

LEVEL 4 ASSESSMENT SPECIFICATION

Student name:	Morgan Layton	Student ID number:	172325
Programme:	HNC Electrical/Electronic Engineering, Power Systems Engineering & Instrumentation Engineering		
Module:	Embedded Systems		
Module code:	PCEE4004	Contribution to Overall Module Assessment (%):	100%
Lecturer:	Sarah Williams	Internal Verifier:	Jason Pointer
Assignment Title:	Alarm System	Word count (or equivalent):	2000 words
Submission deadline:	19 th May 2021	Return date of provisional marks & written feedback:	2 nd June 2021
Submission method:	All written assessments, where practical and possible, must be submitted via Turnitin unless otherwise instructed by the Lecturer. (Please DO NOT put this assessment specification into Turnitin or it will match many similarities with other students' submissions.) Alternative submission method (if applicable): Late submission of the assessment will result in a late penalty mark. Penalties for late submission: Up to one week late, maximum mark of 40%. Over one week late, 0%. Only the Extenuating Circumstances Panel may approve a change to submission dates.		
Academic honesty / referencing:	Academic honesty is required. In the main body of your submission you must give credit to authors on whose research and ideas your work is based. Append to your submission a reference list that indicates the books, articles, etc. that you have used, cited or quoted in order to complete this assessment.		

Module Learning Outcomes (from module syllabus)

1. Demonstrate a good understanding of the fundamental principles of programming an embedded system.
2. Describe the basic elements of typical embedded hardware and apply that knowledge in the analysis of a number of simple system applications.

Introduction

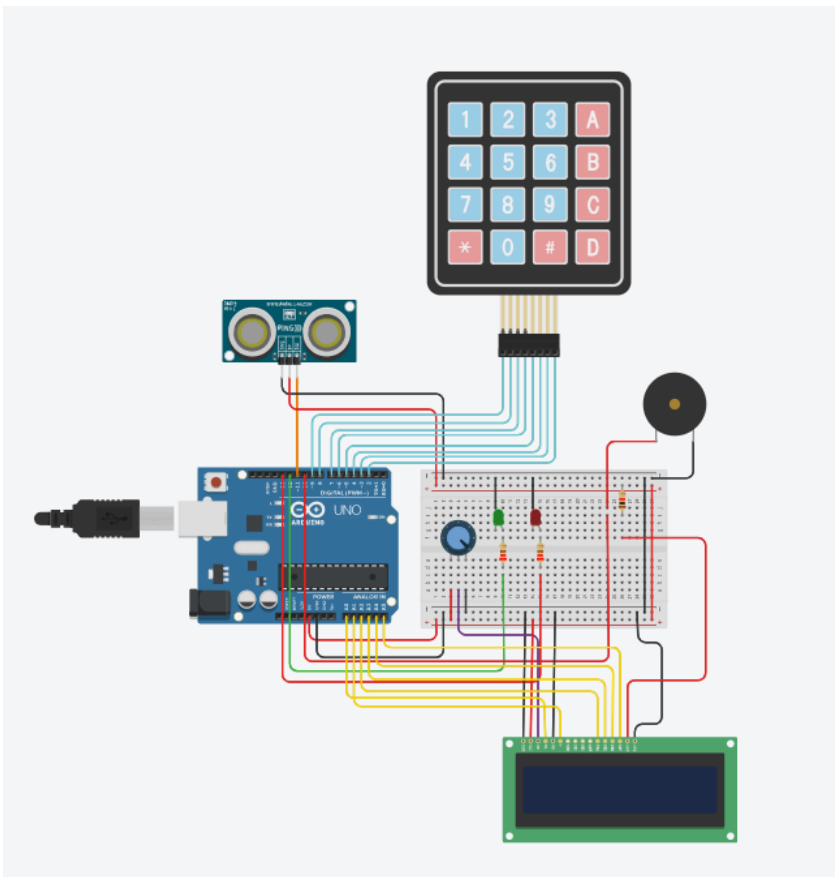
The design brief for this embedded system is to create a simple alarm system that notifies the user when an object is within a 2M vicinity of the device. I am therefore concluding that the device should go off once an object is within this range so anything within 2M but not including 2M. The alarm must also cease when an object leaves the designated 2M vicinity so a way of letting the user know that the alarm is set but not active would be appropriate. Although the design brief only mentions LED's as a way of notifying the user it does leave scope for other components to be used therefore, I feel that a component capable of producing a loud and clear warning sound is suitable addition, for example a piezo buzzer. Another important addition I plan on implementing would be a way of disabling the alarm for a period of time if the user needs to disable the alarm for maintenance or temporary access for practical use.

Task 1- Hardware design

Using Tinkercad simulator I came up with a hardware design for a microcontroller-based alarm system outlined in the introduction. Firstly, the microcontroller available for this project is the Arduino Uno single board computer which uses the C++ programming language which will be connected to a breadboard to hold the circuitry. To detect when an object is within a 2M vicinity an ultrasonic sensor is to be deployed and will relay its information to the microcontroller. When an object is detected by the sensor that is within the defined vicinity a red LED is to be turned on and a piezo buzzer will an alarm sound.

A keypad will be introduced to the circuit to enable the user to enter a set password, which they will cause the alarm to be disabled for a duration for practical use. Another element intended to improve practical use is the LCD screen which will aid the user when inputting their password and giving them the alarm's status.

It is also a requirement that if an object exits the designated vicinity that the piezo buzzer and red LED cease. In this case a useful addition will be a green LED that notifies the user that the alarm is armed but has not been set off or has been disabled via keypad input.

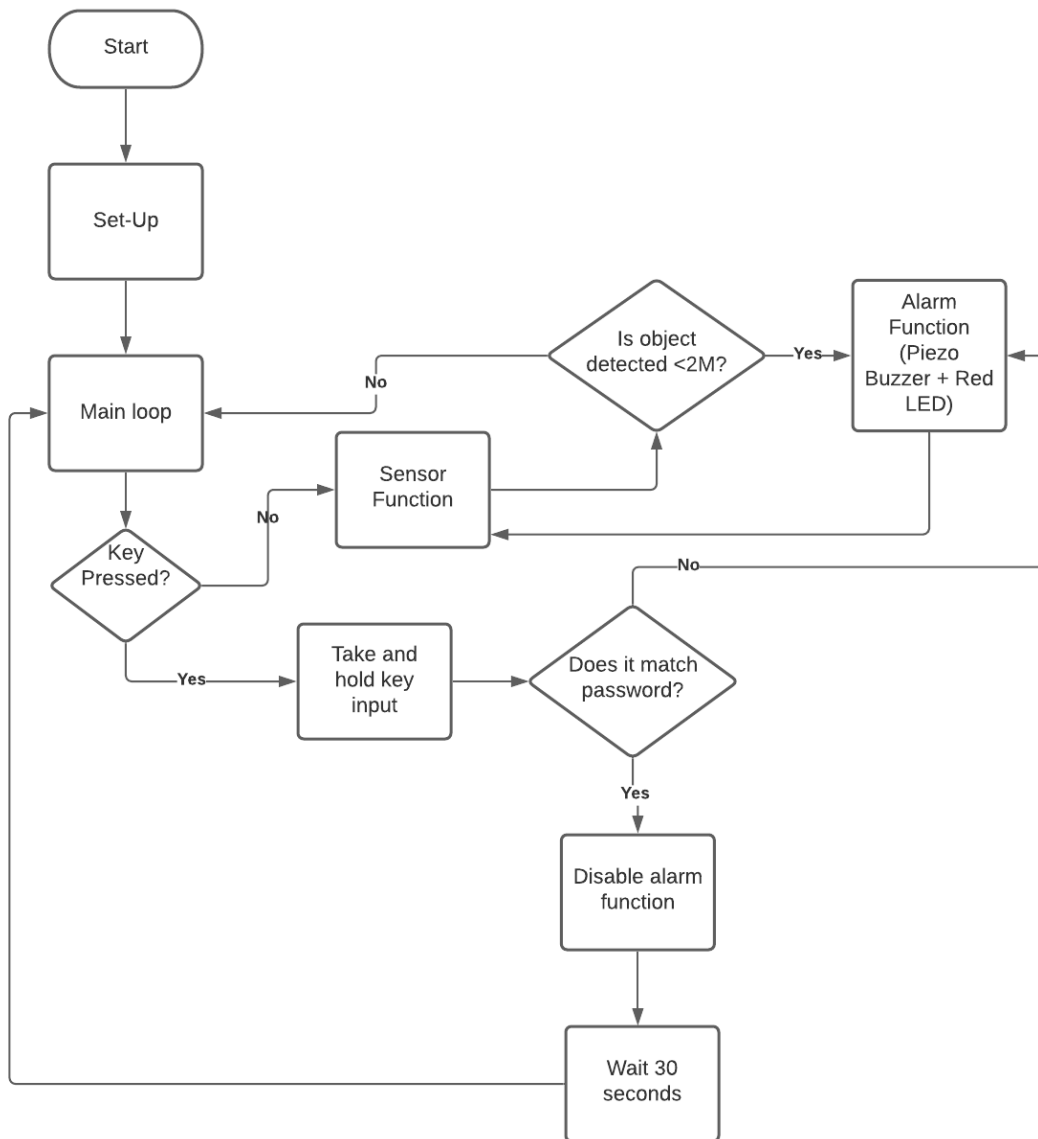


Task 2- Software Design

Below is the software design flowchart created using Lucid chart to help design the embedded system program. The program starts and enters set up where pins will be allocated, and global variables introduced. The main body of code then executes which then looks to see if a key has been passed via the keypad. A decision is then made if no key pressed the sensor function which sits outside this body of code is called activating the ultrasonic sensor. A second decision is made where we look to see if an object is present if not, it returns to the main loop. If an object is present however the alarm function is called which will set off the piezo buzzer and illuminate the red led. This function will then return to the sensor function and continue to loop until the object leaves the area.

If a key is pressed it will be stored in a variable and compared with the original password variable defined in the set-up. If these match the alarm will be disabled by calling the disable alarm function the code delayed for 30 seconds. After the 30 seconds the variable that the keys are passed into will be reset and the main loop begins again. However, if they do not

match the alarm function will be called setting off the alarm.



Pseudocode for Alarm Program

Variable String password = 8921;

Variable String temporaryPassword = Nothing;

Variable byte ROWS = 4;

Variable byte COLUMNS = 4;

Variable char keypressed;

Array keymap[ROWS][COLUMNS]= Keypad Values;

Pin Values [ROWS]={9,8,7,6};

Pin Values [COLUMNS]={5,4,3,2};

LCD Pins (14, 15, 16, 17, 18, 19);

Function Sensor()

```
{
  Pin State (Sensor Output Pin, On);
  Delay;
  Pin State (Sensor Output Pin, Off);
  Delay;
  Variable time(microseconds) = Sensor Input Pin, On);
  Variable distance = time/29/2;
  if (distance <=199){
    Function Alarm();
    Pin State (Green Led, Off);
    Pin State (Red Led, On);

  } else if (distance >=200){
    Pin State (Green Led, On);
    Pin State (Red Led, Off);
    Pin State (Piezo Buzzer, Off);
  }
}
```

Function Alarm()

```
{

Pin State (Piezo Buzzer, On);
Delay;
Pin Sate (Piezo Buzzer, Off);
Delay;

}
```

Function disable Alarm()

```
{
  While loop (temporaryPassword== password
  {
    Pin State (Green Led, On);
    Pin State (Red Led, Off);
    Pin State (Piezo Buzzer, Off);
    Delay(30 sec);
    Temporary Password== Nothing.
    Return Main loop
  }
}
```

Setup Function()

```
{
Set Pin's for Buzzer, Led's and Sensor to Output
}
```

```

Main loop Function()/
{
    if (keypressed == No key)
    {
        Function Sensor();
    }
    else if (keypressed != No key)
    {
        if (keypressed == any number);
        {
            temporaryPassword+= keypressed;
            LCD.print('*');
        }
        if (keypressed == *)
        { if (temporaryPassword != password)
        {
            Function Alarm();
            LCD.print('WRONG');
        }
        if (temporaryPassword==password)
        {
            Function disableAlarm();
            LCD.print('CORRECT');
        }
        }
    }
}

```

Task 3- Implementation

Code

```

//-----User Guide-----
//please note the password to disable the alarm is '8921'
//press * to enter the password
//press # if you have made a mistake to start again
//this will disable the alarm for 30 seconds
//-----User Guide-----

#include <Keypad.h>
#include <LiquidCrystal.h>

#define ALARM 10
#define SENSOR_PIN 11
#define LED_GREEN 12
#define LED_RED 13

```

```

//^--Giving logical names to arduino pins for readability and future maintenance.

LiquidCrystal lcd (14, 15, 16, 17, 18, 19);

const String password="8921";//Variable to hold password.

String tempPassword="";//Variable to temporary password attempt.

const byte ROWS=4;//Variable to hold rows for keypad matrix.

const byte COLUMNS=4;//Variable to hold columns for keypad matrix.

char keypressed;//Variable to hold char value input from keypad.

char keymap[ROWS][COLUMNS]= {{'1','2','3','A'},{'4','5','6','B'},{'7','8','9','C'},{'*','0','#','D'}};
//^--A character array of set values matching the keypad hardware.

byte rowPins[ROWS]={9,8,7,6};//Pin values for Rows.

byte columnPins[COLUMNS]={5,4,3,2};//Pin values for Columns.

Keypad myKeypad= Keypad(makeKeymap(keymap),rowPins,columnPins,ROWS,COLUMNS);
//^--Calling keymap from library and assigning it values.

//-----Keypad/Password-----Global variables--^

int Sensor()

{
  int distance,microseconds;//initialising two variables to hold distance + time.
  pinMode(SENSOR_PIN,OUTPUT);//Sensor pin set to output a volatge
  digitalWrite(SENSOR_PIN,LOW);//Setting pin to 0V.
  delayMicroseconds(10);//A delay to clear residual volatge.
  digitalWrite(SENSOR_PIN,HIGH);//Setting pin to output a volatge.
  delayMicroseconds(10);//Leaving volatge on for sensor to detect and send.
  digitalWrite(SENSOR_PIN,LOW);//Sensor pin back to 0V.
  pinMode(SENSOR_PIN,INPUT);//Setting pin to recieve input.
  microseconds=pulseIn(SENSOR_PIN,HIGH);//Assigning time taken from pulseIn.
  distance=microseconds/29/2;//A sound wave travels at 1cm every 29 microseconds /2 gives
  //distance to object.
  if (distance <=199)//If distance is equal to or less than 199cm
  {
    funcAlarm();//The alarm function is called to sound the buzzer.
  }
  else if (distance >=200)//If distance is greater than or equal to 200cm.
  {
    digitalWrite(LED_GREEN, HIGH);//Green led is lit
    digitalWrite(LED_RED, LOW);//Red led extinguished
  }
}

//-----Sensor-----

```

```
void funcAlarm()//Function to set off buzzer
```

```
{
  tone(ALARM, 700, 100);//Setting pin, tone and duration.
  digitalWrite(LED_GREEN, LOW);//Green led extinguished.
  digitalWrite(LED_RED, HIGH);//Red led lit.
}
```

```
//-----StartAlarm-----
```

```
void disableAlarm()//Function to stop the alarm.
```

```
{
  while (tempPassword==password)//while loop executes while var == '8921'.
  {
    digitalWrite(LED_GREEN, HIGH);//Green led lit.
    digitalWrite(LED_RED, LOW);//Red led extinguished.
    Serial.println("ALARM DISABLED FOR 30 SECONDS");
    delay(30000);//30 second delay.
    tempPassword=="";//Password reset
    return;//return breaks out of the while loop.
  }
}
```

```
//-----DisableAlarm-----
```

```
void ledBlink()//function to blink led if user gets password wrong.
```

```
{
  digitalWrite(LED_RED, HIGH);//Red Led lit.
  delay(50);//small delay to make visible.
  digitalWrite(LED_RED, LOW);//Red Led extinguished.
  delay(200);//small delay to make visible to human eye.
}
```

```
//-----BlinkLED-----
```

```
void setup()//Code setup only runs at the start of program.
```

```
{
  lcd.begin(16, 2);//Setting out lcd screen dimensions.
  pinMode(ALARM, OUTPUT);//Alarm pin set to output.
  pinMode(LED_GREEN, OUTPUT);//green led pin set to output.
  pinMode(LED_RED, OUTPUT);//red led pin set to output.
  Serial.begin(9600);//baud rate set to 9600 for serial interface.
}
```

```
void loop()//the main program loop that runs repeatedly.
```

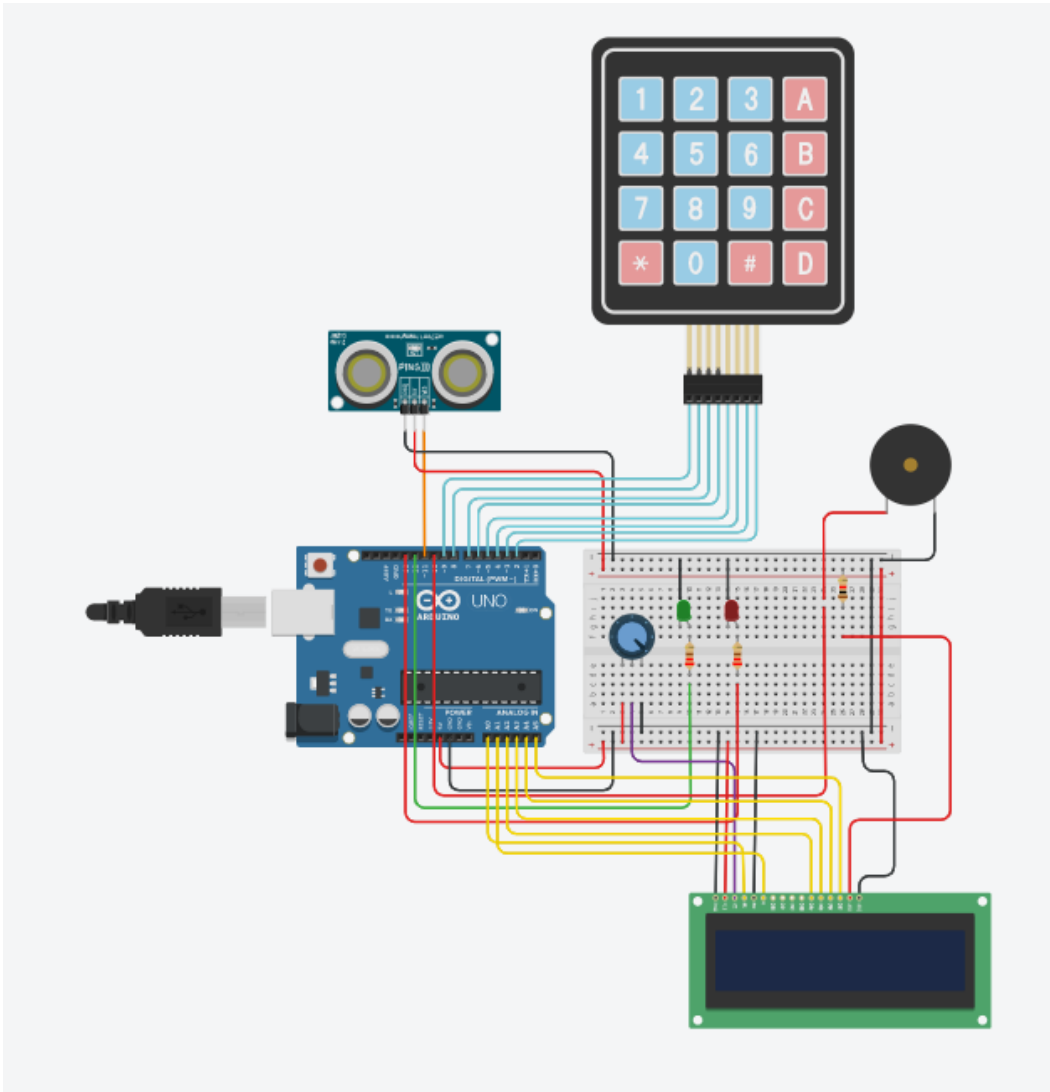
```
{
  keypressed = myKeypad.getKey();//var keypressed must have value from predefined library.
  if (keypressed ==NO_KEY)//No key is present
  {
    Sensor();//Call sensor function
  }
  else if (keypressed != NO_KEY)//if a key is selected
  {
```

```

if (keypressed == '0' || keypressed == '1' || keypressed == '2' ||
    keypressed == '3' || keypressed == '4' || keypressed == '5' ||
    keypressed == '6' || keypressed == '7' || keypressed == '8' ||
    keypressed == '9') // and is a number for password.
{
    tempPassword += keypressed; // each keypressed char is passed into temp string variable.
    lcd.print("*"); // For each key show '*' on lcd.
}
if (keypressed == '*') // if star asterix is selected the password is entered.
{
    if (tempPassword != password) // if entered password is wrong
    {
        lcd.clear(); // Clear lcd screen.
        lcd.print("**ACCESS DENIED**"); // Print 'ACCESS DENIED' on lcd screen.
        ledBlink(); // the blink led function is called
        lcd.clear(); // Clear lcd screen.
    }
    if (tempPassword == password) // if password matches correct
    {
        lcd.clear(); // Clear lcd screen.
        lcd.setCursor(0, 0); // first line of lcd screen.
        lcd.print("**ALARM DISABLED**"); // print 'ALARM DISABLED'
        lcd.setCursor(0, 1); // second line of lcd screen.
        lcd.print("**FOR 30 SECONDS**"); // print 'FOR 30 SECONDS'
        disableAlarm(); // the disable alarm function is called
        lcd.clear(); // Clear lcd screen.
    }
}
if (keypressed == '#') // if hashkey pressed
{
    lcd.clear(); // Clear lcd screen.
    Serial.println("PASSWORD INPUT RESET"); // message to user on serial monitor.
    tempPassword = ""; // temp password reset e.g. incorrect pin answered.
}
}
}

```

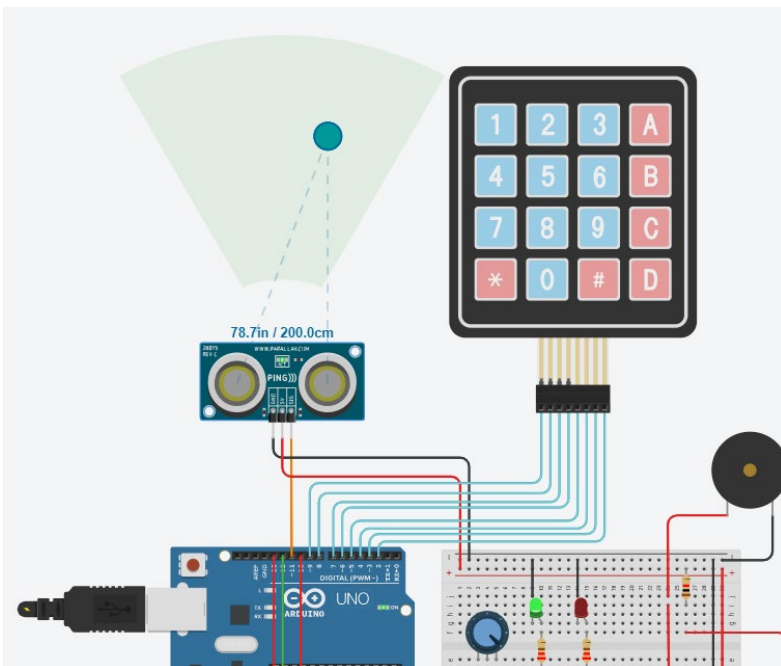
Circuit Layout and Components



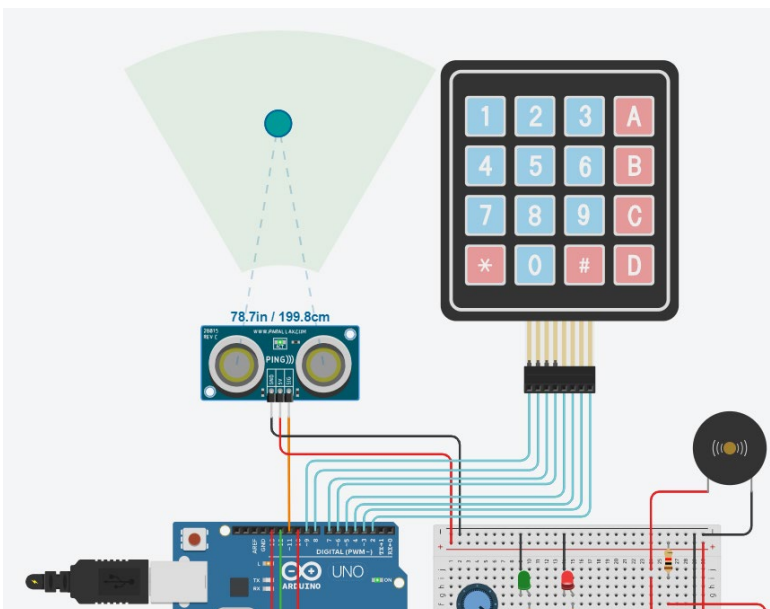
Task 4- Testing

Test Number	What is the test?	Expected Outcome	Actual Outcome	Evidence Number
Test 1	Placing an object at 200cm and 199cm.	Buzzer to go off and red led to light up at 199cm, nothing at 200cm.	At 200cm the alarm was inactive and green led lit. When the object was placed at 199.8cm the buzzer became active and the red led illuminated.	1 + 2
Test 2	Placing an object within 200cm and then removing it.	Buzzer to cease, red led to extinguish and green led to light up when object removed.	The object was placed within 200cm distance to set off the alarm and when removed back beyond the buzzer stopped, red led extinguished and the green led became lit.	3 + 4
Test 3	Entering password '8921' followed by '*'.	Alarm to successfully take a correct entry of '8921' and for the alarm to cease for 30 seconds.	The password '8921' was successfully entered and registered by the alarm resulting in the buzzer and red led to stop while the green led was illuminated for 30 seconds.	5 + 6 + 7
Test 4	Entering incorrect password '0000' followed by '*', then entering '#' followed by '8921' followed by '*'.	The alarm will continue until '8921' followed by '*' is entered.	The alarm did not cease with incorrect entry '0000' and a successful re-attempt with the correct password was made after pressing '#'. The alarm did not sound due to the ultrasonic sensor facing forward. This will be brought up in my evaluation.	8 + 9
Test 5	Placing an object within 200cm from the side.	The alarm will sound and red led will light up.		10
Test 6	Entering incorrect password '0000' followed by '*'.	The LCD screen will display message 'ACCESS DENIED' and red led will blink once.	The LCD screen clearly displayed 'ACCESS DENIED' and red led did momentarily turn on and off again however so did the buzzer. This seems to be an unintended side effect of the 'delay' command.	11 + 12
Test 7	Entering 4 digits followed by '#'.	The LCD screen will display each input with a '*' while the '#' will clear the screen.	The LCD screen successfully saw my inputs and displayed '*' according to each one. Pressing the '#' cleared the screen notifying me my attempt was reset.	13 + 14

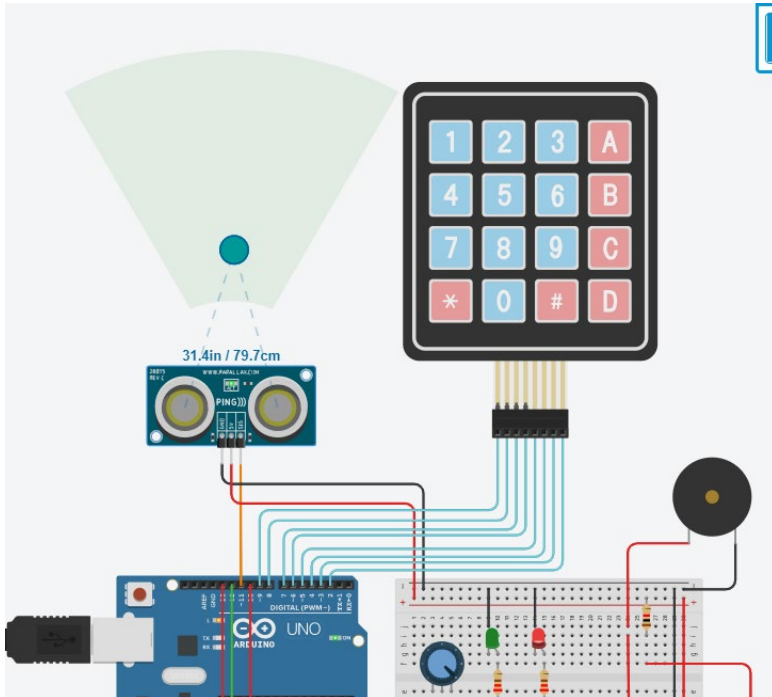
Evidence 1- Object at 200cm, alarm inactive.



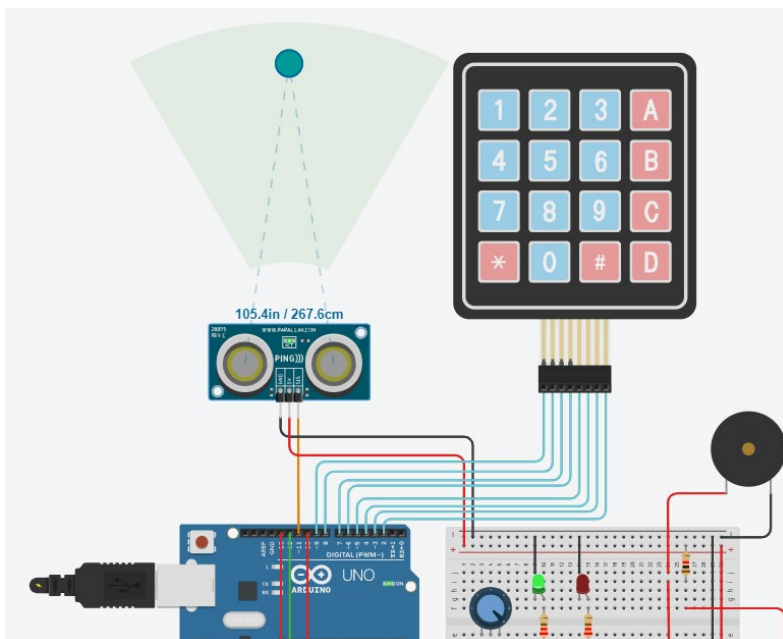
Evidence 2- Object at 199.8cm, alarm activated.



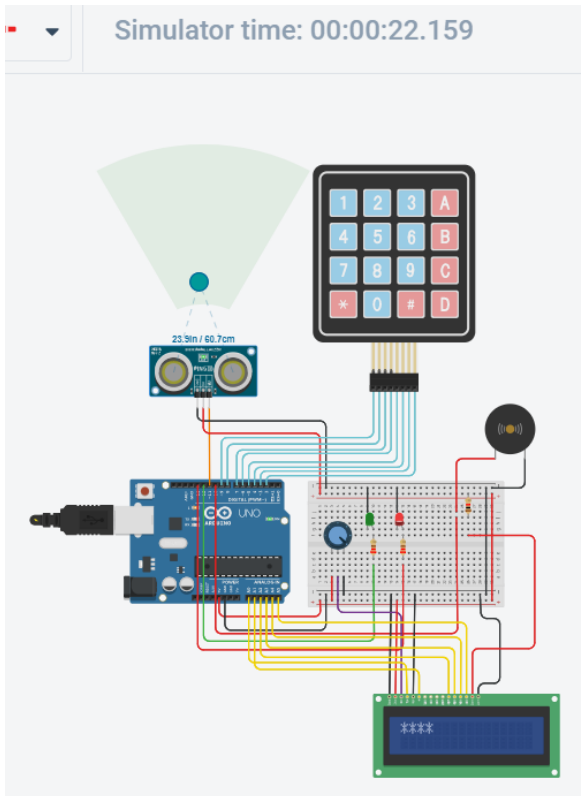
Evidence 3- Object <200cm and alarm active.



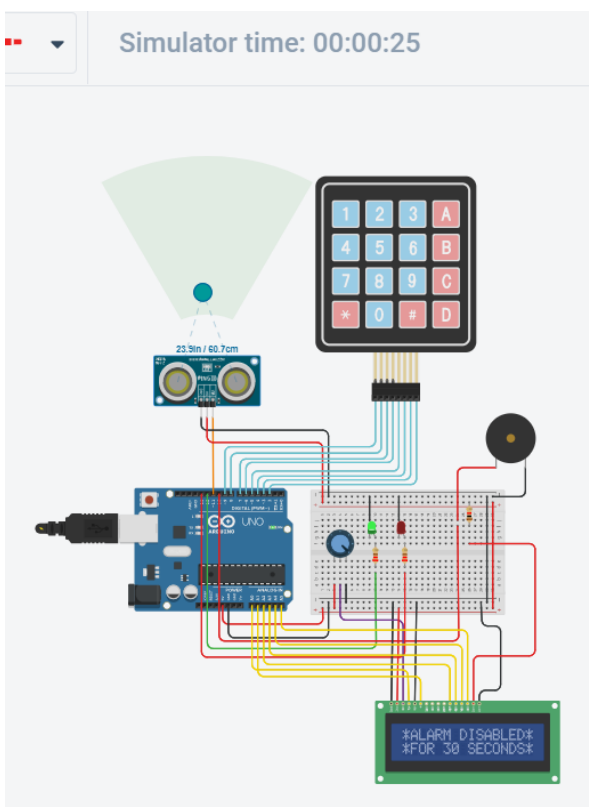
Evidence 4- Object removed >200cm, alarm ceased.



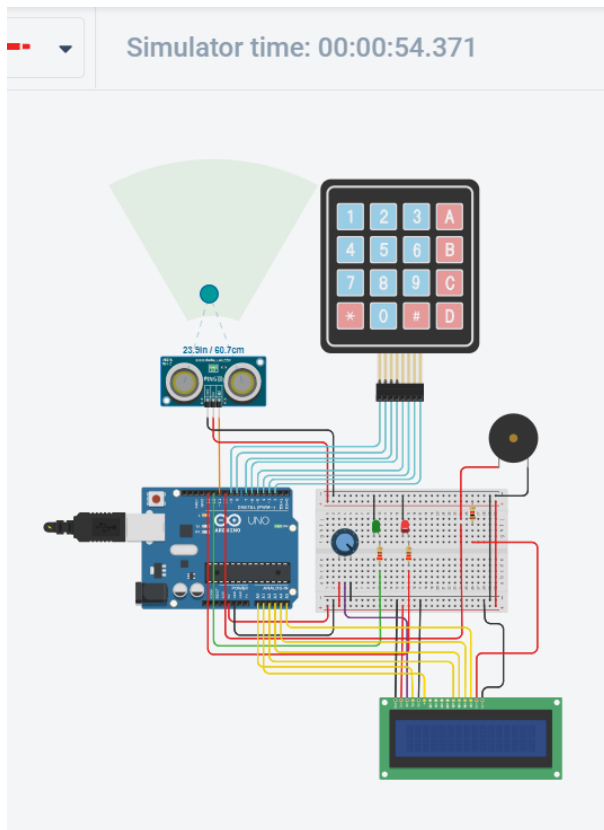
Evidence 5- Password '8921' entered via keypad.



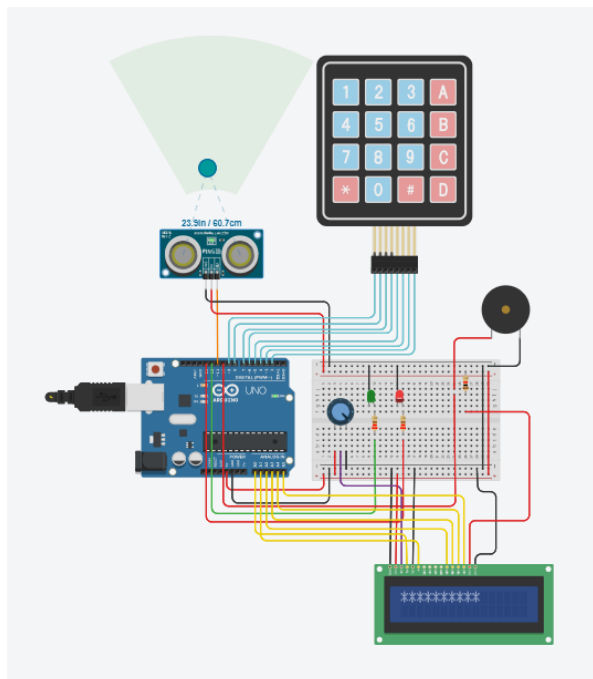
Evidence 6- '*' pressed password successfully submitted alarm disabled.



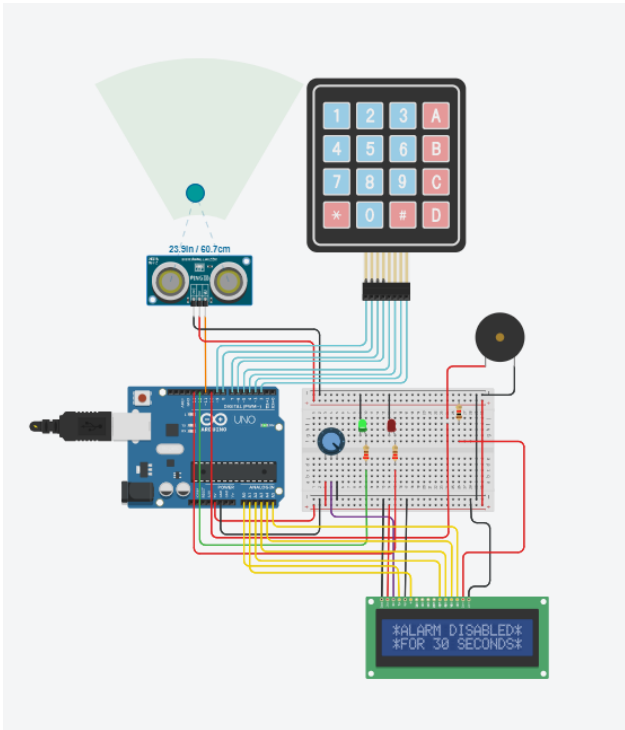
Evidence 7- Object <200cm, alarm active.



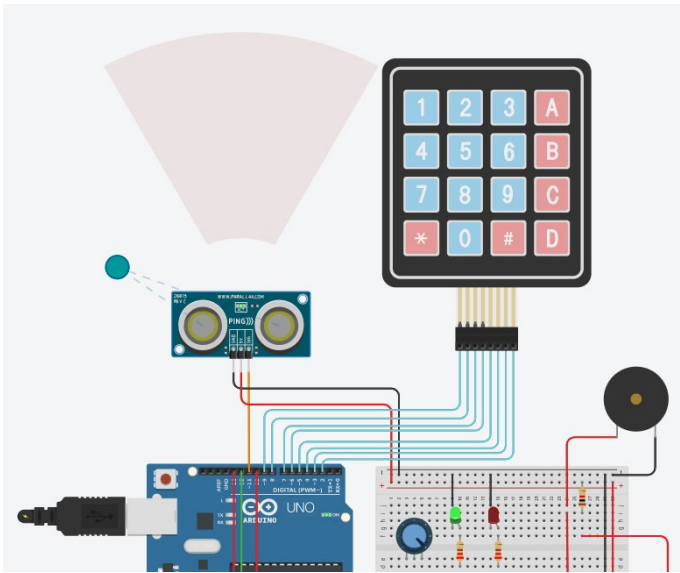
Evidence 8- Incorrect password submitted.



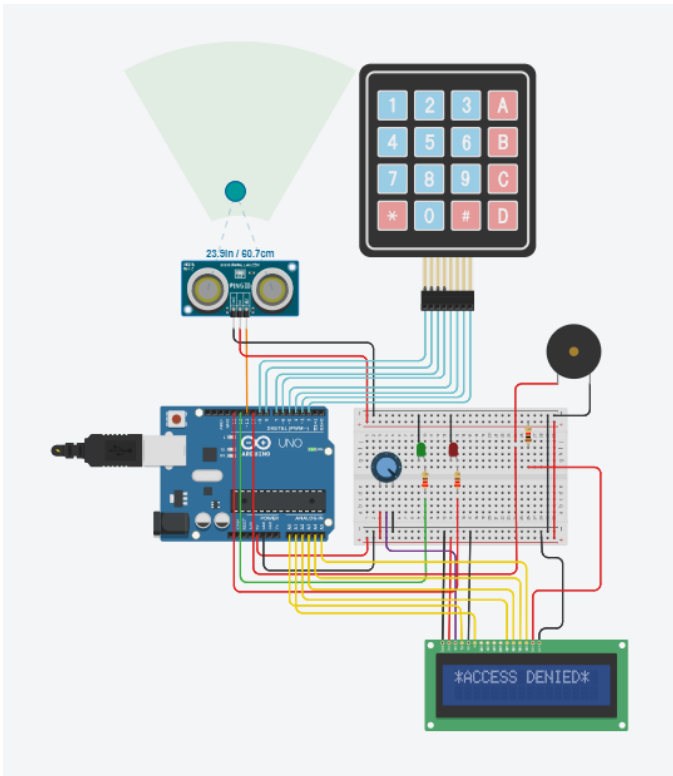
Evidence 9- Password reset and re-entered, alarm disabled.



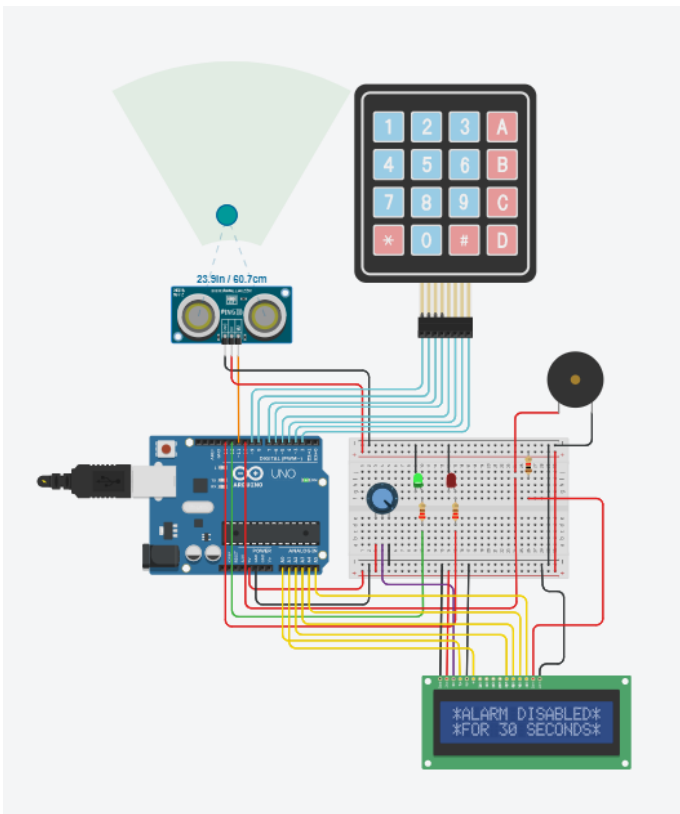
Evidence 10- Object <200cm but to the side, alarm inactive.



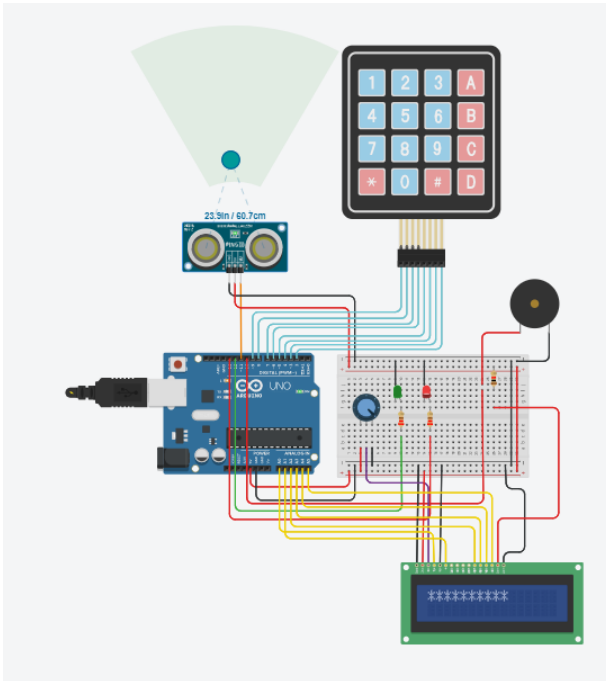
Evidence 11- Correct LCD message displayed after wrong entry.



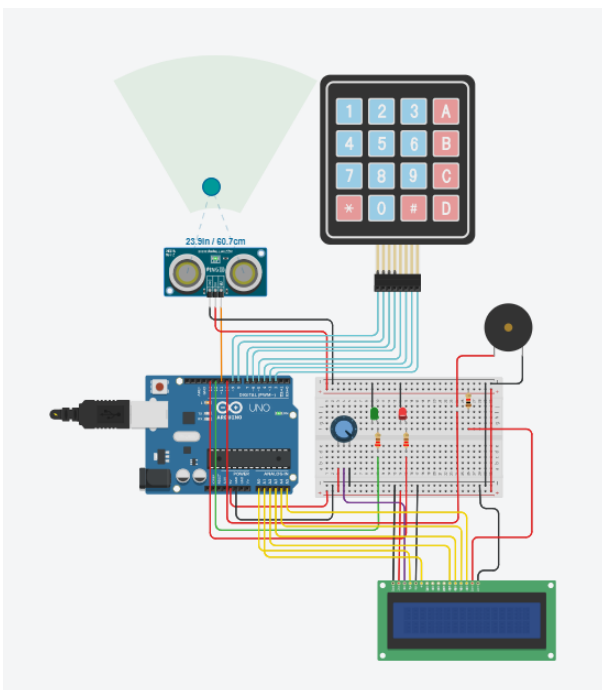
Evidence 12- Correct LCD message displayed for successful entry.



Evidence 13- LCD printing '*' for each key entered.



Evidence 14- LCD going blank when input reset using '#'.



Task 5- Evaluation

As far as the brief goes the alarm system designed and implemented in this project performs its required functions. It is able to detect an object within 200cm using the ultra-sonic sensor which then via the microcontroller then lights LED's and either sounds or doesn't sound the piezo buzzer depending whether that statement is true or false. It is then able to automatically cease if that object moves to 200cm or beyond as detailed in the brief. Further to this for the purpose of practicality a keypad enabling the alarm to be disabled for a short period of time has been introduced with a blink led function and lcd screen to display messages to the user so they can see if they have entered the password correctly or incorrectly. However, there are definitely things that could be added or improved on in future iterations of this design.

Firstly, as outlined in testing the alarm only works if the object approaches from the direction the ultrasonic sensor is facing. If that object approaches from the side or from behind, the alarm will not activate. Clearly this issue results in scenarios where the design doesn't meet the brief so is a cause for concern. To ensure the alarm is performing its intended purpose it is advised that this characteristic be considered when deploying the alarm into a given situation although the brief doesn't detail what that situation would be. An improvement to combat this issue could be done by adding multiple sensors facing in all directions to get a wider range of detection although this would require more pins than are currently available within this design.

A further issue with the hardware of the design is that the piezo buzzer lets off a monotone sound which is annoying rather than alerting. This piece of code makes a siren sound much more suitable to the design which I did try to implement during development:

```
for(int hz = 440; hz < 1000; hz+=25){//for loop to create whoop up via analog frequencies on alarm pin.
    tone(ALARM, hz, 10);//Setting pin, tone and duration.
    delay(5);//Small delay to allow clean transition to whoop down
}
for(int hz = 1000; hz > 440; hz-=25){//for loop for the whoop down.
    tone(ALARM, hz, 10);//Setting pin, tone and duration.
    delay(5);//Small delay to allow clean transition to whoop up
}
```

The reason this code was removed from the final implementation was because a delay function in milliseconds is too big and interferes with other parts of the code so when the keypad button is pressed the char is unlikely to be passed to the tempPassword variable. Without this relatively long delay however the for loop executes so quickly it's impossible to hear the frequencies being stepped up and down. I decided to go with the keypad over the siren noise due to the practicality the keypad brings enabling the user to disarm the alarm temporarily.

The delay function causes the entire program to stop meaning no other bits of code can be compiled. The delay in the ultrasonic sensor function is in microseconds so doesn't appear to affect the design but it is entirely possible once every so often this situation could occur. This is an issue when entering a keypad for example if you're unlucky enough to press it while the delay is active it won't be passed into the tempPassword variable. After some research there is a way around this with C++ on Arduino using the millis() value which can be used in a for loop to count the time it has taken for the program to compile plus or minus another value to cause a delay for however long you need. (Bayle, 2013) However because the program is counting and not delayed other code can be compiled as it counts meaning the keypad would always work for example instead of being blocked by a total delay.

C++ is a powerful and efficient high-level programming language used across all sorts of applications capable of object orientation, polymorphism, inheritance etc. (Meyers, 2014) However, within Arduino the language is somewhat limited and doesn't include all the characteristics available with C++ simply because the Arduino single board computer doesn't have the RAM to complete these complex tasks and doesn't need to in most cases meaning the language on Arduino is in essence its own version of C++. If this alarm were to be implemented in a real world situation it might be more prudent to use C code which is a lower level language which will take less processing power and possibly use a smaller microcontroller for more efficient use of physical space within the basic alarm design outlined in the brief. (King, 2008)

References

Bayle, J., 2013. *C Programming for Arduino*. 2nd ed. Paris: Packt.
Grimes, R., 2017. *Beginning C++ Programming*. 2nd ed. Birmingham: Packt.
James, M., 2013. *Higher Electronics*. 2nd ed. London: Routledge.
King, K., 2008. *C Programming: A Modern Approach*. 2nd ed. California: Norton.
Meyers, S., 2014. *Effective Modern C++*. 2nd ed. Sebastopol, CA: O'Reily.

Bibliography

Bayle, J., 2013. *C Programming for Arduino*. 2nd ed. Paris: Packt.
Grimes, R., 2017. *Beginning C++ Programming*. 2nd ed. Birmingham: Packt.
James, M., 2013. *Higher Electronics*. 2nd ed. London: Routledge.
King, K., 2008. *C Programming: A Modern Approach*. 2nd ed. California: Norton.
Meyers, S., 2014. *Effective Modern C++*. 2nd ed. Sebastopol, CA: O'Reily.

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 10/ 15 Good knowledge demonstrated of the hardware used with clear coherent reasons given for choices made. Software design 12/15 Flowchart acknowledges setup code is required, but does not cover what this is. Appropriate shapes are used for flowchart and it is logically sound. The inclusion of pseudo code strengthens this section and covers the setup as well as suggests additional functions that could be used. It also identifies constants and variables you may be using. However, for pseudocode code block brackets and semicolons are not needed. You have a good coverage of how the software operates the hardware.	30	22
3. Cognitive and Intellectual Skills		
Discussion, evaluation and conclusion An excellent evaluation of the underlying concepts and principles, you have carefully considered the effectiveness of your solution in different scenarios, given some good suggestion on how to resolve issues and given a valid discussion of the usefulness of C programming language for an embedded system.	15	11
4. Practical Application Skills		
Implementation 33/40 Code is well written with self-annotating variables, use of constants where appropriate, and commented. I would consider using either const or #define when declaring variables, rather than both. Indentation of code blocks could be more pronounced. You have gained further marks in this section for the inclusion of useful functions.	50	41

testing 8/10 The inclusion of a test outside just outside the 2m range on the boundary and just inside the 2m is a useful boundary test to ensure you program operates to the given brief. You have also considered testing what would happen if the alarm is triggered at the 2m and then the intruder moves out of range, overall this is a very well covered section, well done!		
5. Transferable Skills for Life and Professional Practice		
Structure of report Work is coherent, very fluent and is presented professionally. Datasheets for the components selected are included in the appendices . Code layout is indented on the most part correctly and makes good use of self-annotation techniques.	5	4

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)		78%
	Up to 1 week late (40% Max)		
	Over 1 week late (0%)		