

TC50321E Embedded systems

Assessment A1 – Controlling I/O Data in an Embedded System

BSc (Hons) Sound Engineering

21440503

Abstract

This report shows the complete design process of a prototype of an embedded system. The embedded system has three sensors that are controlled by an Arduino Uno board. They are displayed on an LCD. The report includes background information, design, code excerpts, and troubleshooting. Overall the project was a success, with the prototype eventually functioning in the way that was desired.

Contents

[Introduction 3](#_Toc102063230)

[Background Information 4](#_Toc102063231)

[Design 5](#_Toc102063232)

[C++ Code 6](#_Toc102063233)

[Testing/Evaluation 8](#_Toc102063234)

[Bibliography 10](#_Toc102063235)

[Appendix 1 11](#_Toc102063236)

[Figure 1 - Arduino Uno 4](https://uwloffice365live-my.sharepoint.com/personal/21440503_student_uwl_ac_uk/Documents/BSc%20Sound%20Engineering/Year%202/2.%20Embedded%20Systems/A1/21440503%20Assignment%20A1.docx#_Toc102058172)

[Figure 2 - TinkerCAD Design 5](https://uwloffice365live-my.sharepoint.com/personal/21440503_student_uwl_ac_uk/Documents/BSc%20Sound%20Engineering/Year%202/2.%20Embedded%20Systems/A1/21440503%20Assignment%20A1.docx#_Toc102058173)

[Figure 3 - Distance Sensor Troubleshooting 8](https://uwloffice365live-my.sharepoint.com/personal/21440503_student_uwl_ac_uk/Documents/BSc%20Sound%20Engineering/Year%202/2.%20Embedded%20Systems/A1/21440503%20Assignment%20A1.docx#_Toc102058174)

[Figure 4 - Temperature Sensor Display 9](https://uwloffice365live-my.sharepoint.com/personal/21440503_student_uwl_ac_uk/Documents/BSc%20Sound%20Engineering/Year%202/2.%20Embedded%20Systems/A1/21440503%20Assignment%20A1.docx#_Toc102058175)

[Figure 5 - Photoresistor Display 9](https://uwloffice365live-my.sharepoint.com/personal/21440503_student_uwl_ac_uk/Documents/BSc%20Sound%20Engineering/Year%202/2.%20Embedded%20Systems/A1/21440503%20Assignment%20A1.docx#_Toc102058176)

[Figure 6 - Ultrasonic Distance Sensor Display 9](https://uwloffice365live-my.sharepoint.com/personal/21440503_student_uwl_ac_uk/Documents/BSc%20Sound%20Engineering/Year%202/2.%20Embedded%20Systems/A1/21440503%20Assignment%20A1.docx#_Toc102058177)

# Introduction

This report details how a prototype I/O embedded system will be created using an Arduino Uno microprocessor board, C++ code and some sensors. Firstly, the prototype will be designed with the schematic creator TinkerCAD. This has a user-friendly interface that allows you to build your circuit in an intuitive way. A classic schematic is also created alongside. There is an IDE built in to TinkerCAD that allows you to type and load C++ code onto the prototype too. Then, the prototype will be constructed in a lab environment so it can be evaluated and seen in working operation.

There are four main chapters: Background Information; Design; C++ Code; and Testing/Evaluation.

# Background Information

An embedded system is a combination of hardware and software that takes an input from its environment, usually through sensors or human input, conducts operations with the data that it receives, and then provides an output through some kind of display, or through the motion of a motor or actuator. The core of an embedded system is a microprocessor chip. This is an integrated circuit that has the facility to have code loaded and stored on it. The microcontroller board used for this assignment is the Arduino Uno. The Uno board has 14 digital input/output pins (6 of which can be used as PWM outputs), 6 analogue inputs, a 16 MHz quartz crystal, a USB connection, a power jack, and a reset button (M, 2021).

Arduino uses a version of C++ as its native coding language. It has all of the language features of C++, but not all of the libraries. This is because of the memory limitations of Arduino boards. Another difference between regular C++ and the Arduino version is the layout in the IDE: the main() function is replaced with the repeating loop() function, and it is preceded by a one-time setup() function.

A close-up of a circuit board

Description automatically generated with medium confidence

Figure - Arduino Uno

# Design

For this assignment, three sensors will be used: an ultrasonic distance sensor; a photoresistor; and a temperature sensor.

The ultrasonic distance sensor operates by sending out an ultrasonic wave from its trigger pin and waiting for it to return at the echo pin. With some code, it can output a distance value in a specified unit. A photoresistor is sensitive to light, and its resistance value is affected by this. This consequently affects the voltage that is allowed to pass through. This voltage value can be processed to get a ‘light level’ value which can be outputted to the user. The temperature sensor uses the known resistances of materials such as platinum, and how the material’s resistance changes predictably with heat, to provide a temperature controlled resistor of sorts. It has three pins, but only the middle (V+) and right (V-) pin are used in this design. More accurate readings can be achieved using the third pin to offset the resistance of the wires themselves.

These sensors will be connected onto a breadboard, which will allow them to interface with the Arduino Uno. Also connected will be a 16x2 LCD (Liquid Crystal Display), which will provide the user with an output display. The LCD will display the output of one sensor at a time. A button will also be on the breadboard, allowing the user to go between sensor outputs on the display. Seen below is the TinkerCAD design schematic.

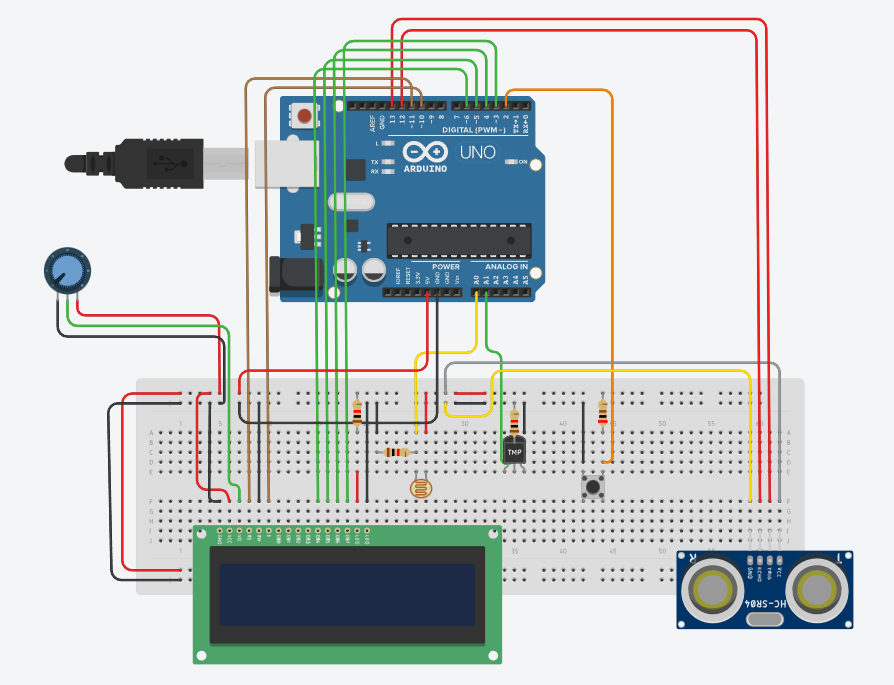


Figure - TinkerCAD Design

# C++ Code

#include <LiquidCrystal.h> //Liquid Crystal library allows use of LCD

Liquid Crystal lcd(11, 10, 6, 5, 4, 3); //assigning pins on Uno for LCD

#define trigPin 13

#define echoPin 12 //defining distance sensor's trigger and echo pin locations

#define LM335\_pin 1 //defining temperature sensor's pin

#define intPin 2 //defining the pin for the hardware interrupt

volatile long duration, distance; //defining two variables for the ultrasonic sensor, stored in RAM, and used with the interrupt

volatile int counter = 3; //initialising a variable for the interrupt's cycle

The section of code above comes before the setup() function. It includes libraries in the sketch; defines some of the Uno’s pins; and defines variables for some functions/interrupts.

void setup() {

lcd.begin(16, 2); //defining the size of the LCD

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT); //assigning pin mode to distance sensor's trigger and echo

}

Within setup(), the LCD size and the distance sensor’s pin I/O is defined.

void loop() {

attachInterrupt(0, Interrupt, LOW); //hardware interrupt (ID 0) occurs when the button's state is LOW

if (counter % 3 == 0) {

DistSens();

}

if (counter % 3 == 1) {

PhotoRes();

}

if (counter % 3 == 2) {

TempSens(); //if statements decide which function to run using the modulo operator

}

}

Within loop(), an interrupt is defined. Conditional statements decide which function displays on the LCD.

void Interrupt() {

counter++; //1 added to counter every time the button is pressed

}

The Interrupt() function adds one to the variable ‘counter’ every time it is called. This happens when the button on the breadboard is pressed and temporarily put into a LOW state.

void DistSens() { //function for using the ultrasonic distance sensor

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW); //sends a 10 second HIGH burst out of the trigger sensor (LOW, HIGH, LOW)

duration = pulseIn(echoPin, HIGH); //function detects the HIGH OUTPUT at pin 12 and assigns the value to the variable duration

distance = (duration / 2) / 29.1; //distance in cm is calculated with this line of code, assigned to distance variable

lcd.setCursor(0, 0); //positioning LCD cursor

lcd.print("Distance: "); // Prints string "Distance" on the LCD

lcd.setCursor(0, 1);

lcd.print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"); //print message

if (distance >= 200 || distance <= 0) {

lcd.setCursor(0, 1);

lcd.print("Out of Range! "); //statement is executed if object is out of sensor's range, set at 200cm

}

else {

lcd.setCursor(0, 1);

lcd.print(distance);

lcd.print(" cm"); // Prints the distance value from the sensor

}

delay(500);

}

This is the function for the distance sensor. It directs the sensor as to how it should send a pulse, and calculates the reflection distance/cm. It then displays this on the LCD.

void PhotoRes() { //function for using the photoresistor

lcd.setCursor(0, 0); //positioning LCD cursor

lcd.print("Light Level: "); //print message

int sensorValue = analogRead(A0); //assigning photoresistor output to a pin and reading value

lcd.setCursor(0, 1);

lcd.print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

lcd.setCursor(0, 1);

lcd.print(sensorValue);

delay(260);

}

The function for the photoresistor reads analogue pin A0 and outputs the value to the LCD.

void TempSens() { //function for using the temperature sensor

int Kelvin, Celsius; //defining two variables

delay(100);

Kelvin = analogRead(LM335\_pin) \* 0.489; //formula to calculate the Kelvin value from the LM335 produced voltage

Celsius = Kelvin - 273; //Kelvin value - 273 = Celsius value

lcd.setCursor(0, 0);

lcd.print("Temp/K:\_\_\_\_\_\_");

lcd.print(Kelvin); //prints Kelvin value to LCD

lcd.setCursor(0, 1);

lcd.print("Temp/C:\_\_\_\_\_\_\_");

lcd.print(Celsius); //prints Celsius value to LCD

}

The temperature sensor’s function reads the pin of the LM335 and calculates the temperature value in Kelvin and degrees Celsius. It displays this on the LCD. The temperature is calculated. A suitable current is received at the LM335 pin on the Arduino by an appropriate resistor used to divide the voltage. The 2k2 resistor used provides a 0.90mA current value with 5V from the Arduino. The component’s optimum currency is 1mA, as per the LM335 datasheet, see Appendix 1.

# Testing/Evaluation

When the design was complete, building (which includes testing) commenced.

One issue that was come across when the distance sensor was being evaluated was incorrect output. The sensor was outputting ‘0’ constantly. Initially, it was thought that maybe the code calculations were incorrect, but the error stopped happening after wiring the sensor’s VCC and GND pins closer to the Arduino’s on the breadboard. Also, when the VCC and GND rails on the breadboard were manually wired across (see top left corner of Figure T below), the problem seemed to stop occurring and the sensor behaved as expected.

Diagram

Description automatically generated

Figure - Distance Sensor Troubleshooting

Another issue that occurred was when the hardware interrupt changed the display on the LCD. If a string on row 1 of the LCD was longer than the one that followed when the interrupt happened, remnants of it would remain on the LCD, ruining the intelligibility of the display and the user experience. A simple fix for that was adding print messages that include underscores to overwrite old strings from the previous displays. See code excerpt below.

lcd.setCursor(0, 0); //positioning LCD cursor

lcd.print("Distance: "); // Prints string "Distance" on the LCD

lcd.setCursor(0, 1);

lcd.print("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"); //print message

if (distance >= 200 || distance <= 0) {

lcd.setCursor(0, 1);

lcd.print("Out of Range! "); //statement is executed if object is out of sensor's range, set at 200cm

}

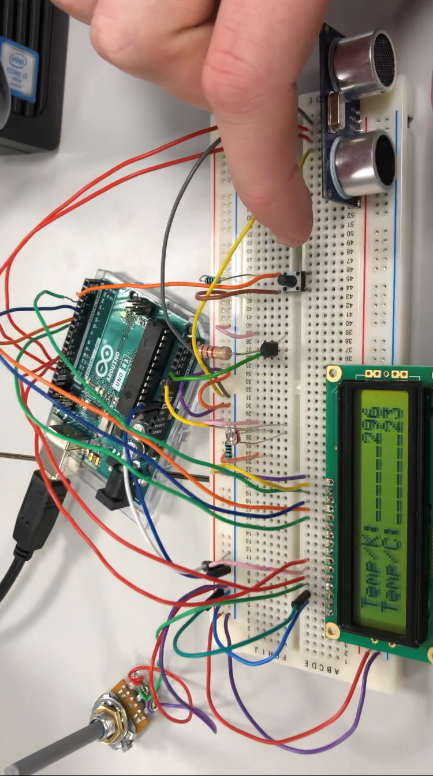
Seen below is the prototype displaying all three sensor outputs on the LCD. The button is used to cycle between the displays. Overall, the build process was frustrating at times, but the errors were smoothed out and the design was a success when it was finished. The most challenging aspect of the prototype build was synthesising smaller projects into one cohesive design that was multifaceted, but still efficient. The experience of the full design process was extremely rewarding.

Figure - Temperature Sensor Display

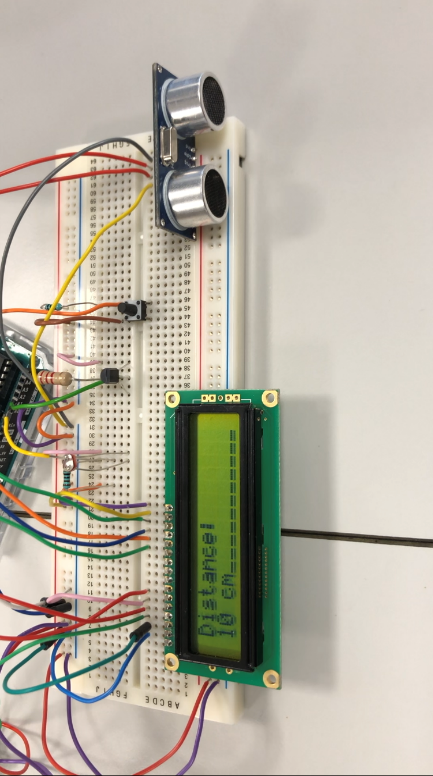
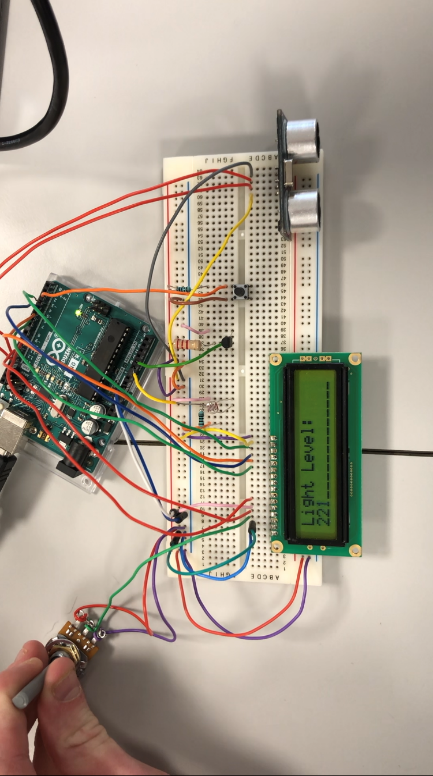


Figure - Photoresistor Display

Figure - Ultrasonic Distance Sensor Display

# Bibliography

M, L. (2021) *What is Arduino Language: Coding for Arduino Boards Explained.*Available at: <https://www.bitdegree.org/tutorials/what-is-arduino-language-coding-for-arduino-boards-explained/> (Accessed: .

# Appendix 1

<https://drive.google.com/file/d/0ByrcgOkOu0vgWng0ZmZQOTRCTk0/view?resourcekey=0-KQSPO166UZFlGV5yKBoEKg>