

**House Key Lockbox**

**Bachelor of Engineering Electronic Engineering**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Author: Godspower Olasupo** |  |  |  | **Student ID: x00150107** |
|  |  |  |  |  |
|  |  |  |  |  |
| **Supervisor:** | ***Michael Gill*** |  |  | **Date:04/05/2021** |

**Declaration**

This report is prepared as a partial fulfilment towards graduation requirements for Bachelor of Engineering (Honours) in Electronic Engineering, at the Institute of Technology Tallaght. I declare that the contents of this report, and the project to which it refers, are entirely the work of the author and have not being submitted for a degree in any other institute.

Signed…………………………………………………………………………….

Date……………………………………………………………………………….

Contents

[**Abbreviations** 4](#_Toc71054034)

[**List of Figures** 5](#_Toc71054035)

[**Preface** 6](#_Toc71054036)

[**Abstract** 6](#_Toc71054037)

[**Acknowledgements** 7](#_Toc71054038)

[**Introduction** 8](#_Toc71054039)

[**Project Plan** 12](#_Toc71054040)

[**Background** 14](#_Toc71054041)

[**What is a Sensor and what do they do?** 17](#_Toc71054042)

[**Socket.io** 18](#_Toc71054043)

[**Node.Js** 20](#_Toc71054044)

[**Seeduino Lotus** 21](#_Toc71054045)

[**Design** 25](#_Toc71054046)

[**Web Server** 26](#_Toc71054047)

[**Web Interface** 28](#_Toc71054048)

[**Arduino IDE** 28](#_Toc71054049)

[**Implementation.** 28](#_Toc71054050)

[**Arduino Code** 38](#_Toc71054051)

[**Running Code** 42](#_Toc71054052)

[**Future Modification** 45](#_Toc71054053)

[**Bill of Materials** 46](#_Toc71054054)

[**Poster Section** 47](#_Toc71054055)

[**Environment and Ethics** 48](#_Toc71054056)

[**Conclusion** 49](#_Toc71054057)

[**References and Bibliography** 50](#_Toc71054058)

[**Appendix** 51](#_Toc71054059)

**Abbreviations**

This is a List of all the abbreviations in this report.

* LED = Light Emitting diode
* JS = JavaScript
* DOM = Document Object Model
* PC = Personal Computer
* CSS = Cascading style Sheet
* HTML = Hypertext Markup Language
* LDR = Light Dependent resistor.
* USB = Universal Serial Bus
* HTTP = Hypertext Transfer Protocol

# **List of Figures**

[Figure 1 Typical Hardware Setup for Lockbox 8](#_Toc71028960)

[Figure 2 communication between client and server [2] 9](#_Toc71028961)

[Figure 3 the EOZ IP40 12 Key Keypad [8] 11](#_Toc71028962)

[Figure 4 Snippet of different topics in HTML,CSS and JavaScript[6] 15](#_Toc71028963)

[Figure 5 The HTML DOM Tree of Objects 15](#_Toc71028964)

[Figure 6 Snippet of calculator in chrome 16](#_Toc71028965)

[Figure 7 light sensor on Groove Beginner kit 17](#_Toc71028966)

[Figure 8 Index.html 19](#_Toc71028967)

[Figure 9 chat messages 19](#_Toc71028968)

[Figure 10 Modified chat program 20](#_Toc71028969)

[Figure 11 Modified chat program 21](#_Toc71028970)

[Figure 12 displays the Seeduino Lotus. 22](#_Toc71028971)

[Figure 13 Hardware Overview of Grove Beginner Kit. 23](#_Toc71028972)

[Figure 14 Zigbee device 24](#_Toc71028973)

[Figure 15 Workflow of Apache web server [12] 27](#_Toc71028974)

[Figure 16 Validate Function. 29](#_Toc71028975)

[Figure 17 Onclick calling validate function. 29](#_Toc71028976)

[Figure 18 Webpage HTML 30](#_Toc71028977)

[Figure 19 Tags contained in the head tag. 30](#_Toc71028978)

[Figure 20 Snippet of a bit of the code in the style tag. 31](#_Toc71028979)

[Figure 21 Login Screen 31](#_Toc71028980)

[Figure 22 Code for Welcome.html 32](#_Toc71028981)

[Figure 23 Contents of the script tag in the welcome.html 33](#_Toc71028982)

[Figure 24 Debug Console of port opened and listening. 33](#_Toc71028983)

[Figure 25 Lockbox Login Webpage 34](#_Toc71028984)

[Figure 26 Welcome webpage. 34](#_Toc71028985)

[Figure 27 Snippet of webpage.js 35](#_Toc71028986)

[Figure 28 Snippets of Ardunio code 38](#_Toc71028987)

[Figure 29 Physical Equipment used for Lockbox Simulation. 42](#_Toc71028988)

[Figure 30 Snippet of Datasheet of Keypad [4] 46](#_Toc71028989)

[Figure 31 Semester 1 poster 47](#_Toc71028990)

# **Preface**

This report was written throughout the Duration of my final year. This was the main project that was accomplished when trying to acquire my Degree in Software Engineering. This project was done at Technological University Dublin in the Tallaght Campus located in Ireland. The Project’s aim was to create s simulated House Key Lockbox.

# **Abstract**

This report will detail how a House Key Lock-box project was designed and implemented. Two criteria’s that the project required was the inclusion of sensors and a web server that can be used to access and perform the different tasks that are required of the Lockbox system. The basis of the House Key Lockbox is for the use of real estate agents or rental companies. They would be able to control a system that would allow them access to the keys of the houses. The lockbox would be opened by inputting a certain 4 figure code into the keypad. An authorised individual would need to login to the website created using the different web technologies to acquire the required 4 figure code. The project involves a lot of self-learning in web technologies such as HTML, JavaScript and Socket.ie. Throughout the project, the progression in knowledge of different aspects such as the web interface design and Arduino will prove to be crucial in completing a functioning House Key Lockbox, that can be controlled using a website specifically designed for that use. Aspects such as passcode control will also be included in the operation of the lockbox.

# **Acknowledgements**

I would like to acknowledge Michael Gill for his advice and guidance throughout the duration of the project, especially when it came to coding. I would also like to acknowledge the examples of the posters that were made by previous students which helped a lot when it came formulating how I wanted to approach my project. I would also like to acknowledge Dave Maguire who is the technical advisor for providing us with any equipment that we required during the project. The University provided us with Grove Beginner kit boards that were very useful. I am very grateful to the University for all they skills they have taught me including research skills and report writing skills.

# **Introduction**

As partially explained in the abstract, this project would require a culmination of different aspects to come together to form the finished product, which is a fully functioning House Key Lockbox. It can be controlled by a real estate agent or company to provide access to a certain property.

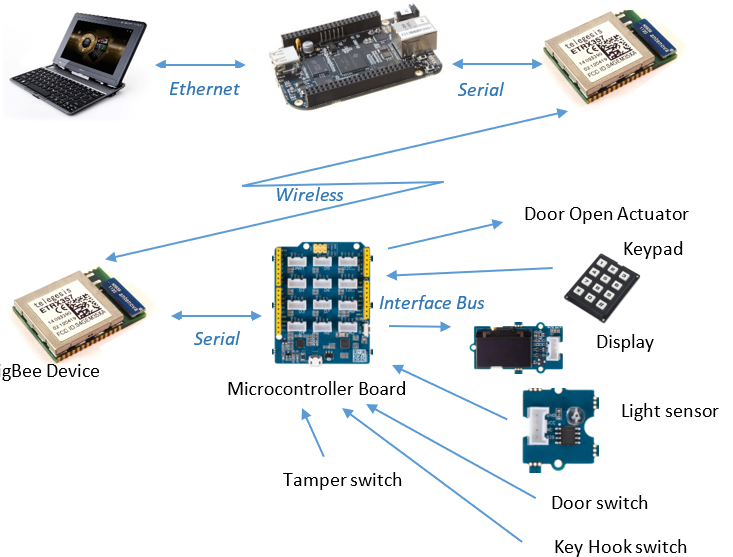


Figure 1 Typical Hardware Setup for Lockbox

Certain sensors will be used in the project. The light Sensor will be used in this project to identify if the lockbox is truly closed. If the light sensor is still able to detect light intensity of a certain value, the box will not be closed. This could indicate that the lockbox has been tampered with, so in return the operation would remain in the door opened sate.

This report will explain how the project involves the use of a web server, a Grove Beginner Kit, web technologies and Arduino. The project will be simulated by a PC and some hardware products.

The system will have both client and server software. The client software will run in the browser and the server software will run in the Linux board. The board is assigned the task of serving up the webpages to the client machines. The server software can be implemented in JavaScript using the Node.js framework. The website will also have to incorporate Socket.io into it. The purpose of Socket.io is to allow for communication in both ways between the client server and the web server.

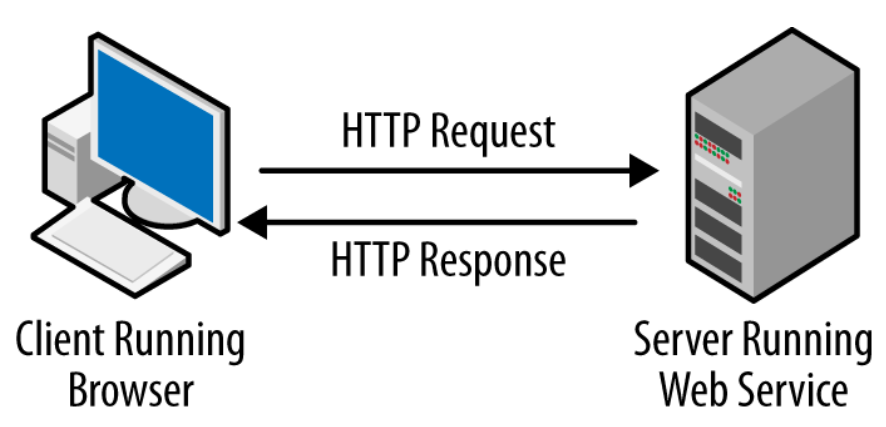


Figure 2 communication between client and server [2]

The features added to the Lockbox to maintain the security will include the use of the buzzer and 3-axis accelerator. Both would have the purpose of signifying when the Lockbox is being tampered with. In this case if the y axis or x axis is greater than a certain number, it is evident that the Lockbox is being moved and this will trigger the Buzzer to make noise if the location of the axis’s stays above that number. At the same time when the Buzzer alerts that the Lockbox is being tampered with, a message is displayed on the OLED display. The message reads “Lockbox Tamper”. At this stage, no buttons will be read from the keypad again. This is the feature that was implemented in my Project.

Ideally there are many other features that would have aided in the overall security of the Lockbox. Some of these features are listed below:

* Features like the duration of time the 4-figure code remains valid, the system being able to detect if a key is in the Lockbox and if the Lockbox door is open or closed.
* A tamper switch could be used to show if the Lockbox were forcibly opened or removed from the wall.
* A sensor that detects if the key gets removed from the box, meaning that it is constantly checking if the key is still in the Lockbox, so it knows whether the key is in the lockbox.

For the project, a website was created using the acquired knowledge of HTML, JavaScript, and Cascading Style Sheets (CSS). Such tasks include having a specific code for the simulated Lockbox. The user would need to be able to login to receive the code for a specific lockbox. The website was designed in a way where there are three Lockboxes shown alongside their respective passcodes. Ideally the website would be designed to only show the code for one Lockbox at a time and not all of them. The website was designed in a way in which it can receive and send information to the Lockbox. When it comes to designing the website, the application Visual Studio code was used. Visual Studio Code is very fast and useful application. Once logged in, the user would be able to select the lockbox that they wish to open. The login page that was designed in the HTML would be able to transition to a welcome HTML. In the welcome HTML the user would be able to select the passcode they required for a specific Lockbox. There are three different Lockboxes and there are also three different Lockbox passcodes available.

The Grove Beginner Kit for Arduino was a very important aspect of this project. The Grove beginner kit has a wide array of sensors and other useful parts programmed onto it that could be useful in the project. These include the OLED display, a few LEDs, the light sensor, the Buzzer, and the 3-axis accelerator. An external keypad would also be connected to the board. The keypad used is called the EOZ IP40 12 Key Keypad. The keys are inputted on the keypad and displayed on the OLED display.



Figure 3 the EOZ IP40 12 Key Keypad [8]

Main Objectives

* Learn how the Grove Beginner kit board works.
* Create a Project Plan to structure how the project should be undertaken.
* Learn how to use HTML, JavaScript, and CSS on [www.w3schools.com](http://www.w3schools.com)
* Become proficient in writing code in Arduino.
* Use these skills to create a client friendly website.
* Learn how to code the use of sensors and output devices into the Arduino.
* Have a good understanding of Socket.io, to create the communication link.
* Understand how a lockbox works and how to integrate the operation of it with the web server.

# 

# **Project Plan**

|  |  |  |
| --- | --- | --- |
|  | | |
| Week No. | Dates: | Description: |
| 1 | 28/09/20 | Selection/Tutorial of Project |
| 2 | 05/10/20 | **Web Interface Design and implementation**   * HTML * JavaScript * Cascading Style Sheets (CSS * jQuery * Document Object Model (DOM)   Practiced implementing and learning to use these to create a calculator and later, a web user interface |
| 3 | 12/10/20 |
| 4 | 19/10/20 |
| 5 | 26/10/20 |
| 6 | 09/11/20 |
| 7 | 16/11/20 | **Web Server**   * NodeJS * Socket.io   Used in developing a web browser that would allow a communication link to be establish between a client and the server via a serial connection. |
| 8 | 23/11/20 |
| 9 | 30/11/20 |
| 10 | 7/12/20 | **Lockbox design and testing of beginner kit board**   * Light sensor * Switches * Creating web page to communicate with the board |
| 11 | 14/12/20 |
| 12 | 21/12/20 |
| 13 | 28/12/20 | **Planning Layout of Report:**   * Creating a plan on when to begin and when I expect to finish. |
| 14 | 4/1/21 |
| 15 | 11/1/21 | **Doing research on the different aspects that must be included in report** |
| 15 | 18/1/21 | **Creation of the Web page and web server**   * Use of the web technologies learned to code in Visual Studio code * Ensure information can be transmitted from webpage to board. The link must work both ways |
| 16 | 25/1/21 |
| 17 | 1/2/21 |
| 18 | 8/2/21 |
| 19 | 15/2/21 | Arduino work   * Coding the OLED display * Wiring up keypad to the Seeduino lotus board * Adding light sensor to program * Adding buzzer and 3 axis accelerator |
| 20 | 22/2/21 |
| 21 | 1/3/21 |
| 22 | 8/3/21 |
| 23 | 15/3/21 |
| 24 | 22/3/21 | **Writing Report**   * Researching and accumulating all the background information needed for the Project. * Acquiring all the necessary diagrams and code snippets. |
| 25 | 29/3/21 |
| 26 | 5/4/21 |
| 27 | 12/4/21 |
| 28 | 19/4/21 |
| 29 | 26/4/21 |
| 30 | 4/5/21 | **Submission of Report** |

# **Background**

This project requires a lot of background knowledge that was not specifically taught in the University so it challenges the students to not only do a lot of their own research but to be able to teach themselves new information that could aid them greatly in the progress of the project.

Research began by learning how to create a calculator. A few weeks were allocated at the beginning to acquire all the necessary information needed to the write code for the calculator in Visual studio Code. Research was done into web interface designing. The main objective at the end of everything was to be able to develop a functioning calculator, which lead to the design of a web user interface for the Lockbox.

The purpose of initially learning to build a calculator was so students could get practice into applying the use of the studied web technologies. The practice was done on the website [www.w3schools.com](http://www.w3schools.com). This website gave a full in-depth tutorial into how HTML worked and all the different aspects that must be taken into consideration when coding it. The website also gave tutorials on CSS and JavaScript.

The website was set up in a way that it would first teach about the different topics in each category of HTML, JavaScript, and CSS and then it would give a small coding test at the bottom to ensure that learning was taking place. The Quiz aspect at the end of each topic was very beneficial because it called upon the students recall ability to not only remember the information but to be able to implement it in the correct manner so that the little snippet of the code would work. Another aspect that was very helpful was the step-by-step method that the website laid out all the different facets of the topics. If you needed help with a specific aspect of HTML, JS or CSS, it was easy to access and learn just that specific category. Before coding, it is vital that a student should become very familiar with the three topics and be proficient in their implementation of them because all that knowledge is what will be used when building the user web interface for the lockbox.

The calculator that was built can be viewed below in Fig 6. When building the calculator, it was advised that it should resemble the windows calculator as much as possible and it should be able to perform the same operations.

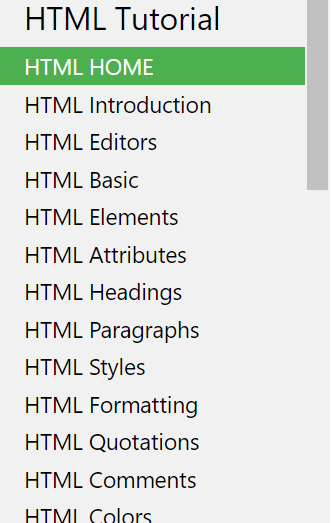
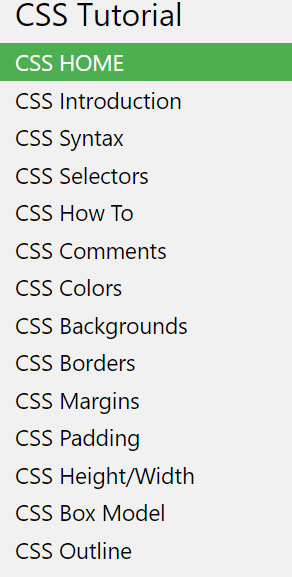
  

Figure Snippet of different topics in HTML,CSS and JavaScript[6]

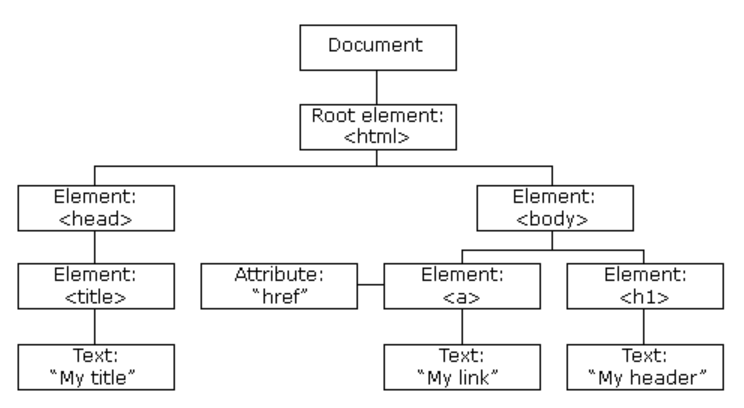


Figure 5 The HTML DOM Tree of Objects

[9]“A browser will create a Document Object Model everytime a webpage is loaded. The DOM is a platform that allows programs and scripts to dynamically access and modify the contents structure and style of a document. The DOM is a World Wide Web Consortium standard and can be separated into 3 different part:

* Core DOM - standard model for all document types
* XML DOM - standard model for XML documents
* HTML DOM - standard model for HTML documents”

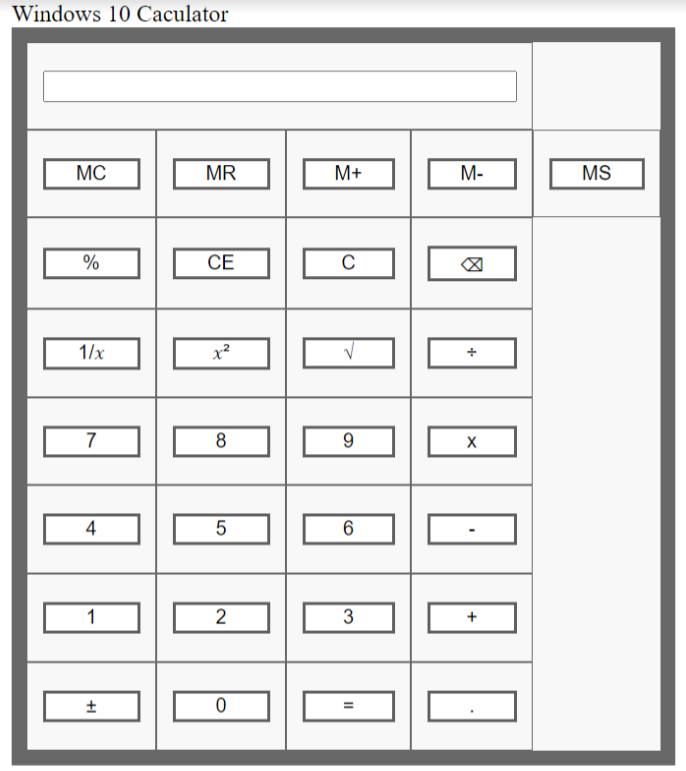


Figure 6 Snippet of calculator in chrome

Fig 6 displays the the calculator that was developed using the coding skills acquired form the website [www.w3schools.com](http://www.w3schools.com). The display is supposed to resemble the standard calculator on windows 10. The calculater above was created using this web technologies and implementing then on Visual Studio Code.

HTML: stands for Hypertext Markup Language. HTML is a standard for describing the structure of a web page and the content within it.

HTML components are the foundation of the designed HTML pages. HTML elements are defined by tags which are function semantics for text, such as heading, paragraphs, quotations, items, and lists. e.g., <h>, <p>. These tags have a purpose and text is placed in-between the tags. The tag must be placed twice in the text for the purpose of enclosing the allocated text within it.

CSS: stands for Cascading Style sheets and it is used to describe how the HTML elements will be displayed on the web page.

JavaScript: is used to create a function that is made to do a certain task. The function will be executed once it is called.

## **What is a Sensor and what do they do?**

[1]A sensor is a device with the capabilities of detecting change in the environment and responding with some output on the other system. A sensor converts the physical information that it detects into readable information for humans to understand.

There are many types of sensors and they all have their own purpose when it comes to the information that they are monitoring. Sensors are used in our everyday lives, such sensors could be a thermometer, a touch sensor, a gyroscope, or a smoke sensor.

The sensor that will be used in this project is a light sensor. Light sensors are photoelectric devices that can convert light energy to an electrical signal output. The light sensor will be incorporated into the lockbox by detecting if the lockbox is still opened or if it may have been tampered with.

There are two main categories of photoelectric devices. There are those which generate electricity when exposed to light and those which change their electrical properties in some way. Examples of photoelectric devices that generate electricity are photo-voltaics and photo- emissives.



Figure 7 light sensor on Groove Beginner kit

The light sensor that can be used to detect light in this project is integrated on the Grove Beginner kit board. Fig 7 displays the light sensor that will be used. The light sensor integrates a photo-resistor to detect the intensity of light. The photo-resistor is also known as an LDR. As the light intensity increases the resistance of the photo-resister decreases. The Operational Amplifier chip on the board produces voltage corresponding to the intensity of the light. The output signal is a represented as an analogue value. This analogue value can either increase or decrease depending on the amount of light intensity the sensor detects.

## **Socket.io**

[7] After all the research, the next task was to be able to build a web server that was able to tackle a client’s request and server data received via a serial port connection. The idea behind the use of Socket.io is that it allows the user to send and receive messages from a server.

The website socket.io was essential when learning how to execute the communication link between a client and a server. Fig 9 shows what it looks like in the HTML page. The messages in grey were written in one tab of localhost:3000 and the messages in white were written in another tab.

The tutorial teaches the user how to create a basic chat application. Sockets are the main solution for real-time chat systems because they are built in a way that allows for bi-directional communication between servers and clients. The Node.js web framework was used to set up a HTML webpage that would serve out a form and list of messages. The package.json manifest file will be used to describe the project and the index.js file will be used to set up the application. Npm install will be used for the purpose of populating the dependencies. The code to do that is “npm install express@4”.

The code used in index.js was set to listen to port 3000.

When the index.js is running, the screenshot in Fig 9 can be seen on the webpage and a hello world message is displayed on the webpage of port 3000.

The steps required before the user can send messages bi-directionally on the webpage are:

* Serve a HTML file called index.html. The file is created and served instead of sending the hello world message.

Text

Description automatically generated

Figure 8 Index.html

* Socket.io needs to be installed by inputting “npm install socket.io” and altering the index.js accordingly.
* Events must be emitted in a way that the server receives it as a chat message. The index.html will need to be altered to resemble the code that was presented from the website.
* Broadcasting is the final stage. In this stage the event needs to be emitted from the server to the other users. The io.emit() method is used to send the event to everyone.
* The snippet in Fig 9 is the result of completing the task provided by the socket.io website. A functioning chat messages.

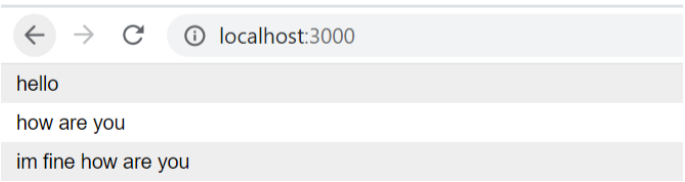


Figure 9 chat messages

## **Node.Js**

Node.js is an open-source server environment. Node.js uses JavaScript on the server and runs on multiple platforms. Node.js files tasks that will be executed on certain events. Also, it usually tries to do this through the use of a port on the server.

The example in fig 9 was the example that was given on the socket.io website. This example was used as a foundation and modified to produce a different variation of a chat room. The example in fig 9 and fig 10 show the new version of the chat room that was made. In this chat room, it notifies one user once the other user begins typing.

Graphical user interface, text, application, Teams

Description automatically generated

Figure 10 Modified chat program

Graphical user interface, text, application, email

Description automatically generated

Figure 11 Modified chat program.

## **Seeduino Lotus**

[14] The Seeduino Lotus board has a USB port, and this is what the board and the PC use to communicate data/information. Fig 13 shows how the Seeduino Lotus board is attached to the beginner kit. Seeduino Lotus board is an ATMEGA328 micro controller development board. The v1.0 board uses an Atmel ATMEGA328P-MU and a CH240. The CH240 is a USB bus converter chip. Its purpose is to realize a USB to serial interface. The Seeeduino Lotus is composed of 7 analog inputs/outputs, an ICSP header, 14 digital inputs/outputs, a micro-USB connection, a reset button and 12 Grove connection. In the case of the Project some of the digital pins were used to wire up the keyboard. Connectors were attached to both the pins on the keyboard and the designated digital pins. The board is fully compatible with Arduino UNO and was programmed using Arduino IDE (Integrated Development Environment). The board also does not require much power which aligns with the ethics of the project, using only 5V to power up the board.

A picture containing calendar

Description automatically generated

Figure 12 displays the Seeduino Lotus.

There are many different parts that make up the hardware of the Seeduino Lotus board.

* **Reset:** This button is used to allow the user to reset the Seeeduino board even when a shield is placed on top.
* **Grove Connectors:** SeeedStudio has an array of sensors/devices that could use the Digital, I2C, Analog and UART connection. Connecters come along with the Grove beginner kit that can be used to connect them with the sensors/devices.
* **LED-D13** An LED is connected to D13 pin of the board. This LED can be used in programs as a switch or whatever purpose the user wants to assign it.
* **Power Pins & Analog Pins:** The power connections are useful if you want to power up other sensor or devices on the breadboard.
* **ICSP**: ICSP is the connection for the ATmega328P. The SPI pins in this ports SCK, MOSI and MISO are also connected to digital pins 13,11 and 12.
* **USB 2 Uart** Pinout of USB-2-Uart. The pads could be used to communicate with other UART devices by putting the on-board ATmega328 in reset mode.
* **USB Input** USB Port is used to connect the board to the PC. This connection is used to power up the board or for programming purposes.

Graphical user interface, application

Description automatically generated

Figure 13 Hardware Overview of Grove Beginner Kit.

Fig 13 is the Grove beginner kit with different sensors, and output devices. They were used to simulate a functioning lockbox.

In Fig 13, it is possible to see the Buzzer and 3-axis accelerator that were used to detect if the board had been tampered with.

**Zigbee Device**

[10] Zigbee is a protocol used to connect smart devices to each other*. “*Zigbee includes a suite of IEEE 802.15.4-based specifications for communication protocols”. The Zigbee chip uses the IEEE 802.15.4 protocol over 2.4GHz. The range of Zigbee is less than that of Bluetooth and WI-FI, which contributes to its low power consumption. Radio transceivers are used by Zigbee smart devices to communicate. These smart devices can send and receive data from other smart devices running the Zigbee protocol. Zigbee provides a device-to-device type of communication. This type of communication can relay messages across larger ranges.

Useful advantages of using Zigbee protocol:

* Security: Zigbee devices are know for being very secure, this is aided using the 128-bit AES encryption keys.
* Connection to cloud: A hub is used to control multiple devices using apps. Zigbee is connected to internet via the smart hub.
* Power-Efficient: since it consumes a small amount of energy while in standby, this will help improve the overall battery life.
* Cost-Efficient- Zigbeee Devices are a more affordable than other devices with the same functions.

The Zigbee device that was supposed to be used for this project can be seen in Fig 14, but because of a change in circumstances, it was not included.

A picture containing text, electronics, circuit

Description automatically generated

Figure 14 Zigbee device

# **Design**

The concept of the project is to create a Lockbox that can be protected using a passcode sent from a web server to the client. The client will be able to enter the passcode into the keypad, automatically opening the lockbox so the key can be retrieved. There are many steps that must be taken to complete the operation. The Lockbox will also incorporate different tools such as a light sensor and output devices.

The Project is divided into three different stages:

* The Web Server stage
* The Web interface stage (web technologies)
* Coding in Arduino IDE

The knowledge developed researching Node.JS, HTML, JavaScript, CSS was used to design the webpage. The Lockbox system was designed in a way in which the web server can communicate directly to the Lockbox. The server is the main communication channel between the webpage and the Lockbox.

The premise is an employee of a real estate company will need to Login to the Lockbox website and then they need to select their desired Lockbox. The Lockbox will contain a key that will grant access to the property. The real estate agent is hosting a showcase for different houses to different people. Having a website that will provide them access to the Lockbox’s is more efficient than them always carrying many different keys with them.

The web page will have its own unique design and will have all its responsibilities and functions incorporated into this design. It took a few weeks to design the webpage but when it was done the webpage could display 3 different Lockbox’s and their unique passcodes.

Also on the webpage is a Login screen that transitions into the display of the passcodes if the user is successful. Ideally the user will need to input the correct password and username to be granted access to view the Lockbox’s and their passcodes. The Login button was designed with the purpose of validating if the inputted details were correct.

The Lockbox will not actually be a real-life lockbox but a more simulated one in which the Grove Beginner kit board will have functions that will mimic the operations of a functioning Lockbox. A keypad will be used so the user will be provided with a place to input the passcode that they receive from the web server. There will be three different switches that would be ideal to incorporate into the Lockbox:

* One switch will be for the key hook, this switch will be used for identifying if the key is still on the hook or if it has been removed by somebody.
* The second switch will be a door locked switch, which will allow the system to check if the door of the lock box is fully closed or still open. A light sensor will be incorporated with this switch and it will provide information as to if any light is being let into the lockbox. If light is entering the lockbox, it is evident that the person that used it last did not close it properly or it may have been tampered with it in a way where the lockbox has been left open ajar.
* The third switch will be monitoring the location of the Lockbox. If the lockbox location changes this will be treated as if the Lockbox has been removed from the position it was placed on the wall. This will affect the state of the tamper switch.

Creating a layout of how the Grove Beginner kit board will be setup is very essential to the project.

Ideally the Lockbox’s will remain in a sleeping state and will only be activated by the pressing of buttons on a keypad, but this was not implemented.

## **Web Server**

The Web server is the link that will allow communication of data from the sensors and output devices. The web socket technology used in this project is socket.io. Socket.io is used in this project to create the bidirectional communication link between the client and the server. The web server in this project will be listening to a port in which data can be transmitted across a serial connection. The message listening to port 4000 is printed on the Debug Console. The message Serial Port open was placed on the Debug Console, so that we know when if the port is open to transmitting data. If the port is not open a different message will be sent to the console.

If the port is unsuccessfully opened, an error message is to be printed to the Debug Console.

The passcode data from the website is the type of information that could be transmitted along the serial port in this project.

[11] The type of web server used is the **HTTP web server. This server can serve up content using the HTTP protocol. The content generated can be made on the fly. It is also known as dynamic contents/pre-existing content. This type of server usually serves up content in the form of web resources, images, or HTML documents. Fig 15 is a representation of the type of server that could be used in this project.**

Diagram

Description automatically generated

Figure 15 Workflow of Apache web server [12]

The webpage sends the designated passcode across the serial link to Grove beginner kit. Although the passcode is not displayed on the screen, it is stored in a character array and the passcode inputted using the keypad gets compared to the passcode that was transmitted over. The passcode that is transmitted over gets inserted into an array called Master and the passcode inputted by the keypad is stored within a character array called Data. If array Master and Data are the exact same the user will be granted access to the Lockbox. Specific data such as “door opened”, “door closed” and the state of the door will be transmitted back across the serial link.

## **Web Interface**

Creating the Web interface requires the use of all the web technologies that was studies throughout the duration of the project. HTML, CSS and JavaScript were combined together to construct a webpage for the Lock box. A Login page and a welcome page will be created. The welcome page will also contain the Lockboxes and their passcodes. The design of the webpage should only show that there are 3 Lockboxes, but not the passcode until a specific lockbox is selected. Clicking the select button should display the passcode on the webpage and transmit the character of the passcode along the server to the Arduino so that it can be stored in the Master array. This is what could ideally happen. Ideally the Login page is supposed to have a username and password set so that not just anybody can access it. Also, the webpage was to be designed in a way in which every 2 hours the passcode for each Lockbox would change. The purpose of changing the passcode every 2 hours was to fortify the security of the passcodes. The ideal webpage would also have multiple firewalls in place to prevent hackers from accessing the information. The passcodes are only supposed to be 4 digits long.

## **Arduino IDE**

[13] Arduino IDE is a open source software that s used to write code and upload it onto a board. Since, the Grove beginner kit is compatible with this software. The software can be used for any type of Arduino board. It is providing access to a multitude of different libraries.

Icon

Description automatically generated

# 

# **Implementation.**

As discussed in the Design section of the project, that are a multitude of things that could have been implemented in the project that would have improved its overall quality. This section will discuss the actual parts that can be found in this project.

The Construction of the webpage took priority, and this is where everything started. Visual Studio code was used to combine all the web technologies and run it on the browser. The first aspect created was the Login page. The purpose of the Login page was so that the real-estate agent would arrive at the house with the Lockbox and would need to access the website through their phone to receive the passcode. The username and password were supposed to be specific so not just anybody could access the website. In my project although there is a Login screen and there is a function called validate, Anybody can Login. The part of the code does not work.

Text

Description automatically generated

Figure 16 Validate Function.

This is the validate function. If It is supposed to only allow you to login if the username is pelumi and the password is abcde.

Text

Description automatically generated

Figure 17 Onclick calling validate function.

Fig 17 is supposed to call the validate function. Although the validate function does not work, it is still possible to input values into the username and password boxes. If the Login button is clicked, it transitions into the welcome.html.

Text

Description automatically generated

Figure 18 Webpage HTML

The body is composed of the Title that is placed in the centre and the username and password are also placed in div classes called form control. <div> is a tag that defines a section of the HTML document. There is also a <div> that contains the information for the button. All this information is enclosed in a <body>.

<body>: the body is a container that hold all the visible elements such as the <div>, headers and lists. The body defines the body of the document.

<title>: This tag is just used to title the HTML page.

Text

Description automatically generated

Figure 19 Tags contained in the head tag.

In-between the tag <head> was the script that contained the validate function. Also, in the head tag is the <style> tag. CSS is used to create a format for how the webpage will look.

Text

Description automatically generated

Figure 20 Snippet of a bit of the code in the style tag.

The code written in the webpage.html created the Login Screen seen in fig 21.

Graphical user interface, text, application

Description automatically generated

Figure 21 Login Screen

After the first HTML was of the webpage was created, next was the layout of the welcome page. The welcome webpage displays a “Welcome! Successful Login” message in the top centre of the page. Ideally the welcome page should not display all the passcodes of the three lockboxes, but in this case that was not implemented. There are three lockboxes and each of them has a different passcode. The code for the select button has the function of transmitting the data of the passcode through the web server to the Lockbox. This way the Lockbox will have a reference of characters that must be matched by the user using the keypad. The welcome.html also contains a variable that keeps track of the state of the Door. Before anything happens, the door state has a variable of A. This value will change to 1 when the Door is opened or 0 when the door of the lockbox is closed. Fig 22 shows the code used to construct the HTML page for the welcome webpage.

Text

Description automatically generated

Figure 22 Code for Welcome.html

Within the script tag is the function that defines what occurs when each select button is clicked. When the buttons are clicked, the debug console emits the “setPW” with the passcode that is tied to that button. The ready function also contains a message handler, so when the socket listens for door state update message it knows to change the door value.

Text

Description automatically generated

Figure Contents of the script tag in the welcome.html

Fig 25 and Fig 26 displays the two transitions of html pages. The code is run using the Node.js. The port used is port 4000.

Graphical user interface, text, application

Description automatically generated

Figure 24 Debug Console of port opened and listening.

Graphical user interface, application

Description automatically generated

Figure 25 Lockbox Login Webpage

Graphical user interface, text, application

Description automatically generated

Figure 26 Welcome webpage.

**Web Server**

The Last piece of code in the file is the JS code named webpage.js. A Node.js server framework called express.js was used as the framework for constructing a server that allows for communication between the web client and server. As seen in fig 24 the server listens to requests from port 4000.

Text

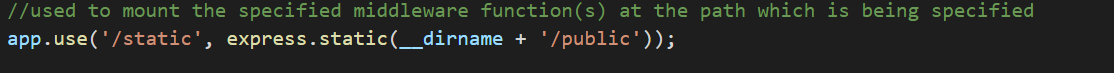
Description automatically generated

Figure 27 Snippet of webpage.js

As is shown in the Snippet of webpage.js the app.get routes the HTTP get request to the path specified which is webpage.html. When the program is being run on Node.js, it starts of by displaying webpage.html. Socket.io is set up in the JS. Its function is to allow for bidirectional communication between the client and the server. The var. io variable holds the web socket socket.io.

App.use is used to mount the specified middleware function on the ‘/public’ path. This means it will investigate the public folder when the login button is pressed. The webpage.html must perform this action when that button is pressed.



Snippet of webpage.js

Text

Description automatically generated

Snippet of webpage.js

The function above is a Callback function. The function is listening for the event called connection. This occurs when a connection is made between the client and the server. Within it the function is also listening for when the connection gets disconnected.

The setPW function’s purpose is to look for the setPW message along with the string attached to it. This string is where the characters of the passcode will be inserted. The string part of the message will be written to the port. The port.write writes data to the serial port of the output buffer.

As soon as it gets this information it then checks the string and if it is what is required it will print out the “message written” to the Debug Console. The error message will return ‘Error on port write’ if the string is not written to the port.

Text

Description automatically generated

The next function is a data handler. It receives data from the serial port via the parser. The data it is reading from the board gets converted to a string and is the printed onto the console. The data it is looking for is the message ‘door opened’ and ‘door closed’. Once it reads the door opened message it then changes the state of the door variable on the webpage to ‘1’. Else, the door closed message changes the state of the door to ‘0’. There is a section on the welcome.html where the state of the door is displayed in real time. The emit purpose is to call the registered message handler in the welcome.html so it gets the message to update the door value.

Text

Description automatically generated

Snippet of webpage.js

The rest of the code in the webpage.js are just functions that are checking the port and printing out ‘port closed’ to the console if that is the case or the error message.

There is also the http.listen function that is constantly listening to the specified port 4000 for connections/information.

## **Arduino Code**

Graphical user interface, text, application, email

Description automatically generated

Figure 28 Snippets of Ardunio code

Text

Description automatically generated with medium confidence

Snippet of Arduino code

Variables are set as outputs, inputs and Input pullups. Also, in fig 28 is the setup of the 3-axis accelerator, the serial monitor and other parts. These parts are setup in here so they can be used in other parts of the code.

Graphical user interface, text

Description automatically generated

Snippet of Arduino code

* In the code snippet above the X and Y locations of the board are being read and then the code enters an if statement
* The if statement just checks if the x and y values are greater than 0.15. If so, the buzzer will output a certain tone and “Lockbox tamper” message will be printed to the OLED display.
* The next if checks for **stringComplete**, it will then trims the \n off the end of the string.
* If the string starts with **setPW**: its prints the **inputString** + the received message to the Serial monitor. In this section the code is taking in the data sent from the webpage. The data that is being sent is the passcode. That is why the passcode gets copied to the Master array. Its only copies the values starting from the 7th position in the **inputString**.
* The **inputString** then gets cleared and the **stringComplete** value is assign the value false.
* **Char c** is initialized with the scan function. The scan function is how keys are read from the keyboard.

Text

Description automatically generated

* The if statement checks if the count value is 0 and when its 0 it will print Enter Password to the OLED display.
* Next, another if statement checks if **c** is not equal to \0 and while **scan ()** is not equal to \0, do nothing.
* Else it reads the values pressed from the keypad and places it in the **Data** array.
* While the value is being placed in the **Data** array, they are also being printed on the OLED display so the user can see the keys they press. The count is then incremented by 1. The count is incremented by 1 every time a key is pressed. The key pressed is shifting position towards the write using count.
* Once the count is equal to the password length the \0 is assigned to the **Data** array with the password length in it.
* The buffer gets cleared, after that the strings of **Master** and **Data** array get compared to each other.

Text

Description automatically generated

* If **Data** and **Master** are not equal the message “Password Incorrect” is printed on the OLED display.
* If **Data** and **Master** are equal the LED on pin 4 is turned on and the message door opened is displayed on the OLED is displayed for 5 seconds. While the light sensor reads a value greater than 100 it will remain in the door opened state.
* Once the light sensor value goes below 100, door closed is placed on the OLED display.
* The message door closed not being printed shows that the amount of light being let in is too much so that must mean the Lockbox is still open.
* When the Lockbox is open, the user can take the key out of the box. The lockbox door being opened and closed will send data to the webpage.
* There is a clear data function at the end of the loop that clears the string in the Data array.

The clear data function and the scan function can be found in the appendix.

A picture containing text, electronics

Description automatically generated

Figure 29 Physical Equipment used for Lockbox Simulation.

# **Running Code**

1.Text

Description automatically generated

The code starts by listening to the port then when a passcode is selected it outputs the “message written” and “received” to the console.

2.A picture containing text, electronics, circuit

Description automatically generated

The user then inputs the passcode using the Keypad.

3.A picture containing text, electronics, circuit

Description automatically generated

If the passcode is correct the door is opened and the LED lights up

4. A picture containing text, electronics

Description automatically generated

Once the door is closed the LED turn off. The door can only be closed if the light sensor reads a value 100 or lower.

5. A close-up of a computer chip

Description automatically generated with low confidence

If the board gets moved from past the restricted x and y locations, the buzzer turns on and the Lockbox tamper message is printed to the OLED display, so the display wont read the keys pressed anymore.

Also if the password is wrong the message “Incorrect Password” is printed on the OLED display.

# **Future Modification**

Although this system was able to simulate a functioning Lockbox, there are many other improvements and modification that can be made to the overall project.

* Bluetooth: One way in which the system can be improved is by adding in a Bluetooth operation. Instead of the user having to physically input the passcode into the keypad, they could instead be able to unlock the Lockbox using an app on their phone. This would remove the use of the keypad.
* Another modification for the project that was not implemented is a switch for the key. The switch will be able to detect whether the key is still in the box. The program only opens and closes the door but is unable to tell if the key is still in the Lockbox.
* Also, maybe the system could be programmed to require a different passcode when opening the lockbox again after the key has been used. So, when the user wants to put the key back, they would need a different password to open the Lockbox again.
* In the welcome webpage, for security reason, the other passcodes of Lockboxes 2 and 3 would remain hidden until they were selected.
* The use of the OLED display could be improved by having a standby mode. So, in standby mode the display is empty until a passcode is sent down.
* The security for the Lockbox website could be improved by having a few firewalls or safety questions in case a user entered the wrong password multiple times.
* An easier method of opening the Lockbox could be by integrating a fingerprint scanner into the Lockbox. That way there would not be a keypad needed. The system would just contain all the information of every employee’s fingerprint.

# **Bill of Materials**

|  |  |  |  |
| --- | --- | --- | --- |
| Component | Quantity | Supplier | Price |
| Grove Beginner kit | 1 | SeedStudio.com | $19.90 |
| EOZ IP40 12 Key Keypad | 1 | RS | € 13,32  (excl. BTW) |

The EOZ IP40 12 Key Keypad is display in fig 1.3. When acquiring the keypad, it is also essential to have the datasheet because this will come in very handy when wiring up the Keypad to the digital input pins of the Seeduino Lotus board.

Table

Description automatically generated with medium confidence

Figure 30 Snippet of Datasheet of Keypad [4]

The snippet of the Datasheet shown in fig 28 is probably the most important aspect of the entire datasheet because it is necessary to know how the contacts are inserted into the keypad. This information is not on the board itself. The pins used on the Seeduino board are pins 3,7,8,9,10,11 and 12.

# **Poster Section**

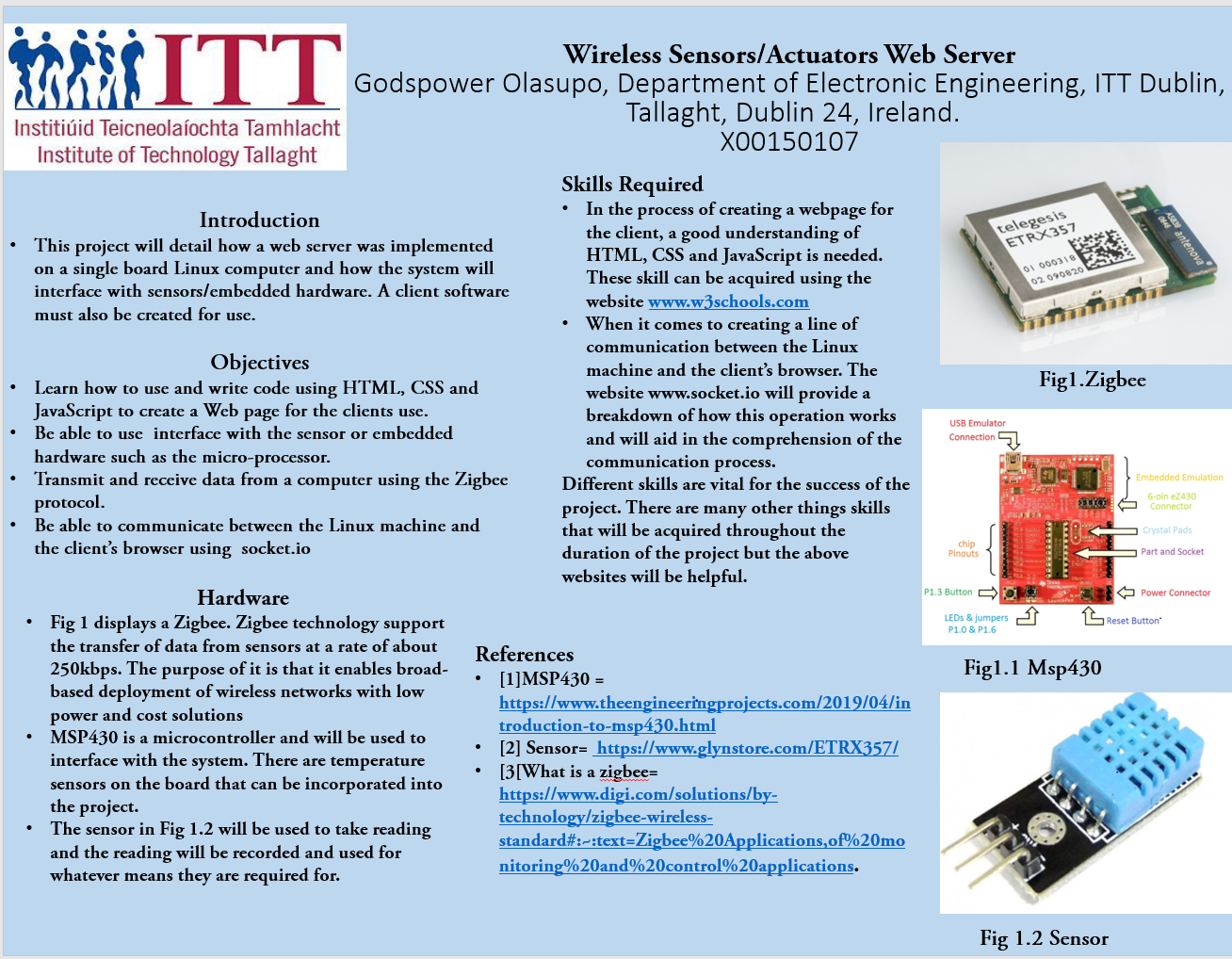


Figure 31 Semester 1 poster

The poster in Fig 29 was created using PowerPoint for the purpose of giving a short and concise amount of information about the project. Posters about the projects are usually made to display to students that come to visit the college on open days. The students receive a bit of information about the type of projects that take place in the University. The poster was also very handy when it comes to recalling information about the main premise of the project.

# **Environment and Ethics**

Engineers should always take into consideration how their work can affect the environment around them. So, there are rules in place that must be obeyed by every engineer. One of the most important rules for an engineer is to not infringe on the safety of other people. The engineering society also has ethical rules that must be adhered to. These ethical rules tend to be the rules of the Land. Different countries could have completely different code of ethics Below are some rules engineers should adhere to. Ideally the system would remain in standby mode and must be touched to wake it up. This is how the energy consumption could be decreased.

Environment requirements:

* Improve the efficiency of designs for circuits thus reducing the number of components used.
* Be very cost aware of materials needed thus minimizing the cost.
* The design and production should consume as little power as possible.
* All waste from electrical designs or projects should be of disposed accordingly.

Ethical requirements:

* The main priority of any project/design should be to maintain the safety and welfare of the public.
* Before undertaking a task, ensure that they are competent in that field.
* Public statements should be issued in a truthful objective manner.
* When creating designs, ensure the design is suitable for all users.
* Plagiarism should never be found in your work and information obtained from another source must be referenced.
* Ensure that the Standards of Comreg are applicable to your project, as they are the governing body for engineers in Ireland.
* Conduct work in a health and safe manner. E.g., maintain a tidy work environment. Understand the correct use of equipment.
* Licencing and copyrights should be respected.
* Waste should be disposed of according to the WEEE standard.

# **Conclusion**

The main goal of this Project was achieved in the end, which was to create a House Key Lockbox. A bidirectional link was created between the client and the server, so now the user can send a passcode to the Lockbox and verify the passcode by inputting the correct 4-digit number into the keypad. Once the correct password confirmed the user will be able to open the Lockbox and use the key in it to access the Home. The Lockbox system was not able to be built physically but it can be simulated. The system will be of use to real estate agents when they are showcasing the homes to potential buyers. They will be able to access the Lockbox webpage from their phones.

The Project discusses in detail all the different aspects that came together to create the functioning Lockbox. Aspects such as the web technologies used to create the webpage, the web server and the coding made in Arduino IDE. The culmination of all these facets was able to create this operating system.

All in all, the objectives of the project were met, and a functioning prototype was able to be simulated. So the project was a success.

# **References and Bibliography**

* [1]"What Is A Sensor and What Does it Do? | Dewesoft", *Dewesoft.com*, 2021. [Online]. Available: https://dewesoft.com/daq/what-is-a-sensor. [Accessed: 26- Jan- 2021].
* [2]"Client-Server Web Apps with JavaScript and Java", *O’Reilly Online Learning*, 2021. [Online]. Available: https://www.oreilly.com/library/view/client-server-web-apps/9781449369323/ch01.html. [Accessed: 26- Jan- 2021].
* [3]2021. [Online]. Available: https://www.electronics-tutorials.ws/io/io\_4.html. [Accessed: 26- Jan- 2021].
* [4] *Docs.rs-online.com*, 2021. [Online]. Available: https://docs.rs-online.com/f361/0900766b80de0aff.pdf. [Accessed: 03- May- 2021].
* [5] B. Zuo, "Grove - Light Sensor - Seeed Wiki", *Wiki.seeedstudio.com*, 2021. [Online]. Available: https://wiki.seeedstudio.com/Grove-Light\_Sensor/. [Accessed: 26- Jan- 2021].
* [6]"W3Schools Online Web Tutorials", *W3schools.com*, 2021. [Online]. Available: https://www.w3schools.com/. [Accessed: 26- Jan- 2021].
* [7]"Get started", *Socket.IO*, 2021. [Online]. Available: https://socket.io/get-started/chat/. [Accessed: 26- Jan- 2021].
* [8]"ECO.12150.06 | EOZ IP40 12 Key Keypad | RS Components", *Nl.rs-online.com*, 2021. [Online]. Available: https://nl.rs-online.com/web/p/keypads/0146014/. [Accessed: 03- May- 2021].
* [[9](https://www.w3schools.com/js/js_htmldom.asp%20%5b9)]"JavaScript HTML DOM", *W3schools.com*, 2021. [Online]. Available: https://www.w3schools.com/js/js\_htmldom.asp. [Accessed: 03- May- 2021].
* [10]"What is Zigbee? Explaining the World’s Most Popular Smart Light Network Technology", *Homey.app*, 2021. [Online]. Available: https://homey.app/en-us/wiki/what-is-zigbee/. [Accessed: 03- May- 2021].
* [11]"What is a Web Server?", *InforamtionQ.com*, 2021. [Online]. Available: https://www.informationq.com/what-is-a-web-server/. [Accessed: 03- May- 2021].
* [12]"Introduction to Apache Web Server", *Medium*, 2021. [Online]. Available: https://medium.com/@jovanshernandez/introduction-to-apache-web-server-2138af643039. [Accessed: 03- May- 2021].
* [13]2021. [Online]. Available: https://www.arduino.cc/en/software. [Accessed: 03- May- 2021].
* [14]B. Zuo, "Seeeduino Lotus - Seeed Wiki", *Wiki.seeedstudio.com*, 2021. [Online]. Available: https://wiki.seeedstudio.com/Seeeduino\_Lotus/. [Accessed: 04- May- 2021].

# **Appendix**

Webpage.js

var express =require("express"); //requires express app

//Setup of express app

var app = express();

var http = require('http').Server(app);

var io = require('socket.io')(http);//sets up socket.io

var SerialPort = require('serialport');

var Readline = require('@serialport/parser-readline');

//checks if port is successfully opened

var port = new SerialPort('COM3', {baudRate: 9600}, function (err) {

  if (err) {

    return console.log('Error on port open: ', err.message);

  }

  console.log('Serial Port Open');

});

var parser = port.pipe(new Readline());  // seperate data input using newlines

var state = 0;

// routes the HTTP GET Requests to the path which is being specified

app.get('/', function(req, res){

  res.sendFile(\_\_dirname + '/webpage.html');

});

//used to mount the specified middleware function(s) at the path which is being specified

app.use('/static', express.static(\_\_dirname + '/public'));

//This code will notify when the user connects, disconnects or writes a message

io.on('connection', function(socket){

  socket.on('disconnect', function(){

  io.emit('nClientUpdate', {});//emit to webpage

  });

  //if the message wriiten starts with setPW and then the string,

  socket.on('setPW',function(msg){

    var str = 'setPW:'+msg.PW.toString()+'\n';

    port.write(str, function(err) {

     // outputs message wriiten to the console or it outputs the error message

    if (err) {

     return console.log('Error on port write: ', err.message);

    }

    console.log('message written');

   });

  });

  io.emit('nClientUpdate', {});//emit to webpage

});

//Data handler for data received from Serialport via parser

parser.on('data', function(data) {

  var dataStr = data.toString();

    console.log(dataStr);

  if(dataStr == 'Door opened\r'){

    state=1;

  }else if(dataStr == 'Door closed\r'){

    state=0;

  }

  io.emit('DoorStateUpdate', {door:state});

});

//is called when port gets closed

port.on('close', function() {

  console.log('Port closed');

})

//called if there is an eror on the port

port.on('error', function(err) {

  console.log('Error on port: ', err.message);

})

//Starts the HTTP server listening for connections.

http.listen(4000, function(){

  console.log('listening on \*:4000');

});

**Webpage.html**

<!DOCTYPE html>

<html lang="en">

    <head>

        <meta charset="UTF-8">

        <meta name="viewport" cotent="width=device-width, initial-scale=1.0">

        <title>Lockox Login Webpage</title>

        <script>

         function validate(){

            var password=document.getElementsByid("pass").value;

             var username=document.getElementsByid("Uname").value;

             if(username== "pelumi" && password== "abcde"){

             return true;

             }else{

             return false;

  }

}     </script>

        <style>

            \*{

                margin: 0;

                padding: 0;

                box-sizing: border-box;

            }

            body{

                min-height: 100vh ;

                background:  #eee;

                display: flex;

                font-family: sans-serif;

            }

            .container{

                margin: auto;

                width: 500px;

                max-width: 100%;

            }

            .container form{

                width: 100%;

                height: 100%;

                padding: 20px;

                background: rgb(248, 247, 247);

                border-radius: 4px;

                box-shadow:0 8px 16px rgba(0, 0, 0, .3);

            }

            .container form h1{

                text-align: center;

                margin-bottom: 24px;

                color: #222 ;

            }

            .container form .form-control{

                width: 100%;

                height: 40px;

                background: rgb(252, 251, 251);

                border-radius: 4px;

                border: 1px solid rgb(46, 153, 10) ;

                margin: 10px 0 18px 0;

                padding: 0 12px;

            }

            .container form .btn{

                margin-left: 35%;

                transform: translateX();

                width: 130px;

                height: 35px;

                cursor: pointer;

                background: black;

                color: white;

            }

            .container form .btn:hover{

                opacity: .7;

            }

        </style>

    </head>

    <body>

        <div class="container">

            <form class="myForm" action="static/welcome.html">

                <h1> Lockox Login</h1>

                <div class="form-group">

                    <label for="">Username:</label>

                    <input type="username:" name="Uname" id="Uname" class="form-control" required>

                </div>

                <div class="form-group">

                    <label for="">Password:</label>

                    <input type="password" name="Pass" id="Pass" class="form-control" required>

                </div>

                <input type="submit" class="btn" value="Login" onclick=validate()>

            </form>

        </div>

    </body>

</html>

**welcome.html**

<!DOCTYPE HTML>

<html lang="en">

<head>

<title>Welcome Page</title>

<script src="/socket.io/socket.io.js"></script>

    <script src="http://code.jquery.com/jquery-1.11.1.js"></script>

    <script>

        $(document).ready(function () {

            //connect using socket.io

            var socket = io();

            $("#but1").click(function(){

                //send setPW+the strinng after when button pressed

                socket.emit('setPW',{PW:$("#lb1").html()});

            });

            $("#but2").click(function(){

                socket.emit('setPW',{PW:$("#lb2").html()});

            });

            $("#but3").click(function(){

                socket.emit('setPW',{PW:$("#lb3").html()});

            });

            //register message handlers

            socket.on('DoorStateUpdate', function(msg){

                $("#doorValue").html(msg.door);

            });

        });

</script>

</head>

<body>

<center>

 <h1> Welcome! Successful Login</h1>

 <h2>Choose the Lockbox you want to open</h2>

 <p><span id="lb1">1257</span> Lockbox key 1:</p>

<button type="button" id="but1">Select</button>

<p><span id="lb2">7517</span> Lockbox key 2:</p>

<button type="button" id="but2">Select</button>

<p><span id="lb3">5428</span> Lockbox key 3:</p>

<button type="button" id="but3">Select</button>

<p><span id="doorValue">A</span> Door state</p>

</center>

</body>

</html>

**Arduino Code**

/\*import libraries for oled display\*/

#include <Arduino.h>

#include <U8g2lib.h>

#ifdef U8X8\_HAVE\_HW\_SPI

#include <SPI.h>

#endif

#ifdef U8X8\_HAVE\_HW\_I2C

#include <Wire.h>

#endif

U8G2\_SSD1306\_128X64\_NONAME\_F\_HW\_I2C u8g2**(**U8G2\_R0**,** /\* clock=\*/ SCL**,** /\* data=\*/ SDA**,** /\* reset=\*/ U8X8\_PIN\_NONE**);** // High speed I2C

/\*import for the 3-axis accelerometer\*/

#include "LIS3DHTR.h"

#ifdef SOFTWAREWIRE

#include <SoftwareWire.h>

SoftwareWire myWire**(**3**,** 2**);**

LIS3DHTR**<**SoftwareWire**>** LIS**;** //Software I2C

#define WIRE myWire

#else

#include <Wire.h>

LIS3DHTR**<**TwoWire**>** LIS**;** //Hardware I2C

#define WIRE Wire

#endif

/\*Definations of contacts to digital pins\*/

#define x1 3

#define x2 7

#define x3 8

#define y4 9

#define y3 10

#define y2 11

#define y1 12

/\*initialised variables\*/

#define Password\_Length 4

String inputString **=** ""**;** // a string to hold incoming data

boolean stringComplete **=** **false;** // whether the string is complete

char Data**[**Password\_Length**+**1**];**

char Master**[**Password\_Length**+**1**];**

int sensorpin **=** A6**;** // Analog input pin that the sensor is attached to

int outputValue **=** 0**;** // value output to the PWM (analog out)

const int LockOutout **=** 4**;**//initialise LockOutput to LED D4

int count **=** 0**;**

char c**;**

int BuzzerPin **=** 5**;**

void setup**(**void**){**

Serial**.**begin**(**9600**);**

u8g2**.**begin**();**

u8g2**.**enableUTF8Print**();**

inputString**.**reserve**(**200**);**

pinMode**(**LockOutout**,**OUTPUT**);** // initialize the LED pin as an output:

pinMode**(**BuzzerPin**,** OUTPUT**);** // initialize the buzzerpin pin as an output:

pinMode**(**sensorpin**,** INPUT**);**// // initialize light sensor as an input:

**while** **(!**Serial**)** **{};**

LIS**.**begin**(**WIRE**,** 0x19**);** //IIC init

//LIS.begin(0x19);

delay**(**100**);**

LIS**.**setOutputDataRate**(**LIS3DHTR\_DATARATE\_50HZ**);**

pinMode**(**x1**,**INPUT\_PULLUP**);**//set x1 as input pullup

pinMode**(**x2**,**INPUT\_PULLUP**);**//set x2 as input pullup

pinMode**(**x3**,**INPUT\_PULLUP**);**//set x3 as input pullup

pinMode**(**y4**,**INPUT\_PULLUP**);**//set y4 as input pullup

pinMode**(**y3**,**INPUT\_PULLUP**);**//set y3 as input pullup

pinMode**(**y2**,**INPUT\_PULLUP**);**//set y2 as input pullup

pinMode**(**y1**,**INPUT\_PULLUP**);**//set y1 as input pullup

**}**

void loop**(**void**){**

LIS**.**getAccelerationX**();**//get location of accelerationsX

LIS**.**getAccelerationY**();**//get location of accelerationY

delay**(**500**);**//delay for half a second

/\*if location y is greater than 0.15 or location x is greater than 0.15\*/

**if** **(**LIS**.**getAccelerationY**()>**0.15**||**LIS**.**getAccelerationX**()>**0.15**){**

analogWrite**(**BuzzerPin**,** 128**);**//write tone 128 to the buzzerpin

u8g2**.**print**(**"Lockbox Tamper!"**);**//display Lockbox tamper on OLED display

delay**(**1000**);**//pause for 1 second

analogWrite**(**BuzzerPin**,** 0**);**//write no tone to buzzer

delay**(**0**);**

**}**

**if** **(**stringComplete**)** **{**//check if for stringComplete

inputString**.**trim**();** //trim '\n'

**if(**inputString**.**startsWith**(**"setPW:"**))** //if inputstring starts with setPW:

**{**

Serial**.**println**(**inputString**+**" received"**);** //print the string plus received

strcpy**(**Master**,**inputString**.**c\_str**()+**6**);** //copy the passcode string into master array

**}**

// clear the string:

inputString **=** ""**;**

stringComplete **=** **false;**//change stringComplete to false

**}**

char c **=** scan**();** //call scan function as save to c

//check if count equals 0,print Enter passcode on the oled display

**if(**count**==**0**){**

u8g2**.**setCursor**(**0**,**10**);**//print at this location of oled display

u8g2**.**setFont**(**u8g2\_font\_ncenB08\_tr**);**

u8g2**.**print**(**"Enter Passcode:"**);**//print this string to oled display

u8g2**.**setCursor**(**0**,**20**);**//print at this location of oled display

u8g2**.**sendBuffer**();**

**}**

**if** **(**c**!=** '\0'**){**//if c is not equal to '\0'

**while(**scan**()!=** '\0'**){**

//do nothing

**}**

Data**[**count**]** **=** c**;**//store the input of keyboard in data array

//u8g2.setCursor(count,20);

Serial**.**println**(**c**);** // print keys that are pressed to serial monitor

u8g2**.**print**(** Data**[**count**]);**//print the keys pressed on the oled display

u8g2**.**sendBuffer**();** //send to buffer

count**++;**//increment count

**}**

**if(**count **==** Password\_Length**){** //check if count is equal to 4

Data**[**Password\_Length**]** **=**'\0'**;** //make '\0' equal to data array

u8g2**.**clearBuffer**();** //clear buffer

//compares to checks if password is corrrect

**if(!**strcmp**(**Data**,**Master**)){** //compare string in data to master

u8g2**.**setCursor**(**0**,**10**);**//print at this location of oled display

digitalWrite**(**LockOutout**,**HIGH**);** //turn led on of pin 4

u8g2**.**setCursor**(**0**,**10**);** //print at this location of oled display

u8g2**.**print**(**"Door opened"**);** //print this string to oled display

Serial**.**println**(**"Door opened"**);**//print this string to serial monitor

u8g2**.**sendBuffer**();**//send to buffer

delay**(**5000**);**

**while(**analogRead**(**sensorpin**)>**100**){**//causes it to remain in the door opened state

**}**

digitalWrite**(**LockOutout**,**LOW**);** //turn off led on pin 4

u8g2**.**setCursor**(**0**,**10**);**//print at this location of oled display

u8g2**.**print**(**"Door closed"**);**//print this string to oled display

Serial**.**println**(**"Door closed"**);** //print this string to serial monitor

u8g2**.**sendBuffer**();** //send to buffer

**}else{**

u8g2**.**setCursor**(**0**,**10**);**

u8g2**.**print**(**"Incorrect Passcode"**);**//print this string to oled display

u8g2**.**sendBuffer**();** //send to buffer

delay**(**2000**);**

**}**

clearData**();**

**}**

**}**

/\*clear data array\*/

void clearData**(){**

//clears data and changes all values in the array to 0

**while(**count**!=**0**){**

Data**[**count**-**1**]=**0**;**

**}**

**return;**

**}**

void serialEvent**()** **{**

**while** **(**Serial**.**available**())** **{**

// get the new byte:

char inChar **=** **(**char**)**Serial**.**read**();**

// add it to the inputString:

inputString **+=** inChar**;**

// if the incoming character is a newline, set a flag

// so the main loop can do something about it:

**if** **(**inChar **==** '\n'**)** **{**

stringComplete **=** **true;**

**}**

**}**

**}**

/\*scan function that read the keys pressed on the keypad\*/

char scan**(**void**)**

**{**

/\*MAKE X1 LOW\*/

pinMode**(**x1**,**INPUT**);**//set as input

pinMode**(**x1**,**OUTPUT**);**//set as output

digitalWrite**(**x1**,**LOW**);**

/\*if statement that reads the number pressed on the keypad\*/

**if** **(**digitalRead**(**y1**)** **==** LOW**){**

**return** **(**'1'**);**

**}else** **if** **(**digitalRead**(**y2**)** **==** LOW**){**

**return** **(**'4'**);**

**}else** **if** **(**digitalRead**(**y3**)** **==** LOW**){**

**return** **(**'7'**);**

**}else** **if** **(**digitalRead**(**y4**)** **==** LOW**){**

**return** **(**'\*'**);**

**}**

/\*MAKE X2 LOW\*/

pinMode**(**x1**,**INPUT\_PULLUP**);**//sets x1 back as a input pullup

pinMode**(**x2**,**INPUT**);**//set as input

pinMode**(**x2**,**OUTPUT**);**//set as output

digitalWrite**(**x2**,**LOW**);**

/\*if statement that reads the number pressed on the keypad\*/

**if** **(**digitalRead**(**y1**)** **==** LOW**){**

**return** **(**'2'**);**

**}else** **if** **(**digitalRead**(**y2**)** **==** LOW**){**

**return** **(**'5'**);**

**}else** **if** **(**digitalRead**(**y3**)** **==** LOW**){**

**return** **(**'8'**);**

**}else** **if** **(**digitalRead**(**y4**)** **==** LOW**){**

**return** **(**'0'**);**

**}**

/\*MAKE X3 LOW\*/

pinMode**(**x2**,**INPUT\_PULLUP**);**//sets x2 back as a input pullup

pinMode**(**x3**,**INPUT**);**//set as input

pinMode**(**x3**,**OUTPUT**);**//set as output

digitalWrite**(**x3**,**LOW**);**

/\*if statement that reads the number pressed on the keypad\*/

**if** **(**digitalRead**(**y1**)** **==** LOW**){**

**return** **(**'3'**);**

**}else** **if** **(**digitalRead**(**y2**)** **==** LOW**){**

**return** **(**'6'**);**

**}else** **if** **(**digitalRead**(**y3**)** **==** LOW**){**

**return** **(**'9'**);**

**}else** **if** **(**digitalRead**(**y4**)** **==** LOW**){**

**return** **(**'#'**);**

**}**

**return** **(**'\0'**);**

**}**