



Embedded Systems

HNC ELECTRICAL/ELECTRONIC ENGINEERING, POWER SYSTEMS
ENGINEERING & INSTRUMENTATION ENGINEERING
STUDENT | SIAN NUNNERY

LECTURER | Sarah Williams

Contents

Introduction.....	2
Task 1 – Hardware Design	3
Task 2 – Software Design.....	7
Task 3 – Implementation.....	8
Task 4 – Testing	13
Task 5 – Discussion, Evaluation and Conclusion	15
References	16
Bibliography.....	16

Introduction

The purpose of this assignment is to design and emulate an alarm system, where it will trigger an alarm if an object is detected within a 2m proximity of the sensor.

The micro-controller-based system will indicate the object within the sensor range, as well as LED (Light Emitting diode) indication of the object being detected.

Throughout this assignment discussion of the hardware and software design will be included, as well as the implementation. Then the software will be tested and discuss any faults with the software, then I will carry out an evaluation of the overall outcome of this assignment.

Task 1 – Hardware Design

Within the design of the microcontroller-based system, the components that have been chosen are an ultrasonic distance sensor, resistors, LEDs, an LCD, a 4x4 KeyPad, breadboard and an Arduino Uno board.

A microcontroller is an integrated circuit device which is used to control elements of an electrical system, this is usually through a microprocessor unit, some peripherals and memory.

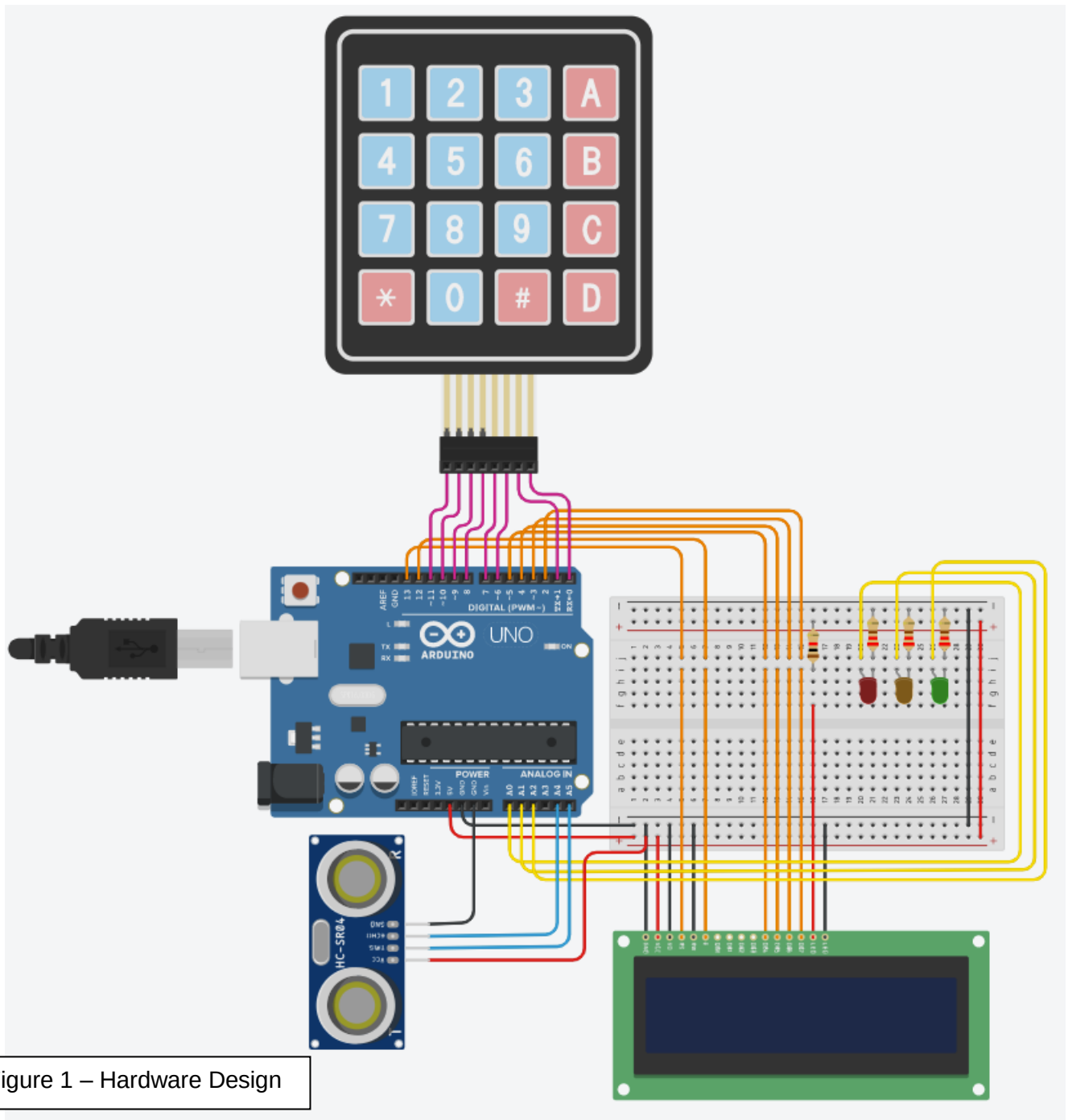


Figure 1 – Hardware Design

Ultrasonic Distance Sensor, this is made up of two main components, these include a transmitter and a receiver. The transmitter emits sound by using piezoelectric crystals, whereas the receiver is the recipient of the sound after it has travelled from the transmitter and back off the target. This sensor measures distance by emitting ultrasonic sound waves, which then converts the reflected ultrasonic sound waves into an electric signal. The distance is measure by using the following formula: $D = \frac{1}{2}T \times C$, where D is for distance, T is time and C is the speed of sound (~343meters/second).

A **Resistor** is a two-terminal passive electrical component to create resistance in the flow of an electrical current. Resistors are used for many different purposes such as to reduce the current flow, to divide voltages, adjust signal levels, heat generation, terminate transmission lines, bias active elements, matching and loading circuits as well as other uses. Resistance is measured in ohms; this is represented by ohms law: $R = \frac{V}{I}$. *“Ohm’s law states that the electrical current through a conductor is proportional to the potential difference across it. Furthermore, the electrical resistance of the conductor is constant.”* (eepower, n.d.)

LCD, a Liquid Crystal Display is a flat display screen, commonly used for technology such as mobile screens, TVs, computer monitors, laptops etc.

The working principle of an LCD is that they block light rather than emitting light. It is different to an LED where it emits light, an LCD uses liquid crystals to create a backlight to produce an image.

To be able to initialise the interface and use of a LCD screen, parameters need to be put in place, as well as specifying the dimensions such as width and height of the display screen.

Lcd: is a type of variable for LiquidCrystal

Cols: defining the number of columns that the display screen has

Rows: defining the number of rows that the display screen has.

Within the setup of the 'lcd.begin(cols,rows)' is used to define the above.

LiquidCrystal is a function (#include <LiquidCrystal.h>) that can be added to your code via the library, this allows the Arduino board to control the LCD screen. This is based on Hitachi HD44780 chipset, this is a commonly found on most text-based LCD codes. The library either

works with 4 or 8 bits, this uses either 4 or 8 data lines, as well as using the RS (Register Select), enable, contrast, Read/Write.

LED, Light Emitting Diode is a bulb that produces light. It produces light by passing an electrical current through a semiconductor, which then emits the light. Electrons within the semiconductor which recombine with the electron holes, releasing photons which is a form of energy. The semiconductor used within an LED is a diode. The light produced from an LED is forward directional lighting, not like a conventional light bulb which is Omni directional.

Light Emitting Diodes are a type of semiconductor light sources as previously stated. This combines a P-type with an N-type semiconductor. A P-type has a larger hole concentration, and an N-type has a larger electron concentration. By applying enough forward voltage will then cause the holes and electrons to recombine at the P-N junction, which then releases energy to form a light.

KeyPad 4x4 is used for entering numbers, letters, characters into a microcontroller, a 4x4 keypad consists of 4 rows and 4 columns and 16 pushbuttons that are built in. Each button acts as a switch as there is a membrane switch underneath the keypad.

Arduino Uno - “Arduino is an open-source platform used for building electronics projects. Arduino consists of both a physical programmable circuit board (often referred to as a microcontroller) and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the physical board.” (What is an Arduino? - learn.sparkfun.com, n.d.)

Features of Arduino Uno include:

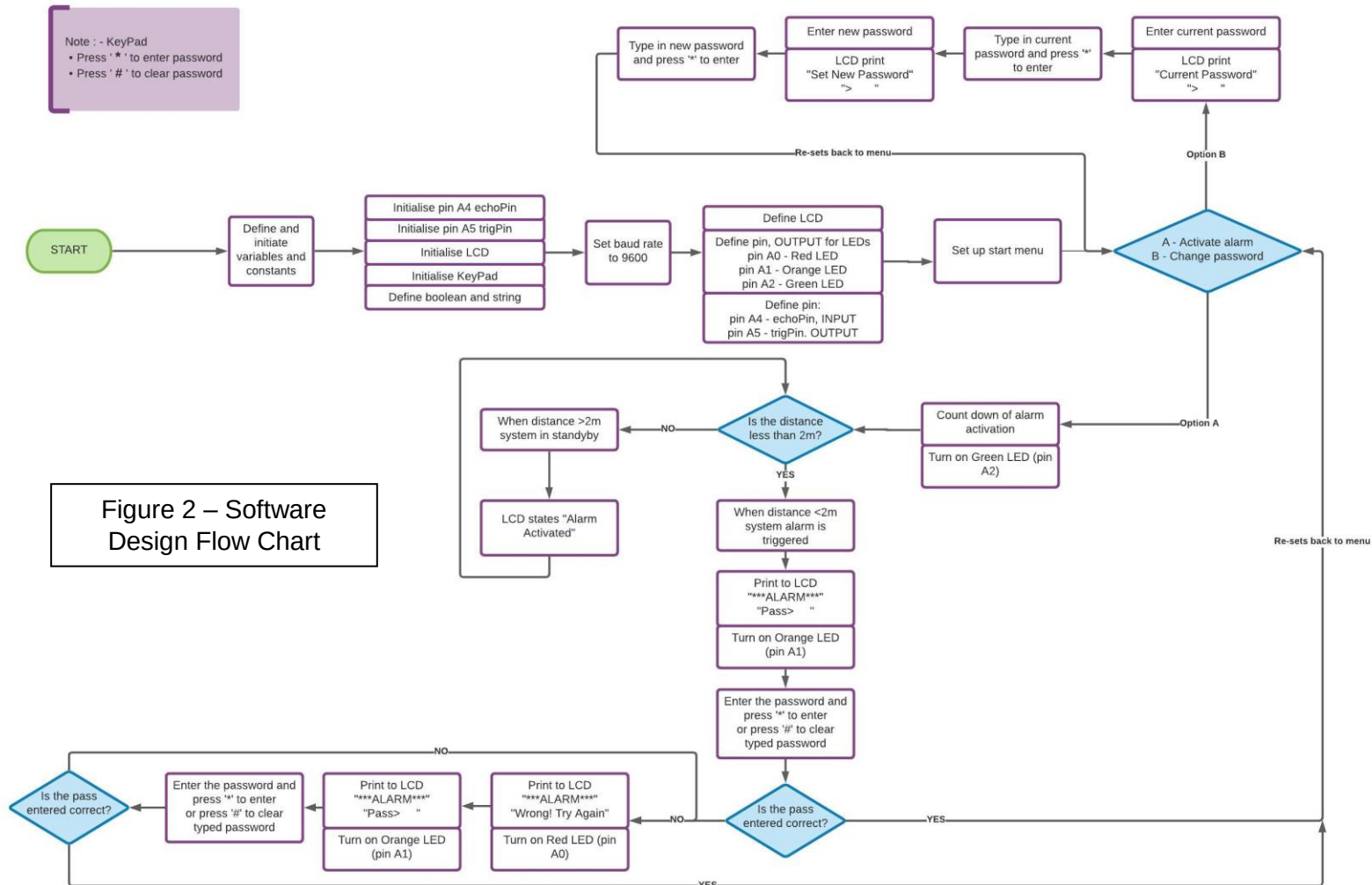
- *“Microcontroller: ATmega328p*
- *Operating Voltage: 5V*
- *Input Voltage (recommended): 7-12V*
- *Input voltage (limit): 6-20V*
- *Digital I/O Pins: 14 (of which 6 provide PWM output)*
- *PWM Digital I/O Pins: 6*
- *Analog input Pins: 6*

- *DC Current per I/O Pin: 20mA*
- *DC Current for 3.3V Pin: 50mA*
- *Flash Memory: 32KB (ATmega328P) of which 0.5KB used by bootloader.*
- *SRAM: 2KB (ATmega328P)*
- *EEPROM: 1KB (ATmega328P)*
- *Clock Speed 16MHz*
- *LED_BUILTIN: 13*
- *Length: 68.6mm*
- *Width: 53.4mm*
- *Weight: 25g”* (Arduino Uno Rev3 | Arduino Official Store, n.d.)

Please see appendix 1 and 2 for Arduino Uno datasheets

Task 2 – Software Design

To show how the software will control the hardware in the system, this has been illustrated by a flow chart as shown in Figure 2 or Appendix 3. The flow chart helps show the breakdown and order of the code, this shows what the code translates to.



See appendix 3 for PDF copy of flowchart.

Task 3 – Implementation

The following set of images which illustrate the C code produced to implement the system, this also includes comments throughout:

```
1 #include <LiquidCrystal.h> // includes the LiquidCrystal Library
2 #include <Keypad.h> // includes Keypad Library
3 #define trigPin A5
4 #define echoPin A4
5 long microseconds;
6 int distance, initialDistance, currentDistance, i;
7 int screenOffMsg = 0;
8 String password = "1234"; //Setting up startup/default password
9 String tempPassword;
10 boolean activated = false; // State of the alarm
11 boolean isActivated;
12 boolean activateAlarm = false;
13 boolean alarmActivated = false;
14 boolean enteredPassword; // State of the entered password to stop the alarm
15 boolean passChangeMode = false;
16 boolean passChanged = false;
17 const byte ROWS = 4; //four rows
18 const byte COLS = 4; //four columns
19 char keypressed;
20 //define the symbols on the buttons of the keypad
21 char keyMap[ROWS][COLS] = {{ '1', '2', '3', 'A' }, { '4', '5', '6', 'B' }, { '7', '8', '9', 'C' }, { '*', '0', '#', 'D' }};
22 byte rowPins[ROWS] = {11,10,9,8}; //Row pinouts of the keypad
23 byte colPins[COLS] = {7,6,1,0}; //Column pinouts of the keypad
24 Keypad myKeypad = Keypad( makeKeymap(keyMap), rowPins, colPins, ROWS, COLS);
25 LiquidCrystal lcd(13,12,5,4,3,2); // Creates an LC object. Parameters: (rs, enable, d4, d5, d6, d7)
26
27 const int RED_LED=A0; //Defines pin for LEDs
28 const int ORG_LED=A1;
29 const int GRN_LED=A2;
30
31 void setup() {
32     lcd.begin(16,2);
33     pinMode(trigPin, OUTPUT); // Sets the trigPin as an Output
34     pinMode(echoPin, INPUT); // Sets the echoPin as an Input
35     pinMode(RED_LED, OUTPUT); // Sets the Red LED pin as an Output
36     pinMode(ORG_LED, OUTPUT); // Sets the Orange LED pin as an Output
37     pinMode(GRN_LED, OUTPUT); // Sets the Green LED pin as an Output
38 }
39 void loop() {
40     if (activateAlarm) {
41         lcd.clear();
42         lcd.setCursor(0,0);
43         lcd.print("Alarm will be");
44         lcd.setCursor(0,1);
45         lcd.print("activated in");
46
47         int countdown = 9; // 9 seconds count down before activating the alarm
48         while (countdown != 0) {
49             lcd.setCursor(13,1);
50             lcd.print(countdown);
51             countdown--;
52             delay(1000);
53         }
54         lcd.clear();
55         lcd.setCursor(0,0);
56         lcd.print("Alarm Activated!");
57     }
```

```

57     initialDistance = getDistance();
58     activateAlarm = false;
59     alarmActivated = true;
60     digitalWrite(GRN_LED,HIGH);
61 }
62 if (alarmActivated == true){
63     currentDistance = getDistance();
64     if (distance>0 && distance<200) {
65         lcd.clear();
66         digitalWrite(GRN_LED,LOW);
67         enterPassword();
68     }
69 }
70 if (!alarmActivated) {
71     if (screenOffMsg == 0 ){
72         lcd.clear();
73         lcd.setCursor(0,0);
74         lcd.print("A - Activate");
75         lcd.setCursor(0,1);
76         lcd.print("B - Change Pass");
77         screenOffMsg = 1;
78     }
79     keypressed = myKeypad.getKey();
80     if (keypressed == 'A'){ //If A is pressed, activate the alarm
81         activateAlarm = true;
82     }
83     else if (keypressed == 'B') { //If B is pressed, chnage password
84         lcd.clear();
85         int i=1;
86         tempPassword = "";
87         lcd.setCursor(0,0);
88         lcd.print("Current Password");
89         lcd.setCursor(0,1);
90         lcd.print(">");
91         passChangeMode = true;
92         passChanged = true;
93         while(passChanged) {
94             keypressed = myKeypad.getKey();
95             if (keypressed != NO_KEY){
96                 if (keypressed == '0' || keypressed == '1' || keypressed == '2' || keypressed == '3' ||
97                     keypressed == '4' || keypressed == '5' || keypressed == '6' || keypressed == '7' ||
98                     keypressed == '8' || keypressed == '9' ) {
99                     tempPassword += keypressed;
100                     lcd.setCursor(i,1);
101                     lcd.print(tempPassword);
102                     i++;
103                 }
104             }
105             if (i > 5 || keypressed == '#') {
106                 tempPassword = "";
107                 i=1;
108                 lcd.clear();
109                 lcd.setCursor(0,0);
110                 lcd.print("Current Password"); //Type in current password, default password = 1234
111                 lcd.setCursor(0,1);
112                 lcd.print(">");

```

```

113     }
114     if ( keypressed == '*' ) {
115         i=1;
116         if (password == tempPassword) {
117             tempPassword="";
118             lcd.clear();
119             lcd.setCursor(0,0);
120             lcd.print("Set New Password");           //Enter new password
121             lcd.setCursor(0,1);
122             lcd.print(">");
123             while(passChangeMode) {
124                 keypressed = myKeypad.getKey();
125                 if (keypressed != NO_KEY){
126                     if (keypressed == '0' || keypressed == '1' || keypressed == '2' || keypressed == '3' ||
127                         keypressed == '4' || keypressed == '5' || keypressed == '6' || keypressed == '7' ||
128                         keypressed == '8' || keypressed == '9' ) {
129                         tempPassword += keypressed;
130                         lcd.setCursor(i,1);
131                         lcd.print("*");           //LCD to print '*' when password is typed
132                         i++;
133                     }
134                 }
135                 if (i > 5 || keypressed == '#') {           //if '#' pressed clear password typed
136                     tempPassword = "";
137                     i=1;
138                     lcd.clear();
139                     lcd.setCursor(0,0);
140                     lcd.print("Set New Password");
141                     lcd.setCursor(0,1);
142                     lcd.print(">");
143                 }
144                 if ( keypressed == '*' ) {           //if '*' pressed enter password typed
145                     i=1;
146                     password = tempPassword;
147                     passChangeMode = false;
148                     passChanged = false;
149                     screenOffMsg = 0;
150                 }
151             }
152         }
153     }
154 }
155 }
156 }
157 }
158 void enterPassword() {
159     int k=5;
160     tempPassword = "";
161     activated = true;           //Alarm triggered = true
162     lcd.clear();
163     lcd.setCursor(0,0);
164     lcd.print(" *** ALARM *** ");           //LCD message 'ALARM'
165     lcd.setCursor(0,1);
166     lcd.print("Pass>");           //LCD message to enter Password
167     digitalWrite(ORG_LED,HIGH);           //Orange LED switches on

```

```

168 while(activated) {
169     keypressed = myKeypad.getKey();
170     if (keypressed != NO_KEY){
171         if (keypressed == '0' || keypressed == '1' || keypressed == '2' || keypressed == '3' ||
172             keypressed == '4' || keypressed == '5' || keypressed == '6' || keypressed == '7' ||
173             keypressed == '8' || keypressed == '9' ) {
174             tempPassword += keypressed;
175             lcd.setCursor(k,1);
176             lcd.print(""+keypressed);
177             k++;
178         }
179     }
180     if (k > 9 || keypressed == '#') { // '#' clears password and resets the LCD screen to re-type password
181         tempPassword = "";
182         k=5;
183         lcd.clear();
184         lcd.setCursor(0,0);
185         lcd.print(" *** ALARM *** ");
186         lcd.setCursor(0,1);
187         lcd.print("Pass>");
188     }
189     if ( keypressed == '*' ) {
190         if ( tempPassword == password ) {
191             activated = false;
192             alarmActivated = false;
193             screenOffMsg = 0;
194         }
195         else if (tempPassword != password) { //if password entered is wrong:
196             lcd.setCursor(0,1);
197             lcd.print("Wrong! Try Again"); //lcd message states error
198             digitalWrite(REDA_LED,HIGH); //Red LED switches on
199             delay(2000);
200             lcd.clear();
201             digitalWrite(REDA_LED,LOW); //Red LED switches off
202             lcd.setCursor(0,0);
203             lcd.print(" *** ALARM *** ");
204             lcd.setCursor(0,1);
205             lcd.print("Pass>"); //Re-enter password
206         }
207     }
208 }
209 digitalWrite (ORGL_LED,LOW); //Orange LED switches off
210 }
211
212 long getDistance(){ // Custom function for the Ultrasonic sensor
213     digitalWrite(trigPin, LOW); // Clears the trigPin
214     delayMicroseconds(2);
215     digitalWrite(trigPin, HIGH); // Sets the trigPin on HIGH state for 10 microseconds
216     delayMicroseconds(10);
217     digitalWrite(trigPin, LOW);
218     microseconds = pulseIn(echoPin, HIGH); // Reads the echoPin, returns the sound wave travel time in microseconds
219     distance = microseconds/58; // Calculating the distance
220     delay(500);
221
222     return distance;
223 }

```

Using Boolean logic in my code, allowed me to be able to use just one of two values – true and false. I used this in my code to define whether the alarm was active, as well as for activating the alarm, these functions are either true or false, therefore by using Boolean allowed me to set these two values.

By using a string function allowed me to create an array of characters, I used this to set up my password, as well as enabling the user to change their password.

Throughout my code I have used 'if' statements, by using this statement allows the simulation to decide where the statement is true or false, if the statement is false then the statement defined in the 'if' is skipped. If and else are used in code for decision control of programming structure.

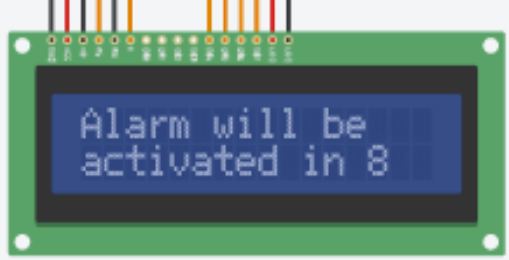
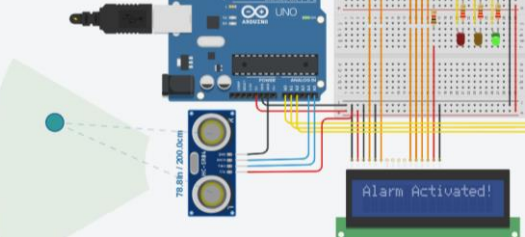
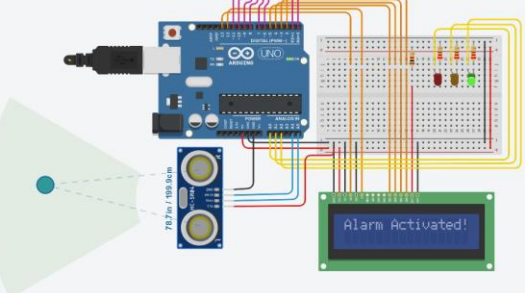
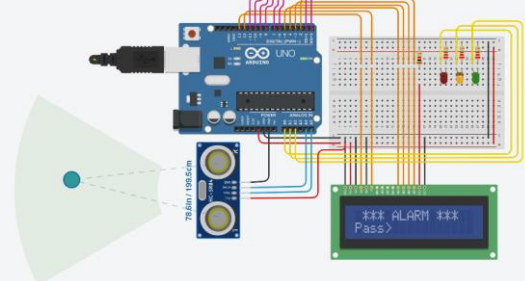
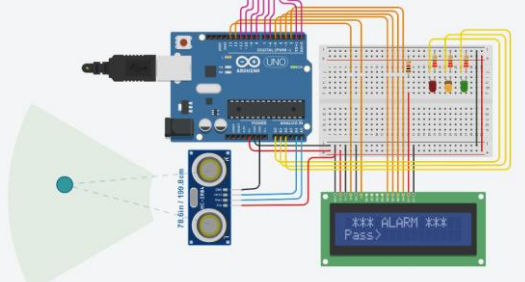
Within the code there are 'while' loop statements these are used to repeat a block of code until the condition of that statement becomes false, and these are given a Boolean condition. For example, until the statement is either true or false / 0 or 1.

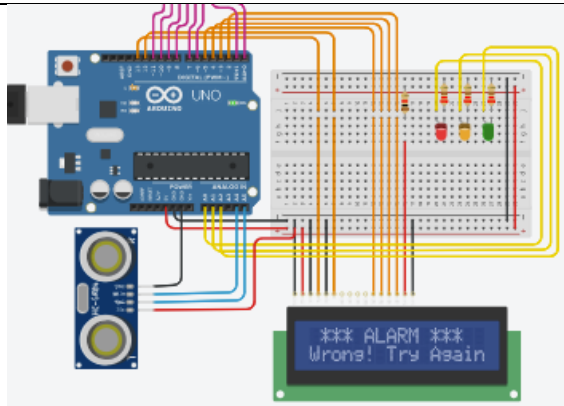
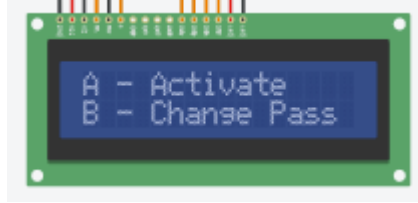
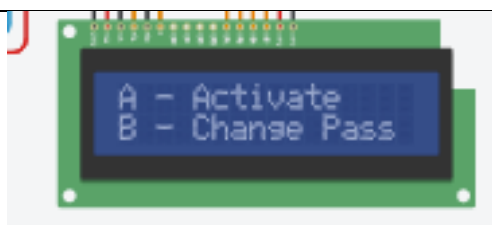
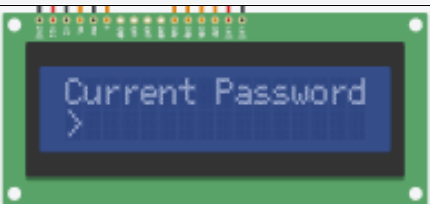
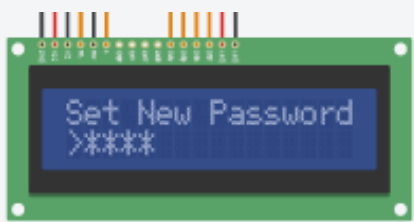
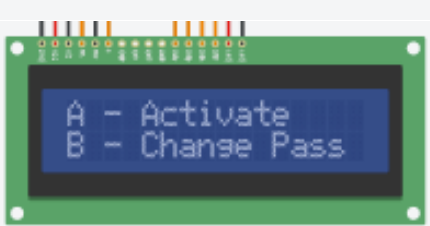
Using 'int' in C programming is short for integer, this is a variable used to define numeric values.

By using #define in C programming, allows the macro definition to be a constant value that will be declared throughout the code. These definitions are not variables, therefore cannot be changed within the code like variables.

Within the code various data types have been used such as int, char and long, these are used during variable declaration. Whereas const is used to define a variable.

Task 4 – Testing

Test	Testing	Expected Output	Actual Output	Pass / Fail	Issues
1	Option A selected, count down	Once option A is selected on the start up menu, a count down from 9 will begin LCD print 'Alarm will be activated in 'X''		Pass	None
2	Alarm not triggered at 2m	LCD print 'Alarm Activated' Green LED		Pass	None
3	Alarm triggered at 1.99m	LCD print '***ALARM***' 'Pass>.....' Orange LED		Fail	Alarm not triggered at 1.99m
4	Alarm triggered (1.95m)	LCD print '***ALARM***' 'Pass>.....' Orange LED		Pass	None
5	Alarm triggered at 1.98m	LCD print '***ALARM***' 'Pass>.....' Orange LED		Pass	None

6	Password entered incorrect	LCD print ‘***ALARM***’ ‘Wrong! Try again’ Orange and Red LED		Pass	None
7	Password entered when alarm triggered correct	Once correct password is entered, LCD to go back to start up menu		Pass	None
8	Start up menu to appear at the start of simulation	LCD print ‘A – Activate’. ‘B – Change Pass’		Pass	None
9	Option B to change password	-Select option B -LCD print ‘Current Password >’ -Type default password 1234, then ‘*’ to enter -LCD print ‘Set New Password>’ -Type new 4-digit password and press ‘*’ to enter -Back to start-up menu	  	Pass	None

Task 5 – Discussion, Evaluation and Conclusion

Overall, the implementation of the system was mostly successful. Within the testing, test 3 failed, where the alarm was not triggered at 1.99m. After some research and discussion with my lecturer, this error could be down to the simulation software, due to a rounding error of the code and/or ultrasonic distance sensor or the communication between the two. But as illustrated in the testing table in Task 4, the alarm was triggered at 1.98m and 1.95m in test 4 and 5.

C programming language was developed by Dennis Ritchie between 1969 and 1976, which is a high-level programming language that is ideal for firmware developing and products used for portable purposes. It is typically used on a processor which has an operating system installed, it is designed to run a system which has high memory. When this program is compiled, its operating system is on a dependent file. In comparison to other high-level languages, C programming offers more flexibility as it is a somewhat small and structured language. It is also able to support low level bit wise data manipulation. Furthermore, C programming is reliable and scalable, as well as portable between various platforms.

The requirement of this assignment was to create an alarm system where the alarm was triggered within a 2m proximity with LED indication. Though I took this further and developed the code with a start up menu to allow for activation of this alarm, as well as the option to change the password to deactivate the alarm. However, if I were to do this assignment again, I would have a limit on the password attempts when attempting to deactivate the alarm when triggered. I would also add a buzzer to the system to alert the intruder to hopefully deter them away from the property, as well as the property owner to inform them that the alarm itself has triggered. Additionally, I would have added admin access, so the user is able to adjust the range of the sensor, for example whether they want the alarm to be triggered at 1m, 3m or 4m instead of 2m. As well as adding multiple user access passwords, so if the sensor was placed at an office, multiple people could have different access code to deactivate the alarm.

Since we do not know the circumstances the sensor would be placed in, we do not know if there is enough span on the sensor to cover all possible angles that an intruder could approach from. Therefore, the intruder could avoid the sensor through possible blind spots, if knowing these parameters then I would either add additional sensors or choose a different sensor with a different range.

References

eeperpower. n.d. [online] Available at: <<https://eeperpower.com/resistor-guide/resistor-fundamentals/ohms-law/>> [Accessed 18 May 2021].

Learn.sparkfun.com. n.d. *What is an Arduino? - learn.sparkfun.com*. [online] Available at: <<https://learn.sparkfun.com/tutorials/what-is-an-arduino/all>> [Accessed 18 May 2021].

Store.arduino.cc. n.d. *Arduino Uno Rev3 | Arduino Official Store*. [online] Available at: <<https://store.arduino.cc/arduino-uno-rev3>> [Accessed 18 May 2021].

Bibliography

WhatIs.com. n.d. *What is LCD (Liquid Crystal Display)?*. [online] Available at: <<https://whatis.techtarget.com/definition/LCD-liquid-crystal-display>> [Accessed 18 May 2021].

Arduino.cc. n.d. *Arduino - LiquidCrystalBegin*. [online] Available at: <<https://www.arduino.cc/en/Reference/LiquidCrystalBegin>> [Accessed 18 May 2021].

Rohm.com. n.d. *Light Emitting Diode <What are LEDs and How Do They Work? > | Electronics Basics | ROHM*. [online] Available at: <<https://www.rohm.com/electronics-basics/leds/what-are-leds>> [Accessed 18 May 2021].

Industry Today. n.d. *LED Lighting: Its Function and Advantages - Industry Today %*. [online] Available at: <<https://industrytoday.com/led-lighting-its-function-and-advantages/>> [Accessed 18 May 2021].

eeperpower. n.d. [online] Available at: <<https://eeperpower.com/resistor-guide/resistor-fundamentals/ohms-law/>> [Accessed 18 May 2021].

Pattabiraman, K., n.d. *How to Set Up a Keypad on an Arduino*. [online] Circuit Basics. Available at: <<https://www.circuitbasics.com/how-to-set-up-a-keypad-on-an-arduino/>> [Accessed 18 May 2021].

Learn.sparkfun.com. n.d. *What is an Arduino? - learn.sparkfun.com*. [online] Available at: <<https://learn.sparkfun.com/tutorials/what-is-an-arduino/all>> [Accessed 18 May 2021].

beginnersbook.com. n.d. *Function call by reference in C Programming*. [online] Available at: <<https://beginnersbook.com/2014/01/c-function-call-by-reference-example/>> [Accessed 18 May 2021].

FierceElectronics. n.d. *What is an Ultrasonic Sensor?*. [online] Available at: <[https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor#:~:text=An%20ultrasonic%20sensor%20is%20an,sound%20that%20humans%20can%20hear\).>](https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor#:~:text=An%20ultrasonic%20sensor%20is%20an,sound%20that%20humans%20can%20hear).>)> [Accessed 18 May 2021].

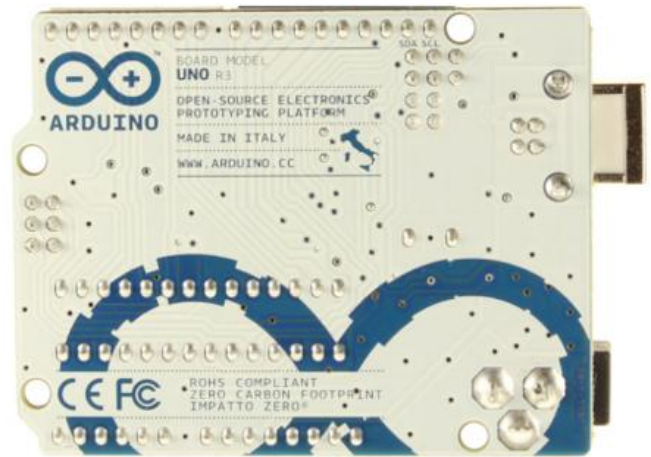
Store.arduino.cc. n.d. *Arduino Uno Rev3 | Arduino Official Store*. [online] Available at: <<https://store.arduino.cc/arduino-uno-rev3>> [Accessed 18 May 2021].

Microcontrollers Lab. n.d. *What is embedded c? Difference between C and Embedded C*. [online] Available at: <<https://microcontrollerslab.com/difference-c-language-embedded-c/>> [Accessed 18 May 2021].

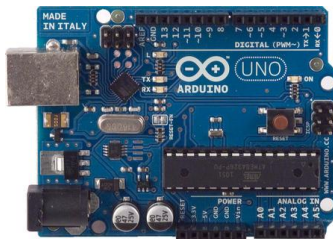
Arduino Uno



Arduino Uno R3 Front



Arduino Uno R3 Back



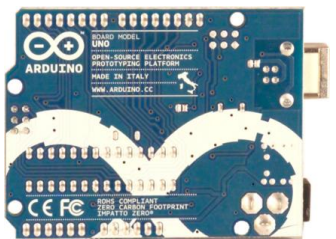
Arduino Uno R2 Front



Arduino Uno SMD



Arduino Uno Front



Arduino Uno Back

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 ([datasheet](#)). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

| [Revision 2](#) of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into [DFU mode](#).

| [Revision 3](#) of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the [index of Arduino boards](#).

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V

Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic & Reference Design

EAGLE files: [arduino-uno-Rev3-reference-design.zip](#) (NOTE: works with Eagle 6.0 and newer)

Schematic: [arduino-uno-Rev3-schematic.pdf](#)

Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the [EEPROM library](#)).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using [pinMode\(\)](#), [digitalWrite\(\)](#), and [digitalRead\(\)](#) functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the [attachInterrupt\(\)](#) function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the [analogWrite\(\)](#) function.

- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the [SPI library](#).
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though it is possible to change the upper end of their range using the AREF pin and the [analogReference\(\)](#) function. Additionally, some pins have specialized functionality:

- **TWI: A4 or SDA pin and A5 or SCL pin.** Support TWI communication using the [Wire library](#).

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with [analogReference\(\)](#).
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the [mapping between Arduino pins and ATmega328 ports](#). The mapping for the Atmega8, 168, and 328 is identical.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, [on Windows, a .inf file is required](#). The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A [SoftwareSerial library](#) allows for serial communication on any of the Uno's digital pins.

The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the [documentation](#) for details. For SPI communication, use the [SPI library](#).

Programming

The Arduino Uno can be programmed with the Arduino software ([download](#)). Select "Arduino Uno" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the [reference](#) and [tutorials](#).

The ATmega328 on the Arduino Uno comes preburned with a [bootloader](#) that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol ([reference](#), [C header files](#)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see [these instructions](#) for details.

The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available. The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use [Atmel's FLIP software](#) (Windows) or the [DFU programmer](#) (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See [this user-contributed tutorial](#) for more information.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload. This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see [this forum thread](#) for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

Arduino(™) UNO Rev3

The schematic diagram illustrates the internal circuitry of the Arduino(™) UNO Rev3 board. It features two main microcontrollers: the ATmega16U2-MU(R) and the ATmega328P-PU. The ATmega16U2-MU(R) handles USB communication, while the ATmega328P-PU provides the core processing capabilities. The power management section includes a USB-to-VCC converter (F1), a 5V LDO (NCP1117ST50T3G), and a 3.3V LDO (LP2985-33DBVR). The I/O section includes a serial-to-USB bridge (ICSP1), a serial-to-I2C bridge (ICSP), and various peripheral components like LEDs, relays, and sensors. The diagram also shows the connection of external components like the USB cable (X2) and the ATmega328P-PU package (ZU4).

Reference Designs ARE PROVIDED "AS IS" AND "WITH ALL FAULTS," ARDUINO SA DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

ARDUINO SA MAY MAKE CHANGES TO SPECIFICATIONS AND PRODUCT DESCRIPTIONS AT ANY TIME, WITHOUT NOTICE. THE CUSTOMER MUST NOT RELY ON THE ABSENCE OR CHARACTERISTICS OF ANY FEATURES OR INSTRUCTIONS MARKED "RESERVED" OR "UNDEFINED". ARDUINO SA RESERVES THESE FOR FUTURE DEFINITION AND SHALL HAVE NO RESPONSIBILITY WHATSOEVER FOR COMPATIBILITIES ARISING FROM FUTURE CHANGES TO THEM.

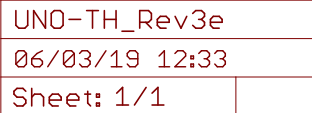
THE PRODUCT INFORMATION ON THE WEB SITE OR MATERIALS IS SUBJECT TO CHANGE WITHOUT NOTICE. DO NOT FINALIZE A DESIGN WITH THIS INFO.

ARDUINO AND OTHER ARDUINO BRANDS AND LOGOS ARE TRADEMARKS OF ARDUINO SA. ALL ARDUINO SA TRADEMARKS CANNOT BE USED WITHOUT OWNER'S FORMAL PERMISSION.

UNO-TH_Rev3e
06/03/19 12:33
Sheet: 1/1

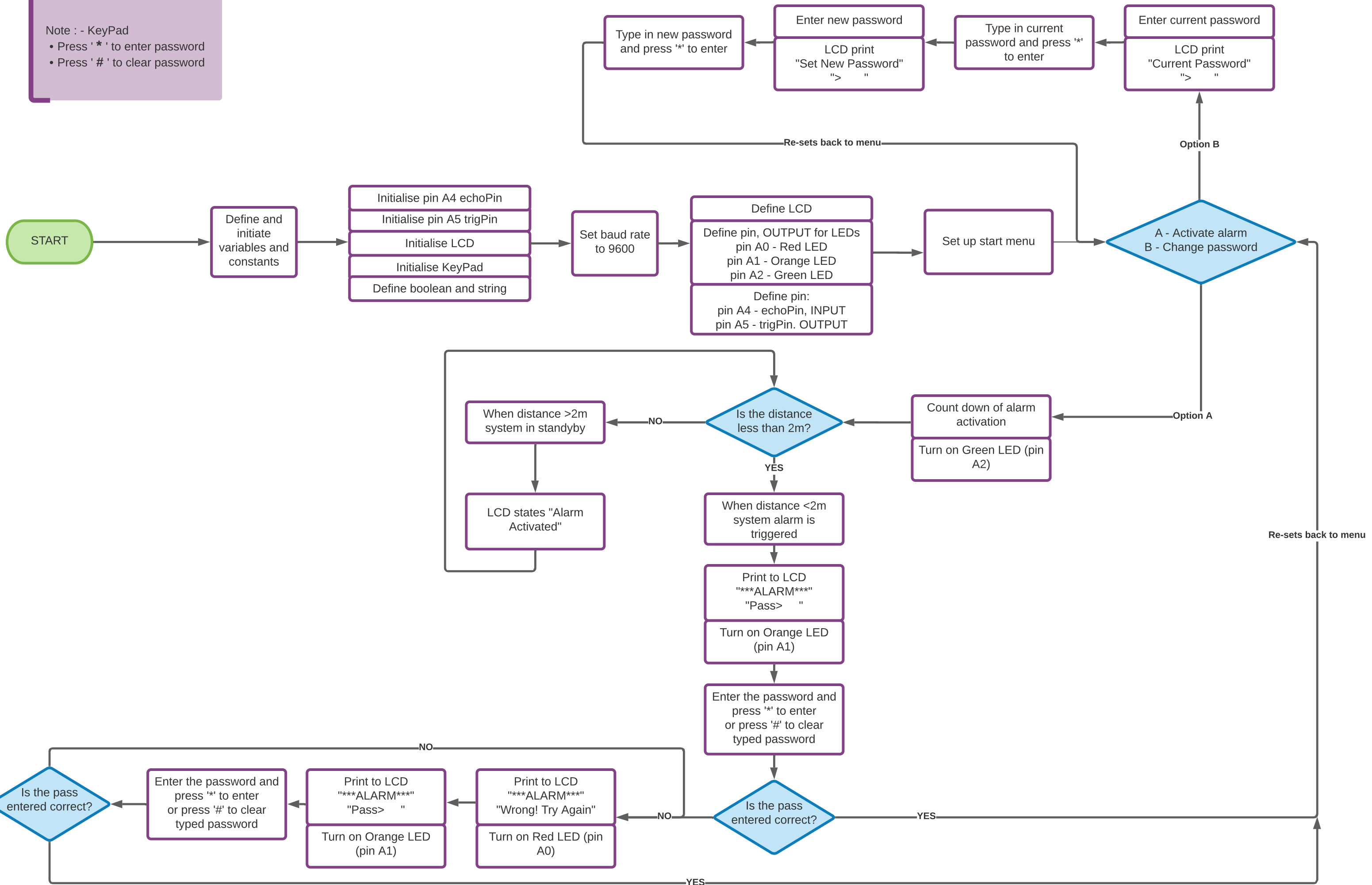
Arduino SA may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." Arduino SA reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this info

ARDUINO and other Arduino brands and logos and Trademarks of Arduino SA. All Arduino SA Trademarks cannot be used without owner's formal permission



Note : - Keypad

- Press ' * ' to enter password
- Press ' # ' to clear password



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 Excellent knowledge demonstrated of the hardware used in system, higher marks would be award here for an explanation why you chose these items and consideration of alternatives available. Software design 11/15 Flowchart is extremely detailed and acknowledges the setup code required. Appropriate shapes are used for flowchart and it is logically sound.	30	21
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 further improve your system, and given a valid discussion of the usefulness of C programming language for an embedded system.	15	12
4. Practical Application Skills		
Implementation 33/40 Code is well written with self-annotating variables, use of constants where appropriate, and commented. You have included some additional functionality to the project making it more versatile and useful. You might have considering checking if the character entered was in the range 0-9 rather than a lengthy or statement. Further marks have been gained in this section for the inclusion of useful functions. testing 7/10	50	40

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 tested all password features implemented.		
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 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)		77%
	Up to 1 week late (40% Max)		
	Over 1 week late (0%)		