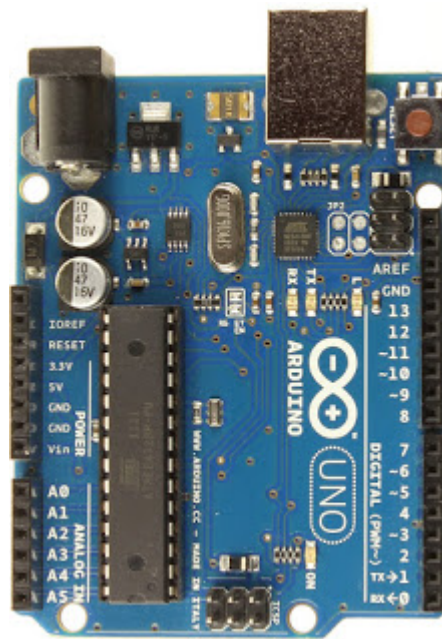


HNC Electrical and Electronic Engineering

PCEE 4004 Embedded Systems



Jim Stevens

172352

Introduction

Hardware Design

Software Design

Evidence of Implementation

Testing

Discussion, Evaluation and Conclusions

References

Appendix

Introduction

As part of the Embedded Systems module, candidates of the HNC in Electrical and Electronic Engineering are tasked with designing, testing and evaluating a simple embedded system. This assessment must demonstrate the candidate's underpinning knowledge of basic electrical components, basic programming using the C++ language, and the interaction between the hardware and software components of the designed system.

The task as set requires the candidate to “design and emulate a simple alarm system that will set off if it detects an object within a 2m vicinity of it's system”. The task as defined is open to a broad in interpretation. However, certain assumptions can be drawn from the brief as laid out, namely that the system must:

1. Detect movement using some form of sensor
2. Trigger a response when movement is within 2 meters of the sensor
3. That the response is that which might be characterised as an alarm
4. Utilise a microcontroller to control inputs and outputs of the system

Any system designed under this assessment must meet these broad criteria.

Commercially available systems will be used to inform the kinds of features and responses that may be desirable in an alarm system. The system, however, will be tailored to meet the requirements of the above brief.

Hardware Design

Commercially available units are constructed by a multitude of manufacturers. For example, Yale (<https://yalehome.co.uk/alarms/>, viewed 20th May, 2021) produce systems that can detect motion from multiple sensors, where the system then provides a number of alerts including a visual alarm, audible alarm, text message alerts and informs local authorities.

Many of these systems hold common features such as the visible and audible alarms, along with programable settings. These would be desirable in a simple alarm system and so will be incorporated with the system designed under this brief.

As instructed under the brief, a microcontroller must be used in the design of this system. The Arduino Uno one such controller, and has been selected for many reasons. These including it's relatively low cost, ease of connection to other components (including other computer systems for compiling purposes), interoperability of other commonly available components, possible operation from either 12 volt d.c. or mains power supply, support of the C++ programming language, and the wider support of a large community of system designers with open-source attitudes to sharing of concepts and designs.

The Arduino Uno will form the core of the system and will serve as the gateway between the various components and stimuli of the system. Other components utilised in this system will include:

- Breadboard

The breadboard will act as the intermediary between the microcontroller and all other components. The breadboard allows for rapid connection of the various components in a reliable and predictable manner, both factors being decisive in effective prototyping of the system.

- Wires

Insulated copper-cored wires are the accepted low-cost and effective method for transmitting electrical inputs and outputs throughout an electronic system, and so will also be implemented in this system.

- Resistors

Simple and cost-effective devices used to increase the impedance within a circuit. This not only allows for reduced power consumption, but also acts to reduce the load passing through any components downstream of the resistor. This can act as a form of protection against overloading the component. It can also, in the case the case of the buzzer, act to suppress the maximum output of the component during prototyping.

As such, it should be noted that the value ascribed to the resistors may need altering prior to mass production of the final product.

- 4x4 Keypad

The keypad is a series of inter-connected push buttons. When programmed appropriately, the keypad allows for the user of the system to manually input commands. This allows for the operation a multi-layered program. In other words, instead of the system being solely in the 'on' or 'off' operation, the system may be used in a broader range of operating parameters. In the case of this system, the ability to arm, disarm and test would be desirable and achievable using the keypad.

- RGB LED

Commercially available systems all provide multiple alerts to the operator of the system and to the possible intruder. Visual alerts are commonplace, and few as simple as the light-emitting diode (LED). A red-green-blue (RGB) LED is not the simplest form. However, the compact packaging and increased functionality are both boons to a system that should be simple to operate.

- Buzzer

Audible alerts within an alarm system are also commonplace amongst the commercially available systems. The buzzer is one of the most basic forms and provides an intense burst of sound which is simply controlled by effective software programming.

- Ultrasonic Sensor

The ultrasonic sensor is a key component within this system. It utilises the transmission and reception of short pulses of very high frequency sound to detect the proximity of objects. Any alarm system must have some form of sensor to detect the proximity. As the detection of motion was not required within the brief, a simpler and more cost-effective method of detection is proximity detection offered by the ultrasonic sensor.

All prototyping was conducted using Tinkercad (<https://www.tinkercad.com>, viewed 20th May 2021). This hardware simulation online platform allows for free rapid prototyping of both physical components and the attendant programming.

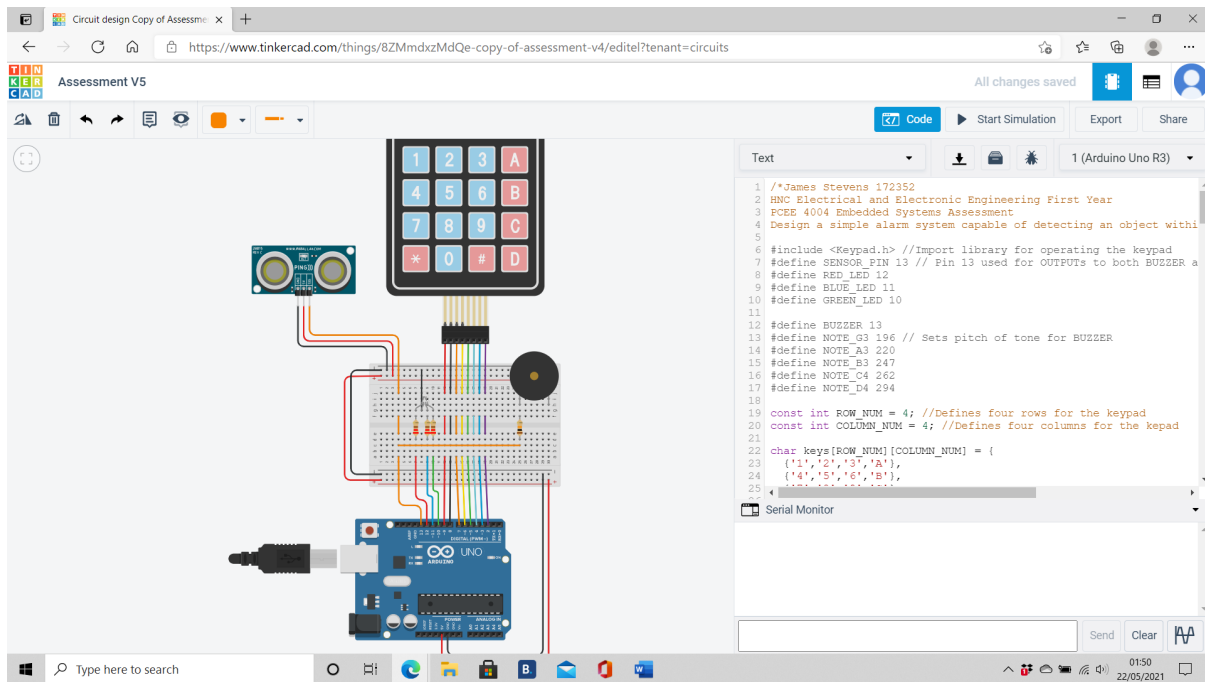


Figure 2: Image of physical layout of hardware components on Tinkercad

Software Design

The principle around which all of the commercially available systems operate is that of a closed-loop system. That is to say, once initiated, the software measures inputs against a set logical paradigm. If this paradigm is met, then further action may be taken by the system to either vary an output to alter the measured input (i.e. fire a central heating system if the ambient temperature drops below a set point), or may trigger further process within the program. This closed-loop system is advantageous as it makes operation of the system extremely simple for the end user as all of the logical processes required to enact a desired outcome are completed by the program upon the input of simple instructions. However, it is up to the compiler to understand both the components, programming language, the brief, and the interaction between all of these variables in order to achieve the desired outcomes.

In the terms of this brief, and in consideration of the available components, the flow chart using Lucidchart (<https://www.lucidchart.com>, viewed on 20th May 2021) listed in Figure 3 was constructed to plan the flow of operations required of the program.

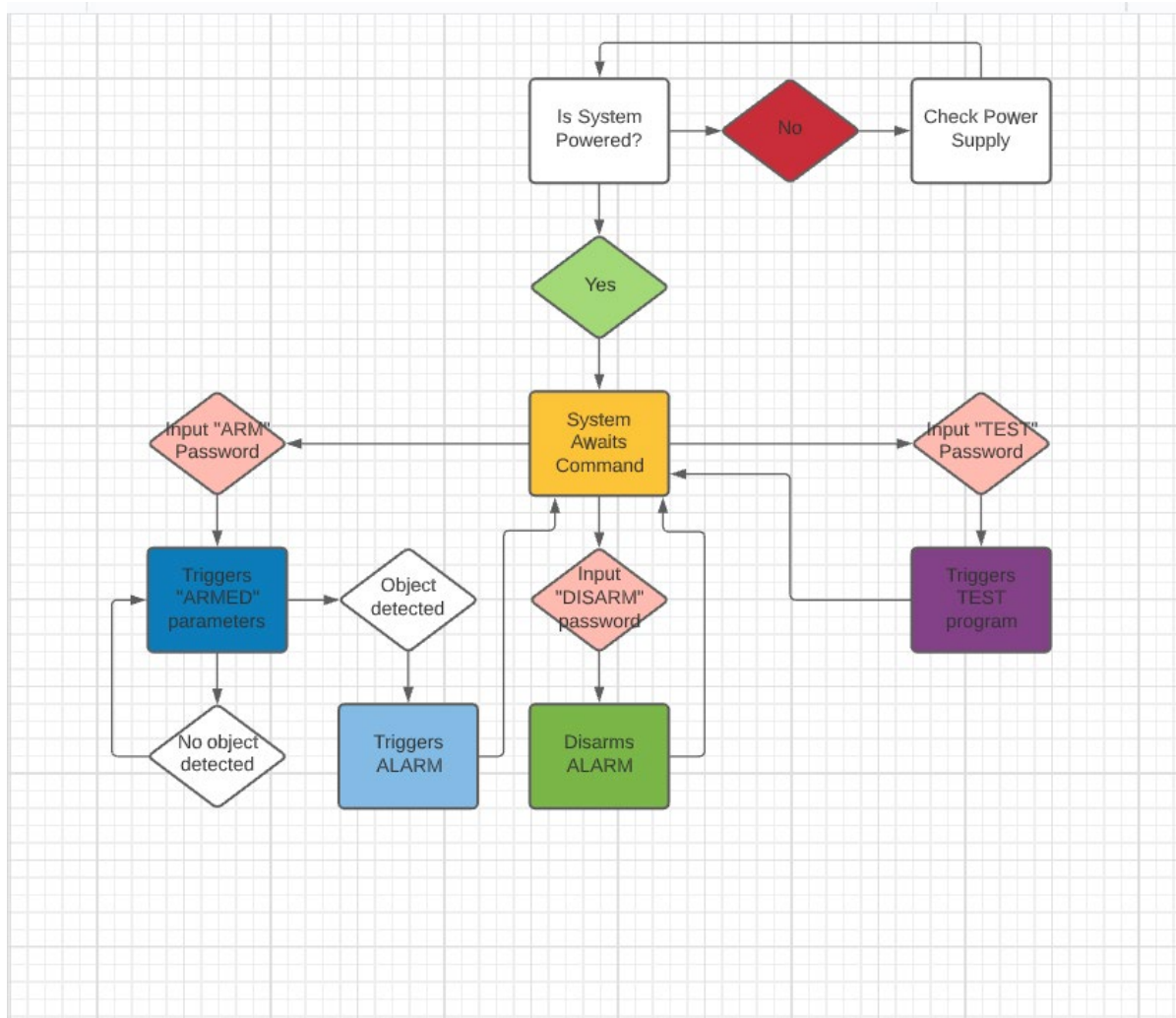


Figure 3: Flowchart generated using Lucidchart

The Arduino Uno functions by dividing its programming logic into three defined groups, those being global definitions, setup and void loops. Global definitions allow for definitions to be saved and initialised on powering the microcontroller, and these definitions then apply to all future actions of the Uno. The setup loop initialises programming relevant to the establishment of further actions, and is a useful method of undertaking actions that are only required once. The void loop is where the Uno will normally spend most of its operating time, and this where most of the logical steps of a program may be undertaken. Unless otherwise instructed, the Uno will repetitiously scan, or loop, through the program. Cues from the code or input from sensors are the normal methods for forcing progression of the program.

With the utility of the keypad in mind, it was decided that a central part of the program would be a central 'while' loop that would be used to capture inputs from the

keypad. After capturing the input (in the form of the appropriate password for the desired task), the program would then progress further onto various actions.

In the case of the “ARMED” path of the flow chart, the program would then scan for proximate objects. Where an object was detected within the defined parameter of 2 meters the program would then progress to activate the alarm. Otherwise, the program would continue to scan for objects. At the conclusion of this path, the program would return to the original ‘while’ loop.

The “DISARM” path, once selected, would override the operations of the “ARMED” path. The “TEST” path would test the visible and audible components of the system.

Implementation

The task required the use of the C++ programming language. Using the conventions set out within this language, this was the program utilised to meet the requirements of the brief.

```
/*James Stevens 172352
```

```
HNC Electrical and Electronic Engineering First Year
```

```
PCEE 4004 Embedded Systems Assessment
```

```
Design a simple alarm system capable of detecting an object within 2 meters*/
```

```
#include <Keypad.h> //Import library for operating the keypad
```

```
#define SENSOR_PIN 13 // Pin 13 used for OUTPUTs to both BUZZER and  
SENSOR
```

```
#define RED_LED 12
```

```
#define BLUE_LED 11
```

```
#define GREEN_LED 10
```

```
#define BUZZER 13
```

```
#define NOTE_G3 196 // Sets pitch of tone for BUZZER
```

```
#define NOTE_A3 220
```

```
#define NOTE_B3 247
```

```
#define NOTE_C4 262
```

```
#define NOTE_D4 294
```

```
const int ROW_NUM = 4; //Defines four rows for the keypad
```

```
const int COLUMN_NUM = 4; //Defines four columns for the keypad
```

```
char keys[ROW_NUM][COLUMN_NUM] = {
```

```
    {'1','2','3','A'},
```

```
    {'4','5','6','B'},
```

```
    {'7','8','9','C'},
```

```
    {'*','0','#','D'}
```

```
};
```

```
byte pin_rows[ROW_NUM] = {9, 8, 7, 6}; //Connects to the row pinouts of the keypad
```

```
byte pin_column[COLUMN_NUM] = {5, 4, 3, 2}; //Connects to the column pinouts of  
the keypad
```

```
Keypad keypad = Keypad( makeKeymap(keys), pin_rows, pin_column, ROW_NUM,  
COLUMN_NUM );
```

```
const String password1 = "123A"; //Enter ARM mode
```

```
const String password2 = "123B"; //Enter Disarm mode
```

```
const String password3 = "123C"; //Enter Test mode
```

```
String input_password; //Initialising characters related to the Keypad.h library
```

```
char state;
```

```
void setup(){
```

```
    Serial.begin(9600); //Set Baud rate
```

```
    input_password.reserve(32); // Maximum input characters is 33, change if needed
```

```
    state=0; //Boolean designed to force a return to the main loop
```

```
    pinMode (RED_LED, OUTPUT); //Setting Arduino pin to output to RGB LED
```

```
    pinMode (BLUE_LED, OUTPUT);
```

```

pinMode (GREEN_LED, OUTPUT);
pinMode (BUZZER, OUTPUT);
}

void loop(){

char key = keypad.getKey();
while(state==0){ //Sets while loop to recur if state=0 is operated

digitalWrite(GREEN_LED, HIGH);
delay(1000);

while (key==NO_KEY){ //Sets a nested while loop to await input from the keypad
key = keypad.getKey();
digitalWrite(GREEN_LED, LOW); //Used to pulse the GREEN_LED as visual cue
whenever a key is pressed
}
Serial.println(key); //Print to serial monitor any inputs from the keypad

if(key == '*') {
input_password = ""; // Device to clear input password
}
else if(key == '#') { // Device used to conclude capture of password
if (password1 == input_password) {
state = 1; //Process initiated by inputing password1
Serial.println("Password correct");
delay(2000);
Serial.println("Please leave area, system arming");
digitalWrite(GREEN_LED, LOW);
digitalWrite(BLUE_LED, HIGH);
delay(1000);
digitalWrite(BLUE_LED, LOW);
}
}
}
}

```

```

    delay(1000);
    digitalWrite(BLUE_LED, HIGH);
    delay(1000);
    digitalWrite(BLUE_LED, LOW);
    delay(1000);
    Serial.println("System armed");
    digitalWrite(BLUE_LED, HIGH);
    delay(1000);
}
else if (password2 == input_password) {
    state = 2; //Process initiated by inputing password2
}
else if (password3 == input_password) {
    state = 3; //Process initiated by inputing password3
}
else if (input_password.length() == 4){
    Serial.println("Incorrect password");
    input_password = ""; //Process initiated by inputing any other 4-key combination
}
}
else { input_password += key; //Method for capturing input passwords
    key = keypad.getKey();
}
key = keypad.getKey();
}

switch (state){
    case 1: //Actions required when alarm is armed
        int distance, microseconds; //Activate ultrasonic sensor for transmit and receive
        pinMode(SENSOR_PIN, OUTPUT);
        digitalWrite(SENSOR_PIN, LOW);
        delayMicroseconds(2);

```

```

    digitalWrite(SENSOR_PIN,HIGH);
    delayMicroseconds(10);
    digitalWrite(SENSOR_PIN,LOW);
    pinMode(SENSOR_PIN,INPUT);
    microseconds=pulseIn(SENSOR_PIN,HIGH);
    distance=microseconds/29/2; //To correct input from sensor from
microseconds to centimeters
    Serial.println(distance);
    if (distance<=200) //Trigger for alarm if target within 200cm distance of sensor
    {
        Serial.println("INTRUDER DETECTED!");
        digitalWrite(GREEN_LED, LOW); //Pulsing RGB LED as visual warning
        digitalWrite(BLUE_LED, LOW);
        delay(1000);
        digitalWrite(RED_LED, HIGH);
        delay(1000);
        digitalWrite(RED_LED, LOW);
        delay(1000);
        digitalWrite(RED_LED, HIGH);
        delay(1000);
        digitalWrite(RED_LED, LOW);
        delay(1000);
        Serial.println("ALARM ACTIVATED");
        digitalWrite(RED_LED, HIGH); //Sounding of alarm
        tone(BUZZER, NOTE_D4);
        delay(2000);
        state=0; //Return to initial state, allowing for further input to the keypad
    }
    break;

case 2: //Actions required when alarm is disarmed
    Serial.println("Password correct");

```

```
delay(2000);
Serial.println("System disarmed");
tone(BUZZER, LOW); //Cancel audible alarm
digitalWrite(RED_LED, LOW); //Cancel visual alarm
digitalWrite(GREEN_LED, HIGH); //Visual indication of return to initial state
delay(1000);
state=0; //Return to initial state, allowing for further input to the keypad
break;
```

case 3: //Actions required when testing the system

```
Serial.println("Password correct");
delay(2000);
Serial.println("System test mode");
digitalWrite(GREEN_LED, HIGH); //Testing of RGB LED as visual test
  delay(500);
digitalWrite(RED_LED, HIGH);
  delay(500);
digitalWrite(BLUE_LED, HIGH);
  delay(500);
digitalWrite(GREEN_LED, LOW);
  delay(500);
digitalWrite(RED_LED, LOW);
  delay(500);
digitalWrite(BLUE_LED, LOW);
  delay(500);
```

```
tone(BUZZER, NOTE_B3, 200); //Play Ode to Joy phrase as audible test
delay(300);
tone(BUZZER, NOTE_B3, 200);
delay(300);
tone(BUZZER, NOTE_C4, 200);
delay(300);
```

```
tone(BUZZER, NOTE_D4, 200);  
delay(300);  
tone(BUZZER, NOTE_D4, 200);  
delay(300);  
tone(BUZZER, NOTE_C4, 200);  
delay(300);  
tone(BUZZER, NOTE_B3, 200);  
delay(300);  
tone(BUZZER, NOTE_A3, 200);  
delay(300);  
tone(BUZZER, NOTE_G3, 200);  
delay(300);  
tone(BUZZER, NOTE_G3, 200);  
delay(300);  
tone(BUZZER, NOTE_A3, 200);  
delay(300);  
tone(BUZZER, NOTE_B3, 200);  
delay(300);  
tone(BUZZER, NOTE_B3, 500);  
delay(500);  
tone(BUZZER, NOTE_A3, 100);  
delay(150);  
tone(BUZZER, NOTE_A3, 600);  
delay(1000);
```

```
Serial.println("Test Complete");  
state=0; //Return to initial state, allowing for further input to the keypad  
break;
```

```
case 4:  
    Serial.println("Password incorrect");  
    delay(2000);
```

```

    break;
}

input_password = ""; // clear input password
}

```

Testing

A process of incremental testing of each component was undertaken during the prototyping phase. It was established that beginning the project with a simple design was most effective, and then adding further functionality as the platform was found to be stable. The keypad was followed by the RGB LED, with the buzzer being the final addition.

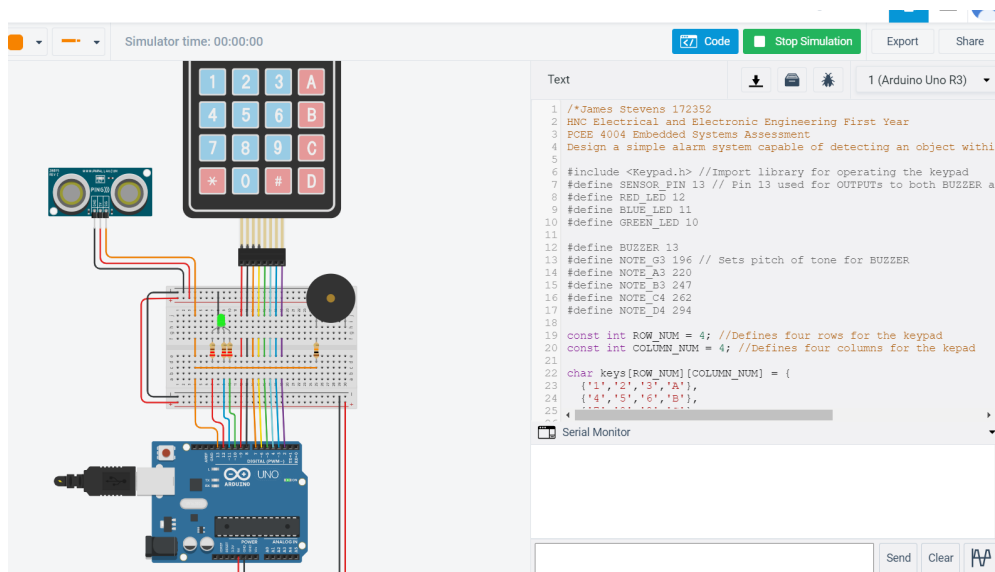


Figure 4 Showing Power-up phase of system

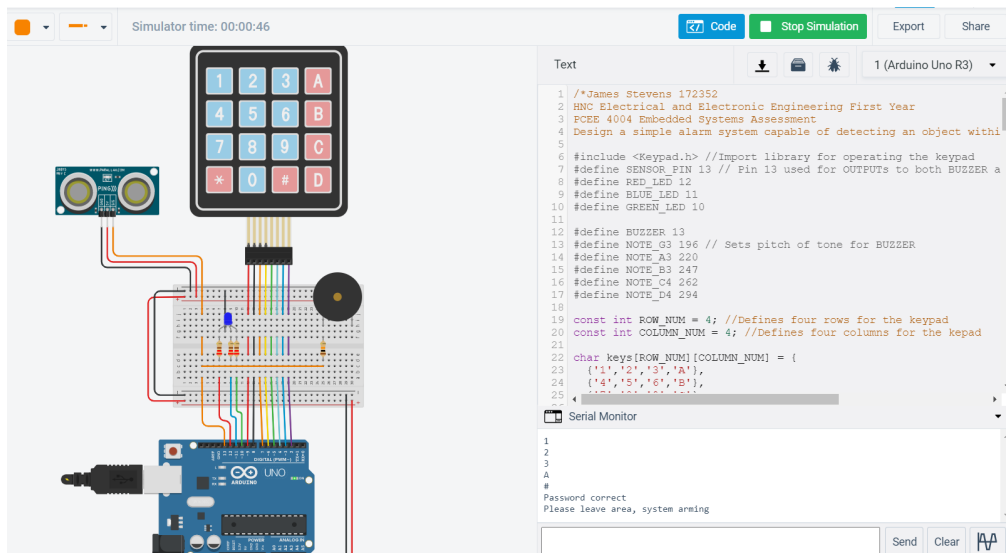


Figure 5 Arming System

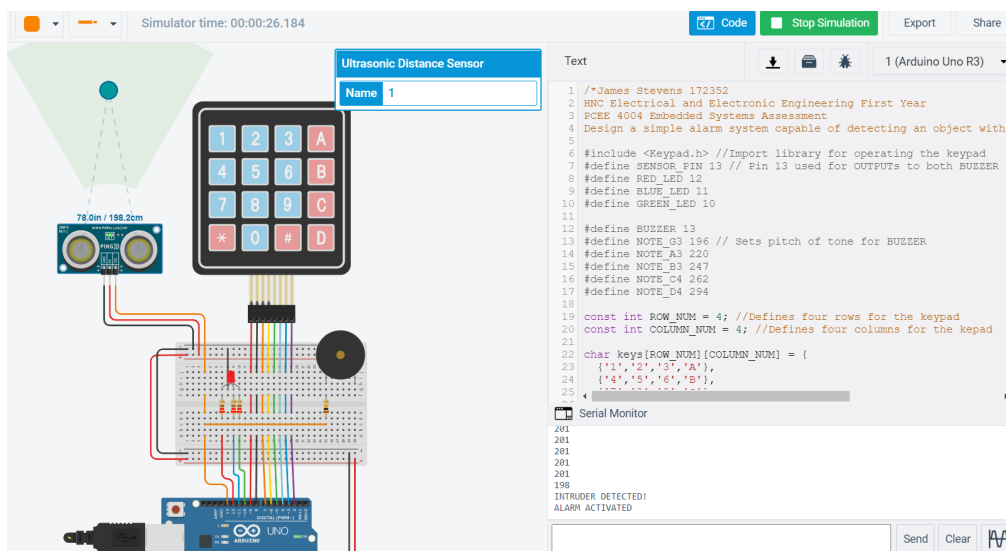


Figure 6 Alarm triggered by proximity sensor

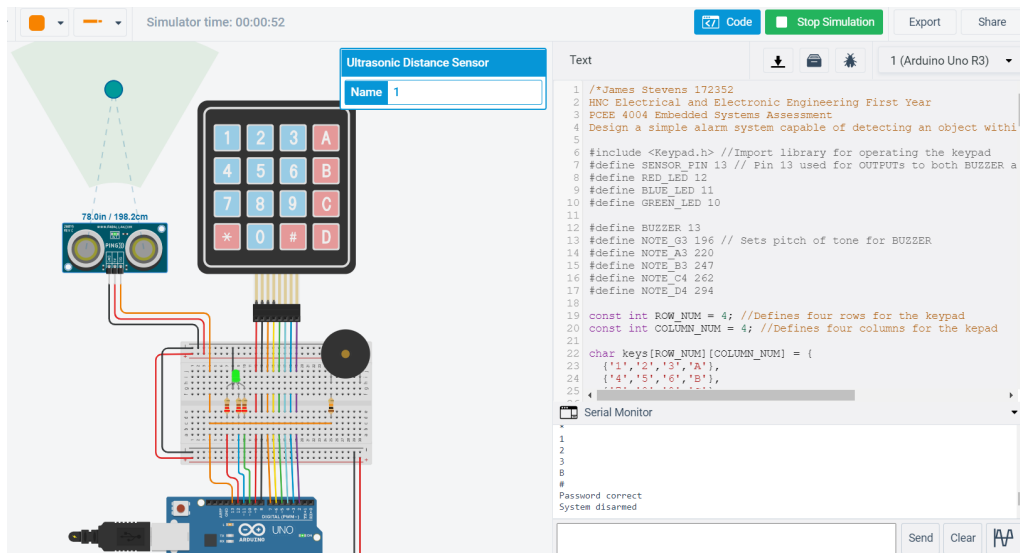


Figure 7 Alarm Disarmed

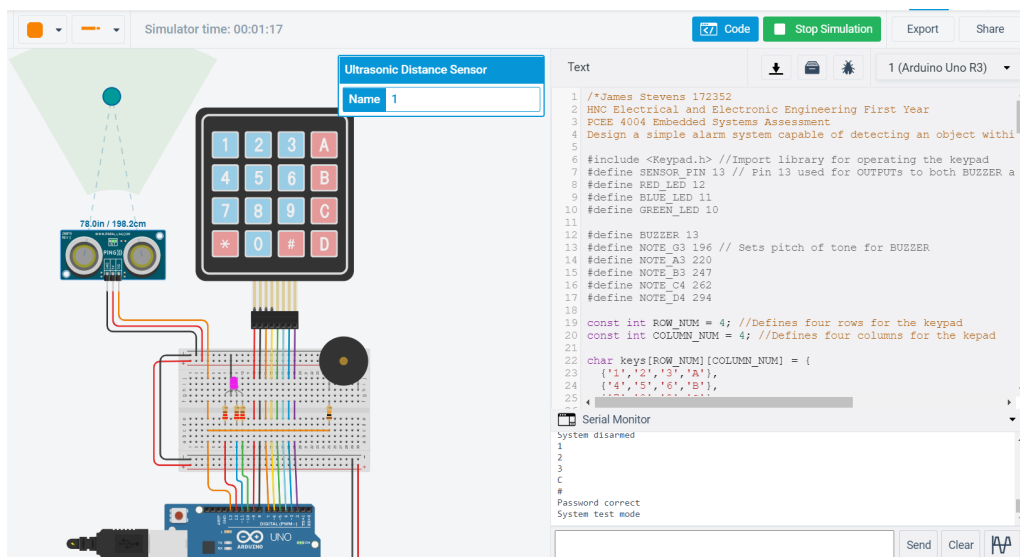


Figure 8 System under “Test” conditions

Discussion, Evaluation and Conclusions

With reference to the initial brief, four requirements were established, namely:

1. Detect movement using some form of sensor
2. Trigger a response when movement is within 2 meters of the sensor
3. That the response is that which might be characterised as an alarm

4. Utilise a microcontroller to control inputs and outputs of the system

Having reviewed the system as implemented, the basic requirements of the brief have been met.

Further utility to the user might have been rendered by utilising the RGB LED more widely, or with greater variation. Alternatively, the use of an LCD screen to output text to confirm inputs to the keypad would have been desirable. However, the limitations of the number of connections of the Arduino Uno made this a challenge.

Similarly, the additive nature of the design process of the programming has led to a very lengthy, unwieldy code. Further thought and design could fruitfully be placed in trying to abbreviate the code.

In part, this is inherent in the C++ coding language. The object-oriented coding can make brief coding very rapid and flexible ([https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/#:~:text=Disadvantages%20of%20C%2B%2B%201%20Use%20of%20Pointers.%20Pointers,Absence%20of%20Garbage%20Collector.%20...%20More%20items...%20,viewed 20th May 2021](https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/#:~:text=Disadvantages%20of%20C%2B%2B%201%20Use%20of%20Pointers.%20Pointers,Absence%20of%20Garbage%20Collector.%20...%20More%20items...%20,viewed%2020%20May%202021).)). However, multiple operations can rapidly become unwieldy.

References

<https://yalehome.co.uk/alarms/>, viewed 20th May, 2021

<https://www.tinkercad.com>, viewed 20th May 2021

<https://blarbl.blogspot.com/2016/03/getting-to-know-arduino-uno-board.html>, viewed 20th May 2021

<https://www.lucidchart.com>, viewed on 20th May 2021

<https://data-flair.training/blogs/advantages-and-disadvantages-of-cpp/#:~:text=Disadvantages%20of%20C%2B%2B%201%20Use%20of%20Pointers.%20Pointers,Absence%20of%20Garbage%20Collector.%20...%20More%20items...%20>

STUDENT FEEDBACK FORM

This section details the *extent* to which the assessment criteria are demonstrated by you, which in turn determines your mark. The marks available for each category of skill are shown. Lecturers will use the space provided to comment on the achievement of the task(s), including those areas in which you have performed well and areas that would benefit from development/improvement.

Generic Assessment Criteria	Marks available	Marks awarded
1. Engagement with Literature Skills		
	0	
2. Knowledge and Understanding Skills		
Hardware design 11/ 15 Excellent knowledge demonstrated of the hardware used with clear coherent reasons given for choices made Software design 7/15 Flowchart is missing any indication of a starting position, and does not use the appropriate shapes for decision/process etc. However, an overview of the software's function can be ascertained from the flowchart. Overall you have not really explored how the software operates the hardware, but have given an overview of the logical steps of the program	30	18
3. Cognitive and Intellectual Skills		
Discussion, evaluation and conclusion Reasonable attempt at evaluation of the underlying concepts and principles, not as full a discussion as other parts of the assignment. Possibly due to word count limitations at this point as generally this assignment shows that it has been very well considered in the design phase. Some valid suggestions are made for improvement along with consideration of the usefulness of C programming language.	15	8
4. Practical Application Skills		
Implementation 27/40 Code is well written with self-annotating variables, use of constants where appropriate and comments.	50	31.5

The additional features of testing and disarming the alarm are useful and extend the functionality of the system. Further marks in this section would have been gained by the use of functions, for example a function for the “tune” played by the buzzer when set off. testing 4.5/10 The inclusion of a test outside the 2m range on the boundary and just on the 2m would have been a useful addition here. Also, what would happen if the alarm is triggered at the 2m and then the intruder moves out of range? This should also be demonstrated. A testing table is a useful way to show what areas are being tested in a more compact fashion.		
5. Transferable Skills for Life and Professional Practice		
Structure of report Work is coherent, very fluent and is presented professionally. However datasheets for the components selected are missing from the appendices . Code layout is indented on the most part correctly and makes good use of self-annotation techniques.	5	3.5

Assessment Mark (Assessment marks are subject to ratification at the Exam Board. These comments and marks are to give feedback on module work and are for guidance only until they are confirmed.)	Late Submission Penalties (tick if appropriate)		61%
	Up to 1 week late (40% Max)		
	Over 1 week late (0%)		