends an email in response. Our original proposal may been a little ambitious for the time scale with which we had for the project but the final result is functional and achieves the basics of our initial goal, which we are all agreed to be happy with.

There are a number of potential improvements that, given additional time, we would have liked to have implemented such as a variable for the COM port so that we don’t have to manually change this value, a more user-friendly GUI including the ability to change the recipient’s email address, email and password verification (it currently accepts any value) and a camera that takes a picture when the security has been breached, allowing the recipient to see what the cause of the breach is.

# Test Cases

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Requirement ID** | **Module To Be Tested** | **Case Type** | **Test Case Description** | **Expected Outcome** | **Actual Outcome** |
| TC001 | RA001 | Ultrasonic Sensor for motion detection | Positive | Initiate the system, and place an object in the path of the ultrasonic sensor | The Arduino would pick up the movement and send a seven to the Java code to initiate the breach email | The Arduino picked up the movement and thus sent an email to the specified recipient. |
| TC002 | RA001/N | Ultrasonic Sensor for motion detection | Negative | Initiate the system, move the Arduino board from its original location | The Arduino board would re-collaborate to the next wall to a distance of two metres | The Arduino board re-calibrate to the next wall to a distance of two metres. |
| TC003 | RA002 | LED's for confirmation of Security Bot status | Positive | When initialised the Arduino boards LED lights will show at what stage it is currently at | The Arduino shows a red LED for not collaborated, yellow LED shows that it is collaborating and green LED to state it is ready to receive movement | The Arduino shows a red LED for not collaborated, yellow LED shows that it is collaborating and green LED to state it is ready to receive movement |
| TC004 | RA002/N | LED's for confirmation of Security Bot status | Negative | When the Arduino is knocked and is facing another object, the Arduino system will loop and therefore LED's will start over. | The LED's will start over when the Arduino has re-calibrated. | The LED's loop when the Arduino re-calibrated |
| TC005 | RA003 | Button for switching the system on and off | Positive | The button is pressed to turn the Arduino off and on. | When the button is pressed once it turns on and starts calibrating and then the second time it turns the Arduino off. | The Arduino turned off and on when expected and ran the Arduino as wanted/expected. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
| **Test Case ID** | **Requirement ID** | **Module To Be Tested** | **Case Type** | **Test Case Description** | **Expected Outcome** | **Actual Outcome** |
| TC007 | RJ004 | Receive Arduino status through serial port | Positive | The Java program will receive a set of numbers stating what status the Arduino is currently at. | The Java will receive the number 5, 6, 7 and 8. All showing the various Arduino status | The Java received the number 5 from the Arduino and sent back a 6 to confirm communication. Arduino printed the number 7 to show a breach which then ran the Java code and received an 8 if there was a problem. |
| TC008 | RJ005 | Use serial port input as part of the program | Positive | Initialise the program and the Java and Arduino will receive and send their status through the serial port. | The Java and Arduino will receive and send numbers through the serial port | The java and Arduino sent numbers through the serial port stating their status. |
| TC009 | RJ005/N | Use serial port input as part of the program | Negative | Through the Arduino send a number other than 5 to 8. | When the number is types through the Arduino serial port nothing will happen. | When the number 4 was sent through the serial port nothing happened as the Java had not been told do anything when receiving the number 4 |
| TC010 | RJ006 | Determine next steps in relation to input | Positive | When the numbers 7 and 8 is received it will implement the send email code either of a breach or a error with the program. | The Java will send a email to the recipient stating whether there was a breach or that there is an error with the program. | The Java sends an email to the recipient with the information that there was a breach when received a 7 and an email stating that there is an error with the program when an 8 was sent. |
| TC011 | RJ007 | Send an email to a recipient | Positive | When the number 7 is received from the Arduino it will send a automated email to the specified recipient. | A email will be sent when the number 7 is received to the recipient specified. | A email was sent when the number 7 was received to the recipient specified. |
| TC012 | RJ007/N | Send an email to a recipient | Negative | When a different number is sent it should not send an email to the specified recipient. | By sending a number other than 5,6,7,8 nothing should happen as it has not been programmed to do anything. | Nothing happened when sending the number 9 as it had not been programmed to do so. |
| TC013 | RJ008 | Confirm sending status | Positive | On the Java serial port when the email is sent it will state that the email was sent successfully. | In the Java serial port it will state that the email was successfully sent to the recipient. | In the Java serial port it stated that the email was successfully sent to the recipient. |
| TC014 | RJ009 | Turning off Java when Arduino is running | Negative | When the overall program is running, turn the Java program off but leave the Arduino program running. | The Arduino program will still keep running even when the Java is turned off. | The Java code was turned off and the Arduino code kept functioning and when there was movement it still displayed a seven on the serial port but however nothing happened because the Java was turned off. |

# 

# Trace Tables

## Arduino Trace Tables

|  |  |  |  |
| --- | --- | --- | --- |
| **LED Setup** | | **‘i =int’** | |
| **Line Number** | **Calculation** | **Expected Outcome** | **Actual Outcome** |
| 30 | i=8 | 8 | 8 |
| 31 | N/A | 8 | 8 |
| 32 | N/A | 8 | 8 |
| 30 | i++ | 9 | 9 |
| 31 | N/A | 9 | 9 |
| 32 | N/A | 9 | 9 |
| 30 | i++ | 10 | 10 |
| 31 | N/A | 10 | 10 |
| 32 | N/A | 10 | 10 |
| 30 | i++ | 11 | 11 |
| 33 | END | END | END |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Serial Communication** | | **‘FinalSetup = Boolean’** | | **‘IncomingByte = int’** | |
| **Line Number** | **Calculation** | **Expected Outcome** | **Actual Outcome** | **Expected Outcome** | **Actual Outcome** |
| 39 | FinalSetup == FALSE| N/A | FALSE | FALSE | 0 | 0 |
| 42 | N/A | N/A | FALSE | FALSE | 0 | 0 |
| 43 | N/A | N/A | FALSE | FALSE | 0 | 0 |
| 46 | N/A | IncomingByte = Serial.read() - '0' | FALSE | FALSE | 0 | 0 |
| 49 | N/A | IncomingByte == ReceivedSetupByte | FALSE | FALSE | 0 | 0 |
| 39 | FinalSetup == FALSE| N/A | FALSE | FALSE | 0 | 0 |
| 42 | N/A | N/A | FALSE | FALSE | 0 | 0 |
| 43 | N/A | N/A | FALSE | FALSE | 0 | 0 |
| 46 | N/A | IncomingByte = Serial.read() - '0' | FALSE | FALSE | 6 | 6 |
| 49 | N/A | IncomingByte == ReceivedSetupByte | FALSE | FALSE | 6 | 6 |
| 50 | FinalSetup = TRUE | N/A | TRUE | TRUE | 6 | 6 |
| 51 | N/A | END | TRUE | TRUE | END | END |
| 39 | FinalSetup == FALSE| END | TRUE | TRUE | END | END |
| 53 | END | END | END | END | END | END |

## Java Trace Tables

**Review Meetings**

4th March 2014

**Attendees:**

* Andrew, Carl, Josh and Reece. No absences.

**Accomplishments:**

Andrew and Josh created the initial email code for java. Andrew uploaded some of the code to GitHub.

A base circuit with the Arduino was created by Reece and Carl which underwent some serial communication testing. Initial software for the Arduino was created by them both.

11th March 2014

**Attendees:**

* Andrew, Josh and Reece. Carl was absent.

**Accomplishments:**

Andrew and Josh integrated the java code for serial port communication with the java email code to create a rough draft of the final java code. Reece created a new, finalized circuit and uploaded the subsequent code to GitHub. The three of us then attempted a working version of the overall project. Initial test case template was completed by Josh with reference to the requirements stated in the proposal; this was agreed upon by all group members in attendance.

16th March 2014

**Attendees:**

* Andrew, Carl and Reece. Josh was unavailable.

**Accomplishments:**

We all established a communication connection between Arduino and Java programs. Andrew added a try/catch for the read/from and send/ to code. Reece made minor edits to the Arduino circuitry.

18th March 2014

**Attendees:**

* Andrew, Carl, Josh and Reece. No absences.

**Accomplishments:**

Andrew and Carl created first final draft of java code with comments. Carl submitted modified Arduino code. Josh created further test cases for code testing. Reece rewrote various sections of Arduino code for a more stable input and output process.

19th March 2014

**Attendees:**

* Andrew, Carl, Josh and Reece. No absences.

**Accomplishments:**

Andrew and Josh finalised the Java program and underwent testing which was successful. Carl and Reece optimised the Arduino code further and implemented delays in order to stop errors. We all created a first test video and began creating subsequent documentation.

25th March 2014

**Attendees:**

* Andrew, Carl, Josh and Reece. No absences.

**Accomplishments:**

Andrew and Reece exported the java program to a .jar file. Josh finalized the test cases and assisted with other documentation. Carl finalised the Arduino code and submitted it to GitHub, along with his personal statement. We all prepared for the submission date by ensuring all loose ends were tied up and that documentation was almost completed.

25th March 2014

**Attendees:**

* Andrew, Carl, Josh and Reece. No absences.

**Accomplishments:**

**Team Member Contribution**

**Andrew**

As a team member of The Kenobis, I spent a lot of time focusing on the java application and the documentation. The java application required communication with the Arduino, with I unintentionally led, and the integration of the email library to send emails which, with the help of Josh, was accomplished with relative ease.

The documentation, however, was created with equal measure by the team and included various components from each of us. I am particularly pleased with the final product.

As a whole, I feel I have positively contributed to the project with both ideas and expertise.

I struggled with several issues during the production of the project and originally approached them with the same brute force method I was accustomed to with my own time spent working with assembly code. I also had to hold back my tendency to lead and take control of the project as a leader when we were all equals within the team. This was particularly challenging and I still feel that I need to work on this as it started to happen here as well.

**Carl**

**Josh**

During the overall group project, my main contribution and role was within the Java application, adapting the sending email function to the specified recipient. With Andrew, we were able to implement a simplified time stamp onto the email and to send the email when the number seven was received from the Arduino through the serial port. Furthermore I aided Andrew and the rest of the group to the best of my ability in any queries or problems that occurred throughout the overall process.

In the very beginning stages of the project I was a too ambitious with the Java code I wanted to implement. For example, the verification process in which I wanted to use but felt that it would take a lot more work to fully develop and implement and so I decided it wasn’t needed.

I also contributed to the group by creating and undertaking the test cases. I included positive and negative test case types to give myself and the group a better understanding of the program and what will happen to the final program in certain situations. Currently I have created the video in which will be presented during the presentation phase of the assignment which will take place after submission.

**Reece**

During the initial discussion of possible ideas, I came up with the idea of a personal motion detector. After an extensive discussion of possible adaptions, we all decided that an ultra-sonic module would be the ideal component.

After the initial idea was developed, I created flow diagrams for both systems to help illustrate how the programs should operate. These diagrams were based on our initial ideas, so certain parts weren’t implemented into the final product, like a variable for the COM port. The flow diagrams also contain missing features, such as user email and password input which weren’t implemented due to time constraints.

I created the setup phase for the Arduino code, which allows the automatic start-up of the Ultrasonic sensor. I also colour-coordinated the LED lights, along with the final optimisation of the code, which involved replacing repeating sections with functions. I also debugged the program alongside its creation.

Areas which I wanted to improve on included, being too ambitious with the Arduino side of the project and as a result wasted time on unfeasible ideas, such as researching a possible web camera adaption.

**References**

Notes:

Word Count – As long as criteria is covered, this isn’t an issue.

Presentation – Use the video to showcase the project working. Explain in detail afterwards